

THE NASTRAN[®] PROGRAMMER'S MANUAL
(Level 17.5)

December 1978 Edition

Reprinted August 1987



Scientific and Technical Information Office

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Washington, D.C.

INTRODUCTION

The Programmer's Manual is one of four manuals that constitute the documentation for NASTRAN, the other three being the Theoretical Manual, the User's Manual and the Demonstration Problem Manual. The Programmer's Manual is divided into seven major sections: Section 1, NASTRAN Programming Fundamentals; Section 2, Data Block and Table Descriptions; Section 3, Subroutine Descriptions; Section 4, Module Functional Descriptions; Section 5, NASTRAN - Operating System Interfaces; Section 6, Modifications and Additions to NASTRAN; and Section 7, NASTRAN Support Programs.

Section 1 is a general overview of the program, and as such it should be read as background material for all sections which follow.

Section 2 contains descriptions of the data blocks, which are the principal means of data communication between the program's functional modules (a module is defined to be a group of subroutines which perform a specific function) and the NASTRAN Executive System. Two indexes for the data block descriptions, one sorted alphabetically on data block names and the other sorted alphabetically on the names of the modules from which the data blocks are output, are given in Sections 2.2.1 and 2.2.2 respectively. Section 2 also contains a) descriptions of tables, both core and noncore resident, maintained by the NASTRAN Executive System and b) descriptions of miscellaneous tables which are accessed by a class of modules. Alphabetical indexes for these tables are given at the beginning of Sections 2.4 and 2.5 respectively.

Sections 3 and 4 contain descriptions of the (utility or general purpose) subroutines and modules of NASTRAN respectively. The reader is directed to the alphabetical indexes, sorted on entry point names, in Sections 3.2 and 4.1.3 respectively for these sections. An index to the Module Functional Descriptions, sorted alphabetically on module names, is given in Section 4.1.2. The reader is urged to read the introductory material to Sections 3 and 4 before using these sections.

Section 5 treats computer and operating system dependent matters such as operating system control cards and generation of the absolute (executable) NASTRAN system.

Section 6 describes the means by which modifications and additions to NASTRAN are implemented.

Section 7 describes several auxiliary programs used to maintain or interface with NASTRAN.

The learning of any new system, whether it be an operating system or a large applications system like NASTRAN, is made more difficult than it ought to be because of the use by the designers of the system of new mnemonics, acronyms, phrases and "buzz" words. In order to aid the reader in

learning such commonly used NASTRAN terms, a single source reference, Section 7, the NASTRAN Dictionary, of the User's Manual is provided. The programmer is advised to secure a copy of at least this section of the User's Manual for his day-to-day reference.

TABLE OF CONTENTS

<u>Section</u>	<u>Page No.</u>
1. NASTRAN PROGRAMMING FUNDAMENTALS	
1.1 PROGRAM OVERVIEW	1.1-1
1.1.1 Objectives	1.1-1
1.1.2 Program Organization	1.1-3
1.2 NASTRAN EXECUTIVE SYSTEM	1.2-1
1.2.1 Introduction	1.2-1
1.2.2 Executive Operations During the Preface	1.2-4
1.2.3 Executive Operations During Problem Solution	1.2-9a
1.3 WORD SIZE AND COMPUTER HARDWARE CONSIDERATIONS	1.3-1
1.3.1 Introduction	1.3-1
1.3.2 Alphanumeric Data	1.3-2
1.3.3 Word Packing	1.3-2
1.4 SYSTEM BLOCK DATA SUBPROGRAM (SEMDBD)	1.4-1
1.5 THE OPEN CORE CONCEPT	1.5-1
1.5.1 Introduction	1.5-1
1.5.2 Definition of Open Core	1.5-1
1.5.3 Example of an Application of Open Core	1.5-1
1.6 NASTRAN INPUT/OUTPUT	1.6-1
1.6.1 Introduction	1.6-1
1.6.2 Use of the Operating System Input File	1.6-1
1.6.3 Use of the Operating System Output File	1.6-2
1.6.4 GINØ	1.6-3
1.6.5 SØF	1.6-7
1.7 NASTRAN MATRIX ROUTINES	1.7-1
1.7.1 Introduction	1.7-1
1.7.2 Matrix Packing and Unpacking	1.7-1
1.7.3 The Nested Vector Set Concept Used to Represent Components of Displacement	1.7-2
1.7.4 Processing of Matrices	1.7-5
1.8 GENERATION OF MATRICES	1.8-1
1.8.1 The EST, EDPT, and GPECT Data Blocks	1.8-1
1.8.2 Structural Elements	1.8-2

TABLE OF CONTENTS (Continued)

<u>Section</u>	<u>Page No.</u>
1.9 TERMINATION PHILOSOPHY AND DIAGNOSTIC MESSAGES	1.9-1
1.10 RESTARTS IN NASTRAN	1.10-1
2. DATA BLOCK AND TABLE DESCRIPTIONS	
2.1 INTRODUCTION	2.1-1
2.2 DATA BLOCK DESCRIPTIONS - GENERAL COMMENTS AND INDEXES	2.2-1
2.2.1 Index for Data Block Descriptions Sorted on Data Block Names	2.2-3
2.2.2 Index for Data Block Descriptions Sorted Alphabetically by Module ..	2.2-20
2.3 DATA BLOCK DESCRIPTIONS	2.3-1
2.3.1 Data Blocks Output From Module IFP1	2.3-1
2.3.2 Data Blocks Output From Module IFP	2.3-5
2.3.3 Data Blocks Output From Module GP1	2.3-41
2.3.4 Data Blocks Output From Module GP2	2.3-46
2.3.5 Data Blocks Output From Module PLTSET	2.3-47
2.3.6 Data Blocks Output From Module PLØT	2.3-50
2.3.7 Data Blocks Output From Module GP3	2.3-51
2.3.8 Data Blocks Output From Module TA1	2.3-56
2.3.9 Data Blocks Output From Module SMA1	2.3-74
2.3.10 Data Blocks Output From Module SMA2	2.3-76
2.3.11 Data Blocks Output From Module GPWG	2.3-77
2.3.12 Data Blocks Output From Module SMA3	2.3-78
2.3.13 Data Blocks Output From Module GP4	2.3-79
2.3.14 Data Blocks Output From Module GPSP	2.3-83
2.3.15 Data Blocks Output From Module MCE1	2.3-84
2.3.16 Data Blocks Output From Module MCE2	2.3-85
2.3.17 Data Blocks Output From Module SCE1	2.3-88
2.3.18 Data Blocks Output From Module SMP1	2.3-92
2.3.19 Data Blocks Output From Module RBMG1	2.3-96
2.3.20 Data Blocks Output From Module RBMG2	2.3-99
2.3.21 Data Blocks Output From Module RBMG3	2.3-101
2.3.22 Data Blocks Output From Module RBMG4	2.3-102

TABLE OF CONTENTS (Continued)

<u>Section</u>	<u>Page No.</u>
2.3.23 Data Blocks Output From Module SSG1	2.3-103
2.3.24 Data Blocks Output From Module SSG2	2.3-104
2.3.25 Data Blocks Output From Module SSG3	2.3-107
2.3.26 Data Blocks Output From Module SSG4	2.3-110
2.3.27 Data Blocks Output From Module SDR1	2.3-111
2.3.28 Data Blocks Output From Module SDR2	2.3-116
2.3.29 Data Blocks Output From Module DPD	2.3-145
2.3.30 Data Blocks Output From Module READ	2.3-157
2.3.31 Data Blocks Output From Module DSMG1	2.3-161
2.3.32 Data Blocks Output From Module SMP2	2.3-162
2.3.33 Data Blocks Output From Module DSMG2	2.3-163
2.3.34 Data Blocks Output From Module PLA1	2.3-165
2.3.35 Data Blocks Output From Module ADD	2.3-172
2.3.36 Data Blocks Output From Module PLA2	2.3-173
2.3.37 Data Blocks Output From Module PLA3	2.3-174
2.3.38 Data Blocks Output From Module PLA4	2.3-175
2.3.39 Data Blocks Output From Module CASE	2.3-176
2.3.40 Data Blocks Output From Module MTRXIN	2.3-177
2.3.41 Data Blocks Output From Module GKAD	2.3-179
2.3.42 Data Blocks Output From Module CEAD	2.3-183
2.3.43 Data Blocks Output From Module VDR	2.3-186
2.3.44 Data Blocks Output From Module FRRD	2.3-195
2.3.45 Data Blocks Output From Module SDR3	2.3-197
2.3.46 Data Blocks Output From Module XYTRAN	2.3-218
2.3.47 Data Blocks Output From Module RANDOM	2.3-222
2.3.48 Data Blocks Output From Module TRD	2.3-224
2.3.49 Data Blocks Output From Module GKAM	2.3-225
2.3.50 Data Blocks Output From Module DDR1	2.3-226
2.3.51 Element Stress Output Data Description	2.3-227
2.3.52 Element Force Output Data Description	2.3-233

TABLE OF CONTENTS (Continued)

<u>Section</u>	<u>Page No.</u>
2.3.53 Data Blocks Output From Module DDR2	2.3-237
2.3.54 Data Blocks Output From Module BMG	2.3-239
2.3.55 Data Blocks Output From Module PLTTRAN	2.3-239
2.3.56 Data Blocks Output From Module RMG	2.3-240
2.3.57 Data Blocks Output From Module TRLG	2.3-240
2.3.58 Data Blocks Output From Module TRHT	2.3-242
2.3.59 Data Blocks Output From Module SSGHT	2.3-242
2.3.60 Data Blocks Output From Module SDRHT	2.3-243
2.3.61 Data Blocks Output From Module GPCYC	2.3-244
2.3.62 Data Blocks Output From Module APD	2.3-245
2.3.63 Data Blocks Output From Module GI	2.3-253
2.3.64 Data Blocks Output From Module AMG	2.3-254
2.3.65 Data Blocks Output From Module AMP	2.3-256
2.3.66 Data Blocks Output From Module FA1	2.3-258
2.3.67 Data Blocks Output From Module FA2	2.3-260
2.3.68 Data Blocks Output From Module ØPTR1	2.3-262
2.3.69 Data Blocks Output From Module EMA	2.3-264
2.3.70 Data Blocks Output From Module EMG	2.3-265
2.3.71 Data Blocks Output From Module ASDMAP	2.3-269
2.3.72 Data Blocks Output From Module CØMB2	2.3-271
2.3.73 Data Blocks Output From Module RCØVR	2.3-272
2.3.74 Data Blocks Output From Module RCØVR3	2.3-274
2.3.75 Data Blocks Output From Module REDUCE	2.3-276
2.3.76 Data Blocks Output From Module SGEN	2.3-278
2.3.77 Data Blocks Output From Module PLTMRG	2.3-281
2.3.78 Data Blocks Output From Module MØDACC	2.3-284
2.3.79 Data Blocks Output From Module DDRMM	2.3-285
2.3.80 Data Blocks Output From Module ØPTR2	2.3-287
2.3.81 Data Blocks Output From Module INPUTT2	2.3-288
2.3.82 Data Blocks Output From Module CURV	2.3-289

TABLE OF CONTENTS (Continued)

<u>Section</u>	<u>Page No.</u>
2.3.83 Data Blocks Output From Module GPFDR	2.3-290
2.3.84 Data Blocks Output From Module CYCT1	2.3-292
2.3.85 Data Blocks Output From Module CYCT2	2.3-293
2.3.86 Data Blocks Output From Module FRLG	2.3-295
2.3.87 Data Blocks Output From Module LAMX	2.3-296
2.3.88 Data Blocks Output From Module IFT	2.3-297
2.3.89 Data Blocks Output From Module FRRD2	2.3-298
2.3.90 Data Blocks Output From Module ADR	2.3-299
2.3.91 Data Blocks Output From Module GUST	2.3-300
2.3.92 Data Blocks Output From Module EQMCK	2.3-301
2.4 EXECUTIVE TABLE DESCRIPTIONS	2.4-1
2.4.1 Executive Tables Which are Permanently Core Resident.....	2.4-2
2.4.2 Executive Tables Not Permanently Core Resident.....	2.4-16
2.5 MISCELLANEOUS TABLE DESCRIPTIONS	2.5-1
2.5.1 Miscellaneous Tables Which are Permanently Core Resident.....	2.5-2
2.5.2 Miscellaneous Tables Not Permanently Core Resident	2.5-6
2.6 SUBSTRUCTURE DATA ITEMS DESCRIPTIONS	2.6-1
2.6.1 Substructure Data "Items" Description	2.6-2
3. SUBROUTINE DESCRIPTIONS	
3.1 INTRODUCTION	3.1-1
3.2 ALPHABETICAL INDEX OF ENTRY POINTS FOR SUBROUTINE DESCRIPTIONS	3.2-1
3.3 EXECUTIVE SUBROUTINE DESCRIPTIONS	3.3-1
3.3.1 XSEM1 (Executive Sequence Monitor, Preface)	3.3-1
3.3.2 BTSTRP (Bootstrap Generator)	3.3-2
3.3.3 SEMINT (Sequence Monitor Initialization)	3.3-3
3.3.4 GNFIAT (Generate FIAT)	3.3-5
3.3.5 ENDSYS (End-of-Link)	3.3-6
3.3.6 SEARCH (Search, Load, and Execute Link)	3.3-8
3.3.7 XSEMi (Link i Main Program, i = 2,3,...)	3.3-9
3.3.8 XSEMXX (Sequence Monitor - Deck Generator)	3.3-11

TABLE OF CONTENTS (Continued)

<u>Section</u>	<u>Page No.</u>
3.3.9 GNFIST (Generate FIAT)	3.3-12
3.3.10 XEØT (End-of-Tape)	3.3-14
3.3.11 SSWITCH (Sense Switches)	3.3-15
3.3.12 CØNMSG (Console Message Writer)	3.3-16
3.3.13 TTLPGØ (Title Page Writer)	3.3-17
3.3.14 SEMTRN (Transliterator) (IBM 360-370 Only)	3.3-19
3.3.15 RETURN (Return)	3.3-20
3.3.16 NASCAR (Read the NASTRAN Card)	3.3-21
3.4 UTILITY SUBROUTINE DESCRIPTIONS	3.4-1
3.4.1 MAPFNS (Machine Word Functions)	3.4-1
3.4.10 NASTIØ (NASTRAN Input/Output Manager)	3.4-12
3.4.11 ØPEN (Initiate Activity on a File)	3.4-14
3.4.12 GINØ (General Input/Output Routine)	3.4-15
3.4.14 GØPEN (Short Form for Subroutine ØPEN with Header Record Processing)	3.4-22
3.4.15 FREAD (Short Form for Subroutine READ)	3.4-23
3.4.16 WRTTRL (Write Trailer)	3.4-24
3.4.17 FNAME (File Name)	3.4-25
3.4.18 CLSTAB (Close A GINØ File and Write a Nonzero Trailer)	3.4-26
3.4.19 XRCARD (Executive Free-Field Card Data Conversion Routine) .	3.4-27
3.4.20 RCARD (Fixed Field Card Data Conversion Routine)	3.4-32
3.4.21 TAPBIT (Tape Bit Test)	3.4-35
3.4.22 PEXIT (Problem Exit)	3.4-36
3.4.23 TMTØGØ (Time-To-Go)	3.4-37
3.4.24 PAGE (Page Heading)	3.4-38
3.4.25 MESSAGE (Message)	3.4-39
3.4.26 MSGWRT (Message Writer)	3.4-40
3.4.27 USRMSG (User Message Writer)	3.4-41
3.4.28 MATDUM (Matrix Dump (Print) Routine)	3.4-42
3.4.29 TABPRT (Table Printer)	3.4-43
3.4.30 PRELØC (Position Data Block to Requested Record)	3.4-44

TABLE OF CONTENTS (Continued)

<u>Section</u>	<u>Page No.</u>
3.4.31 SØRT (Sort a Table)	3.4-46
3.4.32 GMMATD (General Matrix Multiple and Transpose - Double Precision)	3.4-49
3.4.33 GMMATS (General Matrix Multiply and Transpose - Single Precision)	3.4-52
3.4.34 INVERD (Double Precision In Core Inverse Routine)	3.4-53
3.4.35 INVERS (Single Precision In Core Inverse Routine)	3.4-54
3.4.36 PREMAT (Material Property Utility)	3.4-55
3.4.37 PRETRD (Utility for Modules Which Use the CSTM Data Block - Double Precision Version)	3.4-64
3.4.38 PRETRS (Utility for Modules Which Use the CSTM Data Block - Single Precision Version)	3.4-66
3.4.39 PRETAB (Table Look-up)	3.4-67
3.4.40 AXIS (Draw an Axis on a Plot)	3.4-70
3.4.41 AXISi (Axis Routine for Plotter i)	3.4-72
3.4.42 SKPFRM (Skip a Variable Number of Frames)	3.4-73
3.4.43 SELCAM (To Initiate a New Plot)	3.4-74
3.4.44 IDPLØT (Generate an "ID" Plot)	3.4-75
3.4.45 INTGPX (Search a List of Integers)	3.4-76
3.4.46 INTLST (Interpret a List of Integers)	3.4-77
3.4.47 LINE (Draw a Line on a Plotter)	3.4-78
3.4.48 LINEi (Draw a Line on Plotter i)	3.4-79
3.4.49 PRINT (Print a Title on a Plotter)	3.4-81
3.4.50 RDMØDX (Read a File Containing XRCARD Translations)	3.4-83
3.4.51 SGINØ (GINØ for Unformatted Tapes)	3.4-85
3.4.52 STPLØT (To Initiate a New Plot or Terminate the Current Plot)	3.4-87
3.4.53 SYMBOL (Type a Symbol on a Plotter)	3.4-88
3.4.54 TIPE (Type a Line of Characters on a Plotter)	3.4-90
3.4.55 TYPEi (Type a Line of Characters on Plotter i).....	3.4-92
3.4.56 TYPFLT (Type a Floating Point Number on a Plotter)	3.4-94
3.4.57 TYPINT (Type an Integer Number on a Plotter)	3.4-96
3.4.58 WPLT1 (Write a Plotter Command for Plotter 1)	3.4-98

TABLE OF CONTENTS (Continued)

<u>Section</u>	<u>Page No.</u>
3.4.59 WPLT2 (Write a Plotter Command for Plotters 2 and 8)	3.4-100
3.4.60 WPLT3 (Write a Plotter Command for Plotter 3)	3.4-102
3.4.62 EJECT (Automatic Page Eject)	3.4-105
3.4.63 PLAMAT (Material Property Utility for Two-Dimensional Elements in Piecewise Linear Analysis)	3.4-106
3.4.64 WPLT4 (Write a Plotter Command for Plotters 4 thru 7)	3.4-108
3.4.65 WPLT9 (Write a Plotter Command for Plotter 9)	3.4-110
3.4.66 WPLT10 (Write a Plotter Command for the NASTRAN General Purpose Plotter)	3.4-111
3.4-67 PLTSET (Plotting Parameter Initialization).....	3.4-113
3.4.68 DRWCHR (To Draw a Line of Characters)	3.4-115
3.4.69 FNDPLT (Determine the Internal Plotter and Model Indices)	3.4-117
3.4.70 PHDMIA (DMI Punch Routine)	3.4-118
3.4.71 HEAD (Plot Heading)	3.4-120
3.4.72 DELSET (Dummy Element Setup)	3.4-121
3.4.73 HMAT (Setup and Compute Heat Transfer Material Coefficients)..	3.4-122
3.4.74 BISRCH (Binary Search)	3.4-123
3.4.75 FØRFIL (File Unit)	3.4-126
3.4.76 DADØTB (Double Precision Vector Dot Product)	3.4-127
3.4.77 DAXB (Double Precision Vector Cross Product)	3.4-128
3.4.78 SADØTB (Single Precision Vector Dot Product)	3.4-129
3.4.79 SAXB (Single Precision Vector Cross Product)	3.4-130
3.4.80 HBDY (Compute HBDY Element Coefficients)	3.4-131
3.4.81 DECØDE (Decodes "1" Bits in a 32-Bit Computer Word)	3.4-132
3.4.82 ECTLØC (Special Version of Entry Point LØCATE)	3.4-133
3.4.83 MRGE (Merges Two Sorted Lists to Form a New List).....	3.4-134
3.4.84 DMPFIL (Print a NASTRAN Scratch Data Block)	3.4-135
3.4.85 SSPLIN (Produce an Interpolation Matrix)	3.4-136
3.4.86 LSPLIN (Produce an Interpolation Matrix)	3.4-138
3.4.87 BISLØC (Binary Search Routine)	3.4-140
3.4.88 BISHEL (Merge Unique Entries Into an Array)	3.4-142

TABLE OF CONTENTS (Continued)

<u>Section</u>	<u>Page No.</u>
3.4.89 BUG (Produce Debugging Printout)	3.4-144
3.4.90 SKPREC (Skip Records on a File)	3.4-145
3.4.91 RE2AL (Real Number to Alphanumeric)	3.4-146
3.4.92 CPYSTR (Copies a String Formatted Record)	3.4-148
3.4.93 CPYFIL (Copy File)	3.4-149
3.4.94 SETFND (ID Lookup Utility)	3.4-150
3.4.95 SAMB (Vector Subtract)	3.4-152
3.4.96 SAPB (Vector Add)	3.4-153
3.4.97 GMMATC (General Matrix Multiply and Transpose - Complex Single Precision)	3.4-154
3.5 MATRIX SUBROUTINE DESCRIPTIONS	3.5-1
3.5.1 PAKUNPK (PAck/UNPack)	3.5-1
3.5.5 CALCV (Compute a Partitioning Vector)	3.5-8
3.5.6 PARTN - MERGE (Partition a Matrix - Merge Matrices Together).....	3.5-9
3.5.7 SSG2A (Driver for PARTN).....	3.5-12
3.5.8 SDR1B (Driver for MERGE)	3.5-13
3.5.9 UPART (Symmetric Partition Driver)	3.5-14
3.5.10 ADD (Driver for SADD)	3.5-15
3.5.11 SSG2C (Driver for ADD)	3.5-16
3.5.12 MPYAD (Matrix Multiplication Routine)	3.5-18
3.5.13 SSG2B (Driver for MPYAD).....	3.5-31
3.5.14 SDCOMP (Symmetric Decomposition).....	3.5.32
3.5.15 DECOMP (Unsymmetric Matrix Decomposition)	3.5-56
3.5.16 CDCOMP (Complex Matrix Decomposition).....	3.5-74
3.5.17 FBS (Forward - Backward Substitution)	3.5-76
3.5.18 SSG3A (Driver for FBS)	3.5-78
3.5.19 GFBS (General Forward - Backward Substitution).....	3.5-79
3.5.20 SOLVER (Simultaneous Equation Solution Routine)	3.5-81
3.5.21 DMPY (Multiply a Diagonal Matrix by an Arbitrary Matrix)	3.5-83
3.5.22 ELIM (Perform a Matrix Reduction)	3.5-85

TABLE OF CONTENTS (Continued)

<u>Section</u>	<u>Page No.</u>
3.5.23 FACTØR (Decompose a Matrix Into Triangular Factors)	3.5-86
3.5.24 TRANP1 (Driver for TRNSP)	3.5-87
3.5.25 TRNSP (Matrix Transpose)	3.5-88
3.5.26 SADD (Matrix Addition Routine)	3.5-90
3.5.30 MAKMCB (Make a Matrix Control Block)	3.5-93
3.5.31 ATEIG (Find the Eigenvector of an Upper Hessenberg Matrix	3.5-94
3.5.32 HSBG (Reduce a Matrix to Upper Hessenberg Form)	3.5-95
3.5.33 INCØRE (Incore Complex Solve)	3.5-96
3.5.34 EGNVCT (Calculate Eigenvectors)	3.5-97
3.5.35 MPY3 (Triple Matrix Multiply)	3.5-98
3.6 SUBSTRUCTURE OPERATING FILE (SØF) UTILITIES	3.6-1
3.6.1 Tables Associated with the SØF	3.6-1
3.6.2 Organization of the SØF	3.6-8
3.6.3 In-core Handling of the SØF Data and Tables	3.6-10
3.6.4 Index for SØF Utility Functions and Subroutines	3.6-12
3.6.5 CHKØPN	3.6-13
3.6.6 CRSUB (Create Substructures)	3.6-14
3.6.7 DELETE	3.6-15
3.6.8 DSTRØY	3.6-16
3.6.9 EDIT	3.6-18
3.6.10 EQSØF (Equate SØF)	3.6-19
3.6.11 ERRMKN	3.6-20
3.6.12 FDIT (Fetch DIT)	3.6-21
3.6.13 FDNAME	3.6-22
3.6.14 FDSUB	3.6-23
3.6.15 FMDI	3.6-24
3.6.16 FNDNXL	3.6-25
3.6.17 FNXT	3.6-26
3.6.18 GETBLK	3.6-27
3.6.19 ITCØDE	3.6-28

TABLE OF CONTENTS (Continued)

<u>Section</u>	<u>Page No.</u>
3.6.20 MTRXI	3.6-29
3.6.21 MTRXØ	3.6-30
3.6.22 RETBLK	3.6-31
3.6.23 SETEQ	3.6-32
3.6.24 SETLVL	3.6-34
3.6.25 SFETCH	3.6-36
3.6.26 SJUMP	3.6-38
3.6.27 SØFCLS	3.6-39
3.6.28 SØFINT	3.6-40
3.6.29 SØFIØ	3.6-42
3.6.30 SØFØPN	3.6-43
3.6.31 SØFSIZ	3.6-44
3.6.32 SUREAD	3.6-45
3.6.33 SUWRT	3.6-46
3.6.34 Common Blocks Used by the SØF Utility Subroutines	3.6-47
3.6.35 *MSG (Substructure Messages)	3.6-51
3.6.36 SØFTRL	3.6-52
3.6.37 ITTYPE	3.6-53
3.6.38 FNDLVL	3.6-54
 4. MODULE FUNCTIONAL DESCRIPTIONS	
4.1 GENERAL COMMENTS AND INDEXES	4.1-1
4.1.1 Use of Module Functional Descriptions	4.1-2
4.1.2 Alphabetical Index of Module Functional Descriptions.....	4.1-2
4.1.3 Alphabetical Index of Entry Points in Module Functional Descriptions	4.1-8
4.2 EXECUTIVE PREFACE MODULE XSCA (EXECUTIVE CONTROL SECTION ANALYSIS)....	4.2-1
4.3 EXECUTIVE PREFACE MODULE IFP1 (INPUT FILE PROCESSOR, PART 1).....	4.3-1
4.4 EXECUTIVE PREFACE MODULE XSØRT (EXECUTIVE BULK DATA CARD SORT)	4.4-1
4.5 EXECUTIVE PREFACE MODULE IFP (INPUT FILE PROCESSOR).....	4.5-1
4.6 EXECUTIVE PREFACE MODULE IFP3 (INPUT FILE PROCESSOR 3)	4.6-1
4.7 EXECUTIVE PREFACE MODULE XGPI (EXECUTIVE GENERAL PROBLEM INITIALIZATION).....	4.7-1
4.8 EXECUTIVE PREFACE MODULE UMFEDIT (USER MASTER FILE EDITOR)	4.8-1
4.9 EXECUTIVE MODULE XSFA (EXECUTIVE SEGMENT FILE ALLOCATOR).....	4.9-1

TABLE OF CONTENTS (Continued)

<u>Section</u>	<u>Page No.</u>
4.10 EXECUTIVE DMAP MODULE CHKPNT (CHECKPOINT)	4.10.1
4.11 EXECUTIVE DMAP INSTRUCTION REPT (REPEAT A GROUP OF DMAP INSTRUCTIONS)	4.11-1
4.12 EXECUTIVE DMAP INSTRUCTION JUMP (UNCONDITIONAL DMAP TRANSFER)	4.12-1
4.13 EXECUTIVE DMAP INSTRUCTION CØND (CONDITIONAL TRANSFER).....	4.13-1
4.14 EXECUTIVE DMAP INSTRUCTION EXIT (TERMINATE DMAP PROGRAM)	4.14-1
4.15 EXECUTIVE DMAP MODULE SAVE (SAVE VARIABLE PARAMETER VALUES)	4.15-1
4.16 EXECUTIVE DMAP MODULE PURGE (EXPLICIT DATA BLOCK PURGE)	4.16-1
4.17 EXECUTIVE DMAP MODULE EQUIV (DATA BLOCK NAME EQUIVALENCE)	4.17-1
4.18 EXECUTIVE DMAP INSTRUCTION END (END OF DMAP PROGRAM)	4.18-1
4.19 EXECUTIVE DMAP MODULE PARAM (PARAMETER PROCESSOR)	4.19-1
4.20 EXECUTIVE DMAP MODULE SETVAL (SET VALUES)	4.20-1
4.21 FUNCTIONAL MODULE GP1 (GEOMETRY PROCESSOR - PHASE 1)	4.21-1
4.22 FUNCTIONAL MODULE GP2 (GEOMETRY PROCESSOR - PHASE 2)	4.22-1
4.23 FUNCTIONAL MODULE PLTSET (PLOT SET DEFINITION PROCESSOR)	4.23-1
4.24 FUNCTIONAL MODULE PLØT (STRUCTURAL PLOTTER)	4.24-1
4.25 FUNCTIONAL MODULE GP3 (GEOMETRY PROCESSOR - PHASE 3)	4.25-1
4.26 FUNCTIONAL MODULE TA1 (TABLE ASSEMBLER)	4.26-1
4.27 FUNCTIONAL MODULE SMA1 (STRUCTURAL MATRIX ASSEMBLER - PHASE 1)	4.27-1
4.28 FUNCTIONAL MODULE SMA2 (STRUCTURAL MATRIX ASSEMBLER - PHASE 2)	4.28-1
4.29 FUNCTIONAL MODULE GPWG (GRID POINT WEIGHT GENERATOR)	4.29-1
4.30 FUNCTIONAL MODULE SMA3 (STRUCTURAL MATRIX ASSEMBLER - PHASE 3)	4.30-1
4.31 FUNCTIONAL MODULE GP4 (GEOMETRY PROCESSOR - PHASE 4)	4.31-1
4.32 FUNCTIONAL MODULE GPSPC (GRID POINT SINGULARITY PROCESSOR)	4.32-1
4.33 FUNCTIONAL MODULE MCE1 (MULTIPOINT CONSTRAINT ELIMINATOR - PHASE 1) ..	4.33-1
4.34 FUNCTIONAL MODULE MCE2 (MULTIPOINT CONSTRAINT ELIMINATOR - PHASE 2) ..	4.34-1
4.35 FUNCTIONAL MODULE SCE1 (SINGLE-POINT CONSTRAINT ELIMINATOR)	4.35-1
4.36 FUNCTIONAL MODULE SMP1 (STRUCTURAL MATRIX PARTITIONER - PHASE 1)	4.36-1
4.37 FUNCTIONAL MODULE RBMG1 (RIGID BODY MATRIX GENERATOR - PHASE 1)	4.37-1
4.38 FUNCTIONAL MODULE RBMG2 (RIGID BODY MATRIX GENERATOR - PHASE 2)	4.38-1
4.39 FUNCTIONAL MODULE RBMG3 (RIGID BODY MATRIX GENERATOR - PHASE 3)	4.39-1

TABLE OF CONTENTS (Continued)

<u>Section</u>	<u>Page No.</u>
4.40 FUNCTIONAL MODULE RBMG4 (RIGID BODY MATRIX GENERATOR - PHASE 4)	4.40-1
4.41 FUNCTIONAL MODULE SSG1 (STATIC SOLUTION GENERATOR - PHASE 1)	4.41-1
4.42 FUNCTIONAL MODULE SSG2 (STATIC SOLUTION GENERATOR - PHASE 2)	4.42-1
4.43 FUNCTIONAL MODULE SSG3 (STATIC SOLUTION GENERATOR - PHASE 3)	4.43-1
4.44 FUNCTIONAL MODULE SSG4 (STATIC SOLUTION GENERATOR - PHASE 4)	4.44-1
4.45 FUNCTIONAL MODULE SDR2 (STRESS DATA RECOVERY - PHASE 1)	4.45-1
4.46 FUNCTIONAL MODULE SDR2 (STRESS DATA RECOVERY - PHASE 2)	4.46-1
4.47 FUNCTIONAL MODULE DPD (DYNAMICS POOL DISTRIBUTOR)	4.47-1
4.48 FUNCTIONAL MODULE READ (REAL EIGENVALUE ANALYSIS - DISPLACEMENT)	4.48-1
4.49 FUNCTIONAL MODULE DSMG1 (DIFFERENTIAL STIFFNESS MATRIX GENERATOR - PHASE 1)	4.49-1
4.50 FUNCTIONAL MODULE SMP2 (STRUCTURAL MATRIX PARTITIONER - PHASE 2)	4.50-1
4.51 FUNCTIONAL MODULE DSMG2 (DIFFERENTIAL STIFFNESS MATRIX GENERATOR - PHASE 2)	4.51-1
4.52 FUNCTIONAL MODULE PLA1 (PIECEWISE LINEAR ANALYSIS - PHASE 1)	4.52-1
4.53 FUNCTIONAL MODULE PLA2 (PIECEWISE LINEAR ANALYSIS - PHASE 2)	4.53-1
4.54 FUNCTIONAL MODULE PLA3 (PIECEWISE LINEAR ANALYSIS - PHASE 3)	4.54-1
4.55 FUNCTIONAL MODULE PLA4 (PIECEWISE LINEAR ANALYSIS - PHASE 4)	4.55-1
4.56 FUNCTIONAL MODULE CASE (SIMPLIFY CASE CONTROL)	4.56-1
4.57 FUNCTIONAL MODULE MTRXIN (MATRIX INPUT)	4.57-1
4.58 FUNCTIONAL MODULE GKAD (GENERAL K ASSEMBLER DIRECT)	4.58-1
4.59 FUNCTIONAL MODULE CEAD (COMPLEX EIGENVALUE ANALYSIS - DISPLACEMENT) ..	4.59-1
4.60 FUNCTIONAL MODULE VDR (VECTOR DATA RECOVERY)	4.60-1
4.61 FUNCTIONAL MODULE FRRD (FREQUENCY RESPONSE - DISPLACEMENT APPROACH) ..	4.61-1
4.62 FUNCTIONAL MODULE SDR3 (STRESS DATA RECOVERY - PHASE 3 - SORT1 to SORT2 PROCESSOR)	4.62-1
4.63 FUNCTIONAL MODULE XYTRAN (XY - OUTPUT DATA TRANSLATOR)	4.63-1
4.64 FUNCTIONAL MODULE RANDØM (RANDOM ANALYSIS MODULE)	4.64-1
4.65 FUNCTIONAL MODULE TRD (TRANSIENT ANALYSIS - DISPLACEMENT)	4.65-1
4.66 FUNCTIONAL MODULE GKAM (GENERAL K ASSEMBLER MODAL)	4.66-1
4.67 FUNCTIONAL MODULE DDR1 (DYNAMIC DATA RECOVERY - PART 1)	4.67-1
4.68 FUNCTIONAL MODULE DDR2 (DYNAMIC DATA RECOVERY - PART 2)	4.68-1

TABLE OF CONTENTS (Continued)

<u>Section</u>	<u>Page No.</u>
4.69 OUTPUT MODULE XYPLØT (X-Y DATA PLOTTER)	4.69-1
4.70 OUTPUT MODULE ØFP (OUTPUT FILE PROCESSOR)	4.70-1
4.71 OUTPUT MODULE MATPRN (GENERAL MATRIX PRINTER)	4.71-1
4.72 OUTPUT MODULE MATGPR (DISPLACEMENT METHOD MATRIX PRINTER)	4.72-1
4.73 OUTPUT MODULE MATPRT (MATRIX PRINTER)	4.73-1
4.74 OUTPUT MODULE SEEMAT (PICTORIAL MATRIX PRINTER)	4.74-1
4.75 OUTPUT MODULE TABPT (TABLE PRINTER)	4.75-1
4.76 OUTPUT MODULE PRMSG (MESSAGE WRITER)	4.76-1
4.77 OUTPUT MODULE PRTPARM (PARAMETER AND DMAP MESSAGE PRINTER)	4.77-1
4.78 MATRIX MODULE ADD (ADD TWO MATRICES)	4.78-1
4.79 MATRIX MODULE MPYAD (MULTIPLY ADD)	4.79-1
4.80 MATRIX MODULE SØLVE (SOLVES THE MATRIX EQUATION $[A][X] = [B]$)	4.80-1
4.81 MATRIX MODULE DECØMP (MATRIX DECOMPOSITION)	4.81-1
4.82 MATRIX MODULE FBS (FORWARD - BACK SUBSTITUTION)	4.82-1
4.83 MATRIX MODULE PARTN (PARTITION A MATRIX)	4.83-1
4.84 MATRIX MODULE MERGE (MERGE MATRICES TOGETHER)	4.84-1
4.85 MATRIX MODULE TRNSP (TRANSPOSE A MATRIX)	4.85-1
4.86 MATRIX MODULE SMPYAD (STRING MULTIPLY ADD)	4.86-1
4.89 EXECUTIVE PREFACE MODULE IFP4 (INPUT FILE PROCESSOR - PHASE 4)	4.89-1
4.90 FUNCTIONAL MODULE BMG (BOUNDARY MATRIX GENERATOR FOR HYDROELASTIC PROBLEMS)	4.90-1
4.91 EXECUTIVE PREFACE MODULE IFP5 (INPUT FILE PROCESSOR - PHASE 5)	4.91-1
4.92 FUNCTIONAL MODULE PLTTRAN	4.92-1
4.93 MATRIX MODULE UPARTN (PARTITIONS A MATRIX BASED ON USET).....	4.93-1
4.94 MATRIX MODULE UMERGE (MERGES TWO MATRICES BASED ON USET)	4.94-1
4.95 MATRIX MODULE VEC (CREATES PARTITIONING VECTOR BASED ON USET).....	4.95-1
4.96 MATRIX MODULE ADD5 (ADD MATRICES)	4.96-1
4.97 FUNCTIONAL MODULE INPUT (INPUT GENERATOR)	4.97-1
4.98 FUNCTIONAL MODULE INPUTT1	4.98-1
4.99 FUNCTIONAL MODULE INPUTT2	4.99-1
4.100 FUNCTIONAL MODULE ØUTPUT1	4.100-1

TABLE OF CONTENTS (Continued)

<u>Section</u>	<u>Page No.</u>
4.101 FUNCTIONAL MODULE ØUTPUT2	4.101-1
4.102 OUTPUT MODULE ØUTPUT3	4.102-1
4.103 OUTPUT MODULE TABPRT (FORMATTED TABLE PRINTER)	4.103-1
4.104 FUNCTIONAL MODULE RMG (RADIATION MATRIX GENERATOR)	4.104-1
4.105 FUNCTIONAL MODULE SSGHT (STATIC SOLUTION GENERATOR - HEAT TRANSFER) .	4.105-1
4.106 FUNCTIONAL MODULE TRLG (TRANSIENT LOAD GENERATOR)	4.106-1
4.107 FUNCTIONAL MODULE TRHT (TRANSIENT RESPONSE, HEAT TRANSFER)	4.107-1
4.108 FUNCTIONAL MODULE SDRHT (STRESS DATA RECOVERY, HEAT TRANSFER)	4.108-1
4.109 FUNCTIONAL MODULE GPCYC (GEOMETRY PROCESSOR FOR CYCLIC PROBLEMS)	4.109-1
4.110 FUNCTIONAL MODULE CYCT1 (TRANSFORMATION TO CYCLIC COMPONENTS - PHASE 1)	4.110-1
4.111 FUNCTIONAL MODULE CYCT2 (TRANSFORMATION TO CYCLIC COMPONENTS - PHASE 2)	4.111-1
4.112 FUNCTIONAL MODULE APD (AERODYNAMIC POOL DISTRIBUTOR)	4.112-1
4.113 FUNCTIONAL MODULE GI (GEOMETRY INTERPOLATOR)	4.113-1
4.114 FUNCTIONAL MODULE AMG (AERODYNAMIC MATRIX GENERATOR)	4.114-1
4.115 FUNCTIONAL MODULE AMP (AERODYNAMIC MATRIX PROCESSOR)	4.115-1
4.116 FUNCTIONAL MODULE FA1 (FLUTTER ANALYSIS - PHASE 1)	4.116-1
4.117 FUNCTIONAL MODULE FA2 (FLUTTER ANALYSIS - PHASE 2)	4.117-1
4.118 EXECUTIVE DMAP MODULE PARAML (PARAMETERS FROM A LIST)	4.118-1
4.119 EXECUTIVE DMAP MODULE PARAMR (FLOATING POINT PARAMETER PROCESSOR)...	4.119-1
4.120 FUNCTIONAL MODULE ØPTPR1 (FULLY STRESSED DESIGN - PHASE 1).....	4.120-1
4.121 FUNCTIONAL MODULE DSCHK (DIFFERENTIAL STIFFNESS CHECK)	4.121-1
4.122 FUNCTIONAL MODULE TABPCH (TABLE PUNCH)	4.122-1
4.123 FUNCTIONAL MODULE EMA (ELEMENT MATRIX ASSEMBLER)	4.123-1
4.124 FUNCTIONAL MODULE EMG (ELEMENT MATRIX GENERATOR)	4.124-1
4.126 FUNCTIONAL MODULE MØDACC (VECTOR SELECTION MODULE)	4.126-1
4.127 EXECUTIVE MODULE ASDMAP (ASSEMBLE SUBSTRUCTURE DMAP)	4.127-1
4.128 FUNCTIONAL MODULE CØMB1 (SUBSTRUCTURE COMBINATION, STEP 1).....	4.128-1
4.129 FUNCTIONAL MODULE CØMB2 (SUBSTRUCTURE COMBINATION, STEP 2).....	4.129-1
4.130 FUNCTIONAL MODULE EXIØ (EXTERNAL INPUT/OUTPUT FOR THE SØF).....	4.130-1
4.131 FUNCTIONAL MODULE RCØVR (RECOVER PHASE 2 SUBSTRUCTURE RESULTS).....	4.131-1
4.132 FUNCTIONAL MODULE RCØVR3 (RECOVER SUBSTRUCTURE RESULTS FOR PHASE 3)	4.132-1

TABLE OF CONTENTS (Continued)

<u>Section</u>	<u>Page No.</u>
4.133 FUNCTIONAL MODULE REDUCE (REDUCTION OF SUBSTRUCTURE DEGREES OF FREEDOM)	4.133-1
4.134 FUNCTIONAL MODULE SGEN (SUBSTRUCTURE TABLE GENERATOR)	4.134-1
4.135 FUNCTIONAL MODULE SØFI (SØF INTO GINØ MATRIX COPIER)	4.135-1
4.136 FUNCTIONAL MODULE SØFØ (SØF OUT FROM GINØ MATRIX COPIER)	4.136-1
4.137 FUNCTIONAL MODULE SØFUT (SØF UTILITY MODULE)	4.137-1
4.138 FUNCTIONAL MODULE SUBPH1 (SUBSTRUCTURE, PHASE 1)	4.138-1
4.139 FUNCTIONAL MODULE PLTMRG (SUBSTRUCTURE PLØTSET DATA MERGE)	4.139-1
4.140 UTILITY MODULE TIMETEST	4.140-1
4.141 FUNCTIONAL MODULE DDRMM (DYNAMIC DATA RECOVERY - MATRIX METHOD)	4.141-1
4.142 FUNCTIONAL MODULE OPTPR2 (FULLY STRESSED DESIGN - PHASE 2)	4.142-1
4.143 FUNCTIONAL MODULE DIAGØNAL (MATRIX DIAGONAL EXTRACTOR)	4.143-1
4.144 FUNCTIONAL MODULE SCALAR (MATRIX ELEMENT EXTRACTOR)	4.144-1
4.145 FUNCTIONAL MODULES MODULES CURV	4.145-1
4.146 FUNCTIONAL MODULE GPFDR (GRID POINT FORCE DATA RECOVERY)	4.146-1
4.147 FUNCTIONAL MODULE SWITCH	4.147-1
4.148 FUNCTIONAL MODULE COPY	4.148-1
4.149 FUNCTIONAL MODULE FRLG (FREQUENCY RESPONSE LOAD GENERATION)	4.149-1
4.150 FUNCTIONAL MODULE LAMX (EDIT LAMA DATA BLOCK)	4.150-1
4.151 FUNCTIONAL MODULE FRRD2 (FREQUENCY RESPONSE - WITH AEROELASTIC)	4.151-1
4.152 FUNCTIONAL MODULE ADR (AERODYNAMIC DATA RECOVERY)	4.152-1
4.153 FUNCTIONAL MODULE GUST (COMPUTE AERODYNAMIC LOADS DUE TO GUSTS)	4.153-1
4.154 FUNCTIONAL MODULE IFT (INVERSE FØURIER TRANSFORM)	4.154-1
4.155 MATRIX MODULE SDCMPS (MATRIX DECOMPOSITION - SYMMETRIC WITH CONTROLS AND DIAGNOSTICS)	4.155-1
4.156 FUNCTIONAL MODULE EQMCK	4.156-1
4.157 FUNCTIONAL MODULE MPY3 (TRIPLE MATRIX MULTIPLY)	4.157-1
4.158 FUNCTIONAL MODULE MRD1 (MODAL SYNTHESIS REDUCTION CALCULATION INITIALIZATION)	4.158-1
4.159 FUNCTIONAL MODULE MRD2 (MODAL SYNTHESIS REAL REDUCTION CALCULATIONS)	4.159-1
4.160 FUNCTIONAL MODULE CMRD2 (MODAL SYNTHESIS COMPLEX REDUCTION CALCULATIONS)	4.160-1
4.161 FUNCTIONAL MODULE EMA1 (ELEMENT MATRIX ASSEMBLER)	4.161-1

TABLE OF CONTENTS (Continued)

<u>Section</u>		<u>Page No.</u>
5.	NASTRAN - OPERATING SYSTEM INTERFACES	
5.1	INTRODUCTION	5.1-1
5.3	NASTRAN ON THE IBM SYSTEM 360/370	5.3-1
5.3.1	Introduction	5.3-1
5.3.2	Input/Output	5.3-1
5.3.3	Link Switching	5.3-4
5.3.4	Overlay Considerations and Implementation of Open Core ...	5.3-4
5.3.5	PARM Options	5.3-6
5.3.6	Execution Deck Setup	5.3-7
5.3.7	Physical Items and Generation of the NASTRAN System	5.3-18
5.3.8	Machine Dependent Routines	5.3-34
5.3.9	GINØ (Generalized Input/Output Processor for NASTRAN)	5.3-44
5.3.10	Special Error Codes from NASTRAN on the System 360	5.3-48
5.4	NASTRAN ON THE UNIVAC 1108/1110 (EXEC 8)	5.4-1
5.4.1	Introduction	5.4-1
5.4.2	Input/Output	5.4-1
5.4.3	Link Switching	5.4-4
5.4.4	Overlay Considerations and Implementation of Open Core ...	5.4-4
5.4.5	Execution Deck Setup	5.4-9
5.4.6	Description of NASTRAN Physical Items and Generation of the NASTRAN Executable System	5.4-10
5.4.7	Machine Dependent Routines	5.4-18
5.4.8	GINØ (S-Generalized Input/Output Processor for NASTRAN)...	5.4-22
5.4.9	The LINK99 Element	5.4-23
5.4.10	UNIVAC Overlay Diagrams	5.4-23
5.5	NASTRAN ON THE CDC 6000/CYBER (NØS and NØS/BE)	5.5-1
5.5.1	Introduction	5.5-1
5.5.2	Input/Output	5.5-1
5.5.3	NASTRAN on the Segmentation Loader	5.5-2
5.5.4	Physical Deliverables	5.5-13
5.5.5	Example Control Card Setups	5.5-16
5.5.6	Machine Dependent Routines	5.5-26

TABLE OF CONTENTS (Continued)

<u>Section</u>	<u>Page No.</u>
6. MODIFICATION AND ADDITIONS TO NASTRAN	
6.1 INTRODUCTION	6.1-1
6.2 FORTRAN IV LANGUAGE RESTRICTIONS	6.2-1
6.3 THE EXECUTIVE CONTROL DECK	6.3-1
6.4 THE CASE CONTROL DECK	6.4-1
6.5 THE BULK DATA DECK	6.5-1
6.6 RIGID FORMATS	6.6-1
6.7 FUNCTIONAL MODULES	6.7-1
6.8 ADDING A STRUCTURAL ELEMENT	6.8-1
6.8.1 Introduction to the Problem	6.8-1
6.8.2 General Guidelines	6.8-16
6.8.3 Specific Checklists	6.8-27
6.8.4 Updating the NASTRAN Manuals	6.8-53
6.8.5 Dummy User Elements (DUM1 through DUM9)	6.8-58
6.9 PRINTED OUTPUT	6.9-1
6.10 PLOTTER OUTPUT	6.10-1
6.10.1 Changes to the Plotter Software	6.10-1
6.10.2 Changes to the PLØT Module, the Structural Plotter	6.10-4
6.10.3 Changes to the XYPLØT Module, the XY Plotter	6.10-4
6.10.4 Changes to the SEEMAT Module, the Matrix Plotter	6.10-5
6.10.5 Use of the NASTRAN Plotter Software in a New Module	6.10-6
6.10.6 NASTRAN General Purpose Plotter	6.10-14
6.11 ADDITION OF A NEW LINK	6.11-1
6.11.1 Modules to Include	6.11-1
6.11.2 Addition of New Modules	6.11-1
6.11.3 Generation of a New Link Specification Table and a New Link Driver	6.11-2
6.11.4 Subsys the New Link	6.11-4
6.11.5 Increasing the Link Limit	6.11-4
6.11.6 Adding a New Link to the CDC Version of NASTRAN	6.11-4
6.11.7 Adding a New Link to the IBM Version of NASTRAN	6.11-4
6.11.8 Adding a New Link to the UNIVAC Version of NASTRAN	6.11-5

TABLE OF CONTENTS (Continued)

<u>Section</u>		<u>Page No.</u>
6.12	WRITING A NEW MODULE	6.12-1
6.12.1	Summary of NASTRAN Coding Conventions and Terminology	6.12-1
6.12.2	Module Design	6.12-3
6.12.3	Implementing the New Module	6.12-7
6.12.4	Coding a Module Subroutine	6.12-8
6.12.5	Sample Module Coding	6.12-2
6.13	THE NASTPLT MODEL POST PROCESSOR	6.13-1
6.13.1	Introduction	6.13-1
6.13.2	The NASTPLT Post-Processor	6.13-1
6.13.3	NASTPLT Input	6.13-2
6.13.4	Use of NASTPLT with NASTRAN	6.13-3
7.	RIGID FORMAT RESTART TABLES	
7.1	RESTART TABLES	7.1-1
7.1.1	Card Name Restarts	7.1-1
7.1.2	Rigid Format Restarts	7.1-2
7.1.3	File Name Restarts	7.1-2
7.2	RESTART TABLES FOR STATIC ANALYSIS	7.2-1
7.2.1	Bit Positions for Card Name Restart Table	7.2-1
7.2.2	Bit Positions for File Name Restart Table	7.2-3
7.2.3	Card Name Restart Table	7.2-4
7.2.4	Rigid Format Change Restart Table	7.2-9
7.2.5	File Name Restart Table	7.2-11
7.3	RESTART TABLES FOR STATIC ANALYSIS WITH INERTIA RELIEF	7.3-1
7.3.1	Bit Positions for Card Name Restart Table	7.3-1
7.3.2	Bit Positions for File Name Restart Table	7.3-3
7.3.3	Card Name Restart Table	7.3-4
7.3.4	Rigid Format Change Restart Table	7.3-8
7.3.5	File Name Restart Table	7.3-10
7.4	RESTART TABLES FOR NORMAL MODES ANALYSIS	7.4-1

TABLE OF CONTENTS (Continued)

<u>Section</u>		<u>Page No.</u>
	7.4.1 Bit Positions for Card Name Restart Table	7.4-1
	7.4.2 Bit Positions for File Name Restart Table	7.4-3
	7.4.3 Card Name Restart Table	7.4-4
	7.4.4 Rigid Format Change Restart Table	7.4-8
	7.4.5 File Name Restart Table	7.4-10
7.5	RESTART TABLES FOR STATIC ANALYSIS WITH DIFFERENTIAL STIFFNESS	7.5-1
	7.5.1 Bit Positions for Card Name Restart Table	7.5-1
	7.5.2 Bit Positions for File Name Restart Table	7.5-3
	7.5.3 Card Name Restart Table	7.5-4
	7.5.4 Rigid Format Change Restart Table	7.5-9
	7.5.5 File Name Restart Table	7.5-11
7.6	RESTART TABLES FOR BUCKLING ANALYSIS	7.6-1
	7.6.1 Bit Positions for Card Name Restart Table	7.6-1
	7.6.2 Bit Positions for File Name Restart Table	7.6-3
	7.6.3 Card Name Restart Table	7.6-4
	7.6.4 Rigid Format Change Restart Table	7.6-9
	7.6.5 File Name Restart Table	7.6-11
7.7	RESTART TABLES FOR PIECEWISE LINEAR ANALYSIS	7.7-1
	7.7.1 Bit Positions for Card Name Restart Table	7.7-1
	7.7.2 Bit Positions for File Name Restart Table	7.7-3
	7.7.3 Card Name Restart Table	7.7-4
	7.7.4 Rigid Format Change Restart Table	7.7-9
	7.7.5 File Name Restart Table	7.7-11
7.8	RESTART TABLES FOR DIRECT COMPLEX EIGENVALUE ANALYSIS	7.8-1
	7.8.1 Bit Positions for Card Name Restart Table	7.8-1
	7.8.2 Bit Positions for File Name Restart Table	7.8-3
	7.8.3 Card Name Restart Table	7.8-4
	7.8.4 Rigid Format Change Restart Table	7.8-9
	7.8.5 File Name Restart Table	7.8-11
7.9	RESTART TABLES FOR DIRECT FREQUENCY AND RANDOM RESPONSE	7.9-1
	7.9.1 Bit Positions for Card Name Restart Table	7.9-1

TABLE OF CONTENTS (Continued)

<u>Section</u>		<u>Page No.</u>
7.9.2	Bit Positions for File Name Restart Table	7.9.3
7.9.3	Card Name Restart Table	7.9-4
7.9.4	Rigid Format Change Restart Table	7.9-9
7.9.5	File Name Restart Table	7.9-11
7.10	RESTART TABLES FOR DIRECT TRANSIENT RESPONSE	7.10-1
7.10.1	Bit Positions for Card Name Restart Table	7.10-1
7.10.2	Bit Positions for File Name Restart Table	7.10-3
7.10.3	Card Name Restart Table	7.10-4
7.10.4	Rigid Format Change Restart Table	7.10-9
7.10.5	File Name Restart Table	7.10-11
7.11	RESTART TABLES FOR MODAL COMPLEX EIGENVALUE ANALYSIS	7.11-1
7.11.1	Bit Positions for Card Name Restart Table	7.11-1
7.11.2	Bit Positions for File Name Restart Table	7.11-3
7.11.3	Card Name Restart Table	7.11-4
7.11.4	Rigid Format Change Restart Table	7.11-8
7.11.5	File Name Restart Table	7.11-10
7.12	RESTART TABLES FOR MODAL FREQUENCY AND RANDOM RESPONSE	7.12-1
7.12.1	Bit Positions for Card Name Restart Table	7.12-1
7.12.2	Bit Positions for File Name Restart Table	7.12-3
7.12.3	Card Name Restart Table	7.12-4
7.12.4	Rigid Format Change Restart Table	7.12-10
7.12.5	File Name Restart Table	7.12-12
7.13	RESTART TABLES FOR MODAL TRANSIENT RESPONSE	7.13-1
7.13.1	Bit Positions for Card Name Restart Table	7.13-1
7.13.2	Bit Positions for File Name Restart Table	7.13-3
7.13.3	Card Name Restart Table	7.13-4
7.13.4	Rigid Format Change Restart Table	7.13-9
7.13.5	File Name Restart Table	7.13-11
7.14	RESTART TABLES FOR NORMAL MODES WITH DIFFERENTIAL STIFFNESS	7.14-1
7.14.1	Bit Positions for Card Name Restart Table	7.14-1
7.14.2	Bit Positions for File Name Restart Table	7.14-3

TABLE OF CONTENTS (Continued)

<u>Section</u>		<u>Page No.</u>
	7.14.3 Card Name Restart Table	7.14-4
	7.14.4 Rigid Format Change Restart Table	7.14-9
	7.14.5 File Name Restart Table	7.14-11
7.15	RESTART TABLES FOR STATIC ANALYSIS USING CYCLIC TRANSFORMATION	7.15-1
	7.15.1 Bit Positions for Card Name Restart Table	7.15-1
	7.15.2 Bit Positions for File Name Restart Table	7.15-3
	7.15.3 Card Name Restart Table	7.15-4
	7.15.4 Rigid Format Change Restart Table	7.15-8
	7.15.5 File Name Restart Table	7.15-10
7.16	RESTART TABLES FOR NORMAL MODES ANALYSIS USING CYCLIC TRANSFORMATION	7.16-1
	7.16.1 Bit Positions for Card Name Restart Table	7.16-1
	7.16.2 Bit Positions for File Name Restart Table	7.16-3
	7.16.3 Card Name Restart Table	7.16-4
	7.16.4 Rigid Format Change Table	7.16-8
	7.16.5 File Name Restart Table	7.16-10
7.17	RESTART TABLES FOR STATIC HEAT TRANSFER ANALYSIS	7.17-1
	7.17.1 Bit Positions for Card Name Restart Table	7.17-1
	7.17.2 Bit Positions for File Name Restart Table	7.17-3
	7.17.3 Card Name Restart Table	7.17-4
	7.17.4 Rigid Format Change Restart Table	7.17-8
	7.17.5 File Name Restart Table	7.17-10
7.18	RESTART TABLES FOR NONLINEAR STATIC HEAT TRANSFER ANALYSIS	7.18-1
	7.18.1 Bit Positions for Card Name Restart Table	7.18-1
	7.18.2 Bit Positions for File Name Restart Table	7.18-2
	7.18.3 Card Name Restart Table	7.18-3
	7.18.4 Rigid Format Change Restart Table	7.18-6
	7.18.5 File Name Restart Table	7.18-7
7.19	RESTART TABLES FOR TRANSIENT HEAT TRANSFER ANALYSIS	7.19-1
	7.19.1 Bit Positions for Card Name Restart Table	7.19-1
	7.19.2 Bit Positions for File Name Restart Table	7.19-3

TABLE OF CONTENTS (Continued)

<u>Section</u>		<u>Page No.</u>
	7.19.3 Card Name Restart Table	7.19-4
	7.19.4 Rigid Format Change Restart Table	7.19-8
	7.19.5 File Name Restart Table	7.19-10
7.20	RESTART TABLES FOR MODAL FLUTTER ANALYSIS	7.20-1
	7.20.1 Bit Positions for Card Name Restart Table	7.20-1
	7.20.2 Bit Positions for File Name Restart Table	7.20-3
	7.20.3 Card Name Restart Table	7.20-4
	7.20.4 Rigid Format Change Restart Table	7.20-11
	7.20.5 File Name Restart Table	7.20-13
7.21	RESTART TABLES FOR MODAL AEROELASTIC RESPONSE	7.21-1
	7.21.1 Bit Positions for Card Name Restart Table	7.21-1
	7.21.2 Bit Positions for File Name Restart Table	7.21-3
	7.21.3 Card Name Restart Table	7.21-4
	7.21.4 Rigid Format Change Restart Table	7.21-12
	7.21.5 File Name Restart Table	7.21-14
8.	STRUCTURAL ELEMENT DESCRIPTIONS	
8.1	INTRODUCTION TO STRUCTURAL ELEMENT DESCRIPTIONS	8.1-1
8.2	THE BAR ELEMENTS	8.2-1
	8.2.1 Input Data for the BAR Element	8.2-1
	8.2.2 Stiffness Matrix Calculation (Subroutine KBAR of Module SMA1)	8.2-2
	8.2.3 Lumped Mass Matrix Calculation (Subroutine MBAR of Module SMA2)	8.2-9
	8.2.4 Element Load Calculations (Subroutine BAR of Module SSG1).	8.2-9
	8.2.5 Element Stress Calculations (Subroutines SBAR1 and SBAR2 of Module SDR2)	8.2-11
	8.2.6 Differential Stiffness Matrix Calculation (Subroutine DBEAM of Module DSMG1)	8.2-15
	8.2.7 Piecewise Linear Analysis Calculations (Subroutine PSBAR of Module PLA3 and Subroutine PKBAR of Module PLA4)	8.2-18
	8.2.8 "Consistent" Mass Matrix Calculations (Subroutine MCBAR of Module SMA2)	8.2-22
	8.2.9 Thermal Analysis Calculations for the BAR Element	8.2-23

TABLE OF CONTENTS (Continued)

<u>Section</u>		<u>Page No.</u>
8.3	THE SHEAR PANEL AND TWIST PANEL ELEMENTS	8.3-1
8.3.1	Input Data for SHEAR and TWIST Panels	8.3-1
8.3.2	Definition of Element Geometry	8.3-2
8.3.3	Coefficient Generation	8.3-4
8.3.4	Stiffness Matrix Formulation for a SHEAR Panel (Subroutine KANEL of Module SMA1)	8.3-9
8.3.5	TWIST Element Stiffness Matrix Generation (Subroutine KANEL of Module SMA1)	8.3-10
8.3.6	Mass Matrix Generation (Subroutine MASSTQ of Module SMA2).	8.3-11
8.3.7	SHEAR Element Stress and Force Calculations (Subroutines SPANL1 and SPANL2 of Module SDR2)	8.3-13
8.3.8	TWIST Element Stress and Force Calculations (Subroutines SPANL1 and SPANL2 of Module SDR2)	8.3-15
8.3.9	SHEAR Panel Differential Stiffness Calculations (Subroutine DSHEAR of Module DSMG1)	8.3-17
8.4	TRMEM AND QDMEM ELEMENTS	8.4-1
8.4.1	Input Data for the TRMEM and QDMEM Elements	8.4-1
8.4.2	Basic Equations for TRMEM	8.4-2
8.4.3	Stiffness Matrix Calculation for TRMEM (Subroutine KTRMEM of Module SMA1)	8.4-4
8.4.4	Mass Matrix Calculation for the TRMEM Element (Subroutine MASSTQ of Module SMA2)	8.4-5
8.4.5	Element Load Calculations for the TRMEM Element (Subroutine TRMEM of Module SSG1)	8.4-6
8.4.6	Element Stress Calculations for the TRMEM Element (Subroutines STRME1 and STRME2 of Module SDR2)	8.4-6
8.4.7	Differential Stiffness Matrix Calculations for the TRMEM Element (Subroutine DTRMEM of Module DSMG1)	8.4-8
8.4.8	General Calculations for the QDMEM by the QDMEM Driver Routines (Subroutines KQDMEM of Module SMA1, SQDME1 of Module SDR2, DQDMEM of Module DSMG1)	8.4-11
8.4.9	Stiffness Matrix Calculations for the QDMEM	8.4-14
8.4.10	Element Stress Calculations for the QDMEM (Subroutine SQDME1 and STQME2 of Module SDR2)	8.4-14
8.4.11	Mass Matrix Generation for the QDMEM Element (Subroutine MASSTQ of Module SMA2)	8.4-18
8.4.12	Thermal Load Computation for the QDMEM	8.4-20

TABLE OF CONTENTS (Continued)

<u>Section</u>		<u>Page No.</u>
8.4.13	Differential Stiffness Computations for the QDMEM (Subroutines QDMEM and DTRMEM of Module DSMG1)	8.4-20
8.4.14	Piecewise Linear Analysis Calculations (Subroutines PSTRM and PSQDM of Module PLA3 and Subroutines PKTRM and PKQDM of Module PLA4)	8.4-21
8.5	THE TRBSC, TRPLT AND QDPLT ELEMENTS	8.5-1
8.5.1	Input Data for the TRBSC and TRPLT Elements	8.5-1
8.5.2	General Calculation for the TRBSC Element	8.5-2
8.5.3	Stiffness Matrix Calculations for the TRBSC Element (Subroutine KTRBSC of Module SMA1)	8.5-7
8.5.4	Stress Calculations for the TRBSC Element	8.5-8
8.5.5	Stiffness Matrix Calculations for the TRPLT Element (Subroutine KTRPLT of Module SMA1)	8.5-11
8.5.6	Structural Damping Matrices for the TRPLT Element	8.5-19
8.5.7	Stress and Element Force Calculations for the TRPLT Element (Subroutines STRPLT1 and SBSPL2 of Module SDR2) ..	8.5-19
8.5.8	Stiffness Matrix Calculations for the QDPLT Element (Subroutine KQDPLT of Module SMA1)	8.5-22
8.5.9	Stress and Element Force Calculations for QDPLT Element (Subroutine SQDPLT and SBSPL2 of Module SDR2)	8.5-27
8.5.10	Lumped Mass Matrix Generation for the TRBSC, TRPLT, and QDPLT Elements (Subroutine MASSTQ of Module SMA2)	8.5-29
8.5.11	Coupled Mass Matrix Calculations for the TRBSC Element (Subroutine MTRBSC of Module SMA2)	8.5-30
8.5.12	Mass Matrix Calculations for the TRPLT Element (Subroutine MTRPLT of Module SMA2)	8.5-36
8.5.13	Mass Matrix Calculations for the QDPLT Element (Subroutine MQDPLT of Module SMA2)	8.5-39
8.5.14	Thermal Load Equations for the Bending Elements (Sub- routines TRBSC, TRPLT, and QDPLT of Module SSG1)	8.5-43
8.7	THE ELASi, MASSi, AND DAMPi ELEMENTS	8.7-1
8.7.1	Input Data for the ELASi, MASSi and DAMPi Elements	8.7-1
8.7.2	ELASi Stiffness Matrix Generation (Subroutine KELAS of Module SMA1)	8.7-1
8.7.3	MASSi Mass Matrix Generation (Subroutine MASSD of Module SMA2)	8.7-2
8.7.4	DMAPi Damping Matrix Generation (Subroutine MASSD of Module SMA2)	8.7-2

TABLE OF CONTENTS (Continued)

<u>Section</u>		<u>Page No.</u>
8.7.5	ELASi Stress and Force Recovery (Subroutines SELAS1 and SELAS2 of Module SDR2)	8.7-2
8.8	CONCENTRATED MASS ELEMENTS C0NM1, C0NM2	8.8-1
8.8.1	ECPT Entries for the C0NM1 Mass Element	8.8-1
8.8.2	Mass Matrix Calculations for the CONM1 Element (Subroutine MC0NMX of Module SMA2)	8.8-2
8.8.3	ECPT Entries for the C0NM2 Mass Element	8.8-2
8.8.4	Mass Matrix Calculations for the C0NM2 Element (Subroutine MC0NMX of Module SMA2)	8.8-2
8.9	THE C0NEAX ELEMENT	8.9-1
8.9.1	Input Data for the C0NEAX Element	8.9-1
8.9.2	Stiffness Matrix Calculations (Subroutine KC0NE of Module SMA1)	8.9-1
8.9.3	Mass Matrix Computation (Subroutine MC0NE of Module SMA2).	8.9-2
8.9.4	Element Load Calculations (Subroutine C0NE of Module SSG1)	8.9-2
8.9.5	Element Stress Calculations (Subroutines SC0NE1, SC0NE2, SC0NE3 of Module SDR2)	8.9-7
8.9.6	Differential Stiffness Matrix Calculations (Subroutine DC0NE of Module DSMG1)	8.9-13
8.10	THE TRIARG ELEMENT	8.10-1
8.10.1	Input Data for the TRIARG Element	8.10-1
8.10.2	General Geometric Calculations	8.10-2
8.10.3	Integral Calculations	8.10-3
8.10.4	Elastic Constants Matrix Calculations	8.10-5
8.10.5	Stiffness Matrix Generation (Subroutine KTRIRG of Module SMA1)	8.10-6
8.10.6	Mass Matrix Calculations (Subroutine MTRIRG of Module SMA2)	8.10-8
8.10.7	Thermal Load Calculations (Subroutine TTRIRG of Module SSG1)	8.10-9
8.10.8	Element Force and Stress Calculations (Subroutines STRIR1 and STRIR2 of Module SDR2)	8.10-9
8.10.9	Thermal Analysis Calculations for the TRIARG and TRAPRG Elements	8.10-12
8.11	THE TRAPRG ELEMENT	8.11-1
8.11.1	Input Data for the TRAPRG Element	8.11-1

TABLE OF CONTENTS (Continued)

<u>Section</u>		<u>Page No.</u>
8.11.2	General Calculations	8.11-2
8.11.3	Integral Calculations	8.11-4
8.11.4	Elastic Constants Matrix Calculations	8.11-6
8.11.5	Stiffness Matrix Generation (Subroutine KTRAPR of Module SMA1)	8.11-6
8.11.6	Mass Matrix Calculation (Subroutine MTRAPR of Module SMA2)	8.11-8
8.11.7	Thermal Load Calculations (Subroutine TTRAPR of Module SSG1)	8.11-9
8.11.8	Element Force and Stress Calculations (Subroutine STRAP1 and STRAP2 of Module SDR2)	8.11-10
8.11.9	Thermal Analysis Calculations for the TRAPRG Element (Subroutine HRING by Module SMA1)	8.11-13
8.12	THE TØRDRG ELEMENT	8.12-1
8.12.1	Input Data for the TØRDRG Element	8.12-1
8.12.2	General Calculations	8.12-2
8.12.3	Integral Calculations	8.12-5
8.12.4	Elastic Constants Matrix Calculations	8.12-9
8.12.5	Stiffness Matrix Calculations (Subroutine KTØRDR of Module SMA1)	8.12-9
8.12.6	Mass Matrix Calculations (Subroutine MTØRDR of Module SMA2)	8.12-14
8.12.7	Thermal Load Calculations (Subroutine TTØRDR of Module SSG1)	8.12-15
8.12.8	Element Force and Stress Calculations (Subroutines STØRD1 and STØRD2 of Module SDR1)	8.12-17
8.13	THE VISC ELEMENT	8.13-1
8.13.1	Input Data for the VISC element	8.13-1
8.13.2	Damping Matrix Calculations (Subroutine BVISC of Module SMA2)	8.13-1
8.14	INTEGRAL CALCULATIONS FOR THE TRIARG, TRAPRG ELEMENTS	8.14-1
8.14.1	Integral Calculations for $q \geq 0$ and any p (Function DKINT)	8.14-3
8.14.2	Integral Calculations for $p \geq 0$ and $q < -1$ (Function DK89)	8.14-3
8.14.3	Integral Calculation for $p > 0$ and $q < -1$ (Function DK100)	8.14-4

TABLE OF CONTENTS (Continued)

<u>Section</u>		<u>Page No.</u>
8.14.4	Integral Calculations for $p > -1$ and $q = -1$ (Function DKJAB)	8.14-5
8.14.5	Integral Calculations for $p < -1$ and $q = -1$ (Function DK219)	8.14-5
8.14.6	Integral Calculations for $p = -1$ and $q = -1$ (Function DK211)	8.14-6
8.15	THE FLUID2, FLUID3, FLUID4, AXIF2, AXIF3, AXIF4 and MFREE ELEMENTS..	8.15-1
8.15.1	Input Data for the Fluid Elements	8.15-1
8.15.2	Matrix Calculations for the FLUID2 Element (Subroutine KFLUD2 of Module SMA1 and Subroutine MFLUD2 of Module SMA2)	8.15-1
8.15.3	Matrix Calculations for the FLUID3 Element (Subroutine KFLUD3 of Module SMA1 and Subroutine MFLUD3 of Module SMA2)	8.15-4
8.15.4	Matrix Generation for the FLUID4 Element (Subroutine KFLUD4 in Module SMA1 and Subroutine MFLUD4 in Module SMA2)	8.15-6
8.15.5	Matrix Calculations for the MFREE Elements (Subroutine MFREE in Module SMA2)	8.15-7
8.15.6	Stress Calculations for the AXIF Elements, Phase 1	8.15-7
8.15.7	Stress Calculations for the AXIF Elements, Phase 2	8.15-12
8.16	THE SLØT3 AND SLØT4 FLUID ELEMENTS	8.16-1
8.16.1	Input Data for the SLØT3 and SLØT4 Elements	8.16-1
8.16.2	General Calculations for the SLØT Elements	8.16-1
8.16.3	Stiffness Matrix Generation for the SLØT3 Element	8.16-2
8.16.4	Mass Matrix Generation for the SLØT3 Element	8.16-2
8.16.5	Stress Matrix Calculations in the SLØT Elements (Phase 1)	8.16-3
8.16.6	CSLØT1 Element, Phase 2	8.16-5
8.17	SOLID POLYHEDRA ELEMENTS, TETRA, WEDGE, HEXA1, HEXA2	8.17-1
8.17.1	Input Data for the Solid Polyhedra Elements	8.17-1
8.17.2	Basic Equations for the TETRA Element	8.17-2
8.17.3	Stiffness Matrix Generations for the TETRA Element (Subroutine KTETRA of Module SMA1)	8.17-3
8.17.4	Mass Matrix Generation for the TETRA Element (Subroutine MSØLID of Module SMA2)	8.17-3
8.17.5	Thermal Load Generation for the TETRA Element (Subroutine TETRA of Module SSG1)	8.17-3

TABLE OF CONTENTS (Continued)

<u>Section</u>		<u>Page No.</u>
8.17.6	Stress Calculations for the TETRA Elements (Subroutines SSØLID1 and SSØLID2 of Module SPR2)	8.17-4
8.17.7	Basic Equations for the WEDGE, HEXA1, and HEXA2 Elements .	8.17-5
8.17.8	Stiffness Matrix Calculations and Geometry Checks for the WEDGE, HEXA1, and HEXA2 Elements (Subroutine KSØLID for Module SMA1)	8.17-6
8.17.9	Mass Matrix Generation for the WEDGE, HEXA1 and HEXA2 Elements (Subroutine MSØLID of Module SMA2)	8.17-7
8.17.10	Thermal Load Generation for the WEDGE, HEXA1 and HEXA2 Elements (Subroutine SØLID of Module SSG2)	8.17-8
8.17.11	Stress Data Recovery for the WEDGE, HEXA1 and HEXA2 Elements (Subroutines SSØLID1 and SSØLID2 of Module SDR2).	8.17-8
8.17.12	Thermal Analysis Calculations for the Solid Elements	8.17-9
8.18	THE HBDY ELEMENTS	8.18-1
8.18.1	Input Data	8.18-1
8.18.2	Stiffness Matrix Calculations (Subroutine KHBDY of Module SMA1)	8.18-2
8.18.3	Capacity Matrix Calculations for the HBDY Element (Subroutine MHBDY of Module SMA2)	8.18-3
8.18.4	Convective Heat Flux Recovery for the HBDY Element (Subroutines SDHTF1, SDHTFF, and SDHTF2 in Module SDR2)	8.18-4
8.18.5	Area Factor Calculations for the HBDY Element (Utility Subroutine HBDY)	8.18-6
8.19	QDMEM1 ISOPARAMETRIC QUADRILATERAL ELEMENT	8.19-1
8.19.1	Input Data for QDMEM1 Element	8.19-1
8.19.2	Basic Equations for QDMEM1	8.19-2
8.19.3	Stiffness Matrix Calculation for QDMEM1 (Subroutine KQDMM1 of Module SMA1)	8.19-7
8.19.4	Mass Matrix Generation for the QDMEM1 (Subroutine MASSTQ of Module SMA2)	8.19-11
8.19.5	Element Load Calculations for the QDMEM1 Element (Subroutine QDMEM1 of Module SSG1)	8.19-11
8.19.6	Element Stress Calculations for the QDMEM1 Element (Subroutines SQDM11 and SQDM12 of Module SDR2)	8.19-14
8.20	THE QDMEM2 ELEMENT	8.20-1
8.20.1	Input Data for the QDMEM2 Element	8.20-1
8.20.2	Basic Calculations for the QDMEM2 Elements (Subroutine Q2BCS)	8.20-2

TABLE OF CONTENTS (Continued)

<u>Section</u>		<u>Page No.</u>
8.20.3	Subtriangle Calculations for the QDMEM2 Element (Subroutine Q2TRMS)	8.20-3
8.20.4	Stiffness Matrix Calculations for the QDMEM2 Elements (Subroutine QSDM2S of Module SMA1)	8.20-7
8.20.5	Mass Matrix Generation	8.20-9
8.20.6	Thermal Loads	8.20-9
8.20.7	QDMEM2 Element Stress and Force Calculations (Subroutines SQDM21 and SQDM22 of Module SDR2)	8.20-10
8.21	THE IHEX1, IHEX2, AND IHEX3 ELEMENTS	8.21-1
8.21.1	Input Data for the IHEXi Elements	8.21-1
8.21.2	Basic Equations for IHEXi Elements	8.21-2
8.21.3	Stiffness and Mass Matrix Calculation for IHEXi Elements .	8.21-6
8.21.4	Element Load Calculations for IHEXi Elements	8.21-7
8.21.5	Element Stress Calculations for IHEXi Elements	8.21-7
8.21.6	Differential Stiffness Calculations for IHEXi Elements ...	8.21-9
8.21.7	Heat Transfer Calculations for IHEXi Elements	8.21-10
8.22	THE TRAPAX ELEMENT	8.22-1
8.22.1	Input Data for TRAPAX Element	8.22-1
8.22.2	General Calculations	8.22-2
8.22.3	Integral Calculations	8.22-3
8.22.4	Elastic Constants Matrix Calculations	8.22-4
8.22.5	Stiffness Matrix Generation (Subroutine KTPZ of Module EMG)	8.22-4
8.22.6	Mass Matrix Calculations (Subroutine MPZDA of Module EMG)	8.22-7
8.22.7	Thermal Load Calculations (Subroutine TPZTEM of Module SSG1)	8.22-9
8.22.8	Element Stress and Force Calculations (Subroutines STPAX1, STPAX2, and STPAX3 of Module SDR2)	8.22-10
8.23	THE TRIAAX ELEMENT	8.23-1
8.23.1	Input Data for TRIAAX Element	8.23-1
8.23.2	General Geometric Calculations	8.23-2
8.23.3	Integral Calculations	8.23-3
8.23.4	Elastic Constants Matrix Calculations	8.23-3

TABLE OF CONTENTS (Continued)

<u>Section</u>		<u>Page No.</u>
8.23.5	Stiffness Matrix Generation (Subroutine KTRIA of Module EMG)	8.23-4
8.23.6	Mass Matrix Calculations (Subroutine MSTRIA of Module EMG)	8.23-6
8.23.7	Thermal Load Calculations (Subroutine TRTTEM of Module EMG)	8.23-8
8.23.8	Element Stress and Force Calculations (STRAX1, STRAX2, STRAX3 of Module SDR2)	8.23-9
8.24	TRIM6: LINEAR STRAIN TRIANGULAR ELEMENT	8.24-1
8.24.1	Input Data for TRIM6 Element	8.24-1
8.24.2	Basic Equation for TRIM6	8.24-2
8.24.3	Stiffness Matrix Calculation for TRIM6 (Subroutine KTRM6S and KTRM6D)	8.24-5
8.24.4	Mass Matrix Calculation for the TRIM6 Element (Calculated in the Stiffness Subroutine KTRM6S and KTRM6D)	8.24-8
8.24.5	Element Load Calculations for TRIM6 (Subroutine TLØDM6) ..	8.24-9
8.24.6	Element Stress Calculations for TRIM6 Element (Subroutine STRM61 and STRM62 of Module SDR2)	8.24-10
8.25	TRPLT1: HIGHER ORDER PLATE-BENDING ELEMENT	8.25-1
8.25.1	Input Data for TRPLT1 Element	8.25-1
8.25.2	Basic Equation for TRPLT1	8.25-2
8.25.3	Stiffness Matrix Calculation for TRPLT1 (Subroutines KTRPLS and KTRPLD)	8.25-10
8.25.4	Mass Matrix Calculation for TRPLT1 (Calculated in Stiffness Subroutines KTRPLS and KTRPLD)	8.25-13
8.25.5	Structural Damping Matrices for the TRPLT1 Element	8.25-14
8.25.6	Stress and Element Force Calculations for the TRPLT1 Element (Subroutines STRP11 and STRP12 of Module SDR2) ...	8.25-15
8.25.7	Thermal Load Calculations for the Bending Elements (Subroutines TLØDT1, TLØDT2 and TLØDT3 of Module SSG1) ...	8.25-17
8.26	TRSHL: SHALLOW SHELL TRIANGULAR ELEMENT	8.26-1
8.26.1	Input Data for TRSHL Element	8.26-1
8.26.2	Basic Equation for TRSHL	8.26-2
8.26.3	Stiffness Matrix Calculation for TRSHL (Subroutine KTSHLS and KTSHLD)	8.26-4
8.26.4	Mass Matrix Calculation for the TRSHL Element (Calculated in the Stiffness Subroutine KTSHLS and KTSHLD)	8.26-8

TABLE OF CONTENTS (Continued)

<u>Section</u>		<u>Page No.</u>
8.26.5	Structural Damping Matrices for the TRSHL Element	8:26-8
8.26.6	Stress and Element Force Calculations for TRSHL Element (Subroutines STRSL1, STRSLV and STRSL2 of Module SDR2) ...	8.26-9
8.26.7	Thermal Load Calculations for the TRSHL Element (Subroutine TLØDSL of Module SSG1)	8.26-9
8.26.8	Differential Stiffness Matrix Calculations for the TRSHL Element (Subroutine TRSHL of Module SSG1)	8.26-9
8.27	THE RØD, CØNRØD AND TUBE ELEMENTS	8.27-1
8.27.1	Input Data for the RØD, TUBE, CØNRØD Elements	8.27-1
8.27.2	Stiffness Matrix Calculation (Subroutines KRØD and KTUBE of Module SMA1)	8.27-2
8.27.3	Lumped Mass Matrix Calculation (Subroutines MRØD and MTUBE of Module SMA2)	8.27-3
8.27.4	Element Load Calculations (Subroutine EDTL of Module SSG1)	8.27-4
8.27.5	Element Stress Calculations (Subroutines SRØD1 and SRØD2 of Module SDR2)	8.27-4
8.27.6	Differential Stiffness Matrix Calculation (Subroutine DRØD of Module DSMG1)	8.27-6
8.27.7	Piecewise Linear Analysis Calculations (Subroutine PSRØD of Module PLA3 and Subroutine PKRØD of Module PLA4).	8.27-8
8.27.8	Coupled Mass Matrix Calculation (Subroutine MCRØD of Module SMA2)	8.27-11
8.27.9	Thermal Analysis Calculations for the RØD Elements	8.27-11
9.	EIGENVALUE EXTRACTION METHODS	
9.1	DETERMINANT METHOD OF EIGENVALUE EXTRACTION	9.1-1
9.1.1	Fundamentals of the Determinant Method	9.1-1
9.1.2	Evaluation of Determinant	9.1-1
9.1.3	Iteration Algorithm	9.1-2
9.1.4	Scaling	9.1-3
9.1.5	Sweeping of Previously Extracted Eigenvalues	9.1-3
9.1.6	Convergence Criteria	9.1-4
9.1.7	Extraction of Eigenvectors	9.1-6
9.2	GIVENS METHOD OF EIGENVALUE EXTRACTION	9.2-1
9.2.1	General	9.2-1
9.2.2	Reduction to Triangular Form	9.2-2

TABLE OF CONTENTS (Continued)

<u>Section</u>		<u>Page No.</u>
9.2.3	Computation of Eigenvalues for a Tridiagonal Matrix	9.2-4
9.2.4	Determination of Eigenvectors	9.2-9
9.2.5	Computer Program Flow	9.2-12
9.3	INVERSE POWER METHOD WITH SHIFTS	9.3-1
9.3.1	Summary of Procedures for Real Eigenvalue Analysis	9.3-1
9.3.2	Summary of Procedures for Complex Eigenvalue Analysis	9.3-3
9.4	THE TRIDIAGONAL REDUCTION (FEER) METHOD	9.4-1
9.4.1	Background for Real Eigenvalue Analysis	9.4-1
9.4.2	Problem Formulation	9.4-2
9.4.3	Summary of Computational Procedures and Flow Charts - Real Eigenvalue Analysis	9.4-5
9.4.4	Summary of Procedures for Complex Eigenvalue Analysis	9.4-14

1. NASTRAN PROGRAMMING FUNDAMENTALS

1.1 PROGRAM OVERVIEW

1.1.1 Objectives

The NASTRAN program has been designed according to two classes of criteria. The first class relates to functional requirements for the solution of an extremely wide range of large and complex problems in structural analysis with high accuracy and computational efficiency. These criteria are achieved by developing and incorporating the most advanced mathematical models and computational algorithms that have been proven in practice. In particular, they are achieved by providing such features as the bandwidth-with-active-column technique in matrix decomposition; packing routines to take maximum advantage of matrix sparsity so as to conserve input/output time; highly stable and efficient algorithms for the solution of problems in eigenvalue analysis and transient response; and an elegant approach to modeling the effects of control systems and other nonstructural components.

The second class of criteria relates to the operational and organizational aspects of the program. These aspects are somewhat divorced from structural analysis itself; yet they are of equal importance in determining the usefulness and quality of the program. Chief among these criteria are:

1. Simplicity of problem input deck preparation.
2. Minimization of chances for human error in problem preparation.
3. Minimization of need for manual intervention during program execution.
4. Ease of program modification and extension to new functional capability.
5. Ease of program extension to new computer configurations and operating systems, and generality in ability to operate efficiently under a wide set of configuration capabilities.
6. Capability for step by step problem solution, without penalty of repeated problem set up.
7. Capability for problem restart following unplanned interruptions or problem preparation error.
8. Minimization of system overhead, in the three vital areas:
 - a. Diversion of core storage from functional use in problem solution.

- b. Diversion of auxiliary storage units from functional to system usage.
- c. System housekeeping time for performing executive functions that do not directly further problem solution.

These criteria are achieved in NASTRAN through modular separation of functional capabilities, organized under an efficient, problem-independent Executive System.

This approach is absolutely essential for any complex multi-operation, multi-file application program such as NASTRAN. To see this, one must examine the implications of modularity in program organization.

Any application computer program provides a selection of computational sequences. These are controlled by the user through externally provided options and parameter values. Since no user will wish to observe the result of each calculation, these options also provide for the selection of the data to be output.

In addition to externally set options, internal switches whose setting depend upon tests performed during the calculations will control the computation sequences. There is, therefore, a natural separation of computations into functional blocks. The principal blocks are called functional modules; modules themselves of course may, and usually must, be further organized on a sub-modular basis.

Despite this separation, however, it is clear that modules cannot be completely independent, since they are all directed toward solution of the same general problem. In particular, they must intercommunicate data among themselves. The principal problem in organizing any application program, large or small, is designing the data interfaces between modules.

For small programs, the standard techniques are to communicate data via subroutine calling sequences and common data regions in core. For programs that handle larger amounts of data, auxiliary storage is used; however, strict specifications of the devices used and of the data record formats are usually imposed.

The penalty paid is that of "side effects". A change in a minor subroutine initiates a modification of the data interfaces that propagates through the entire program. When the program is small, these effects may not be serious. For a complex program like NASTRAN, however, they would be disastrous.

PROGRAM OVERVIEW

This problem has been solved in NASTRAN by a separation of system functions, performed by an Executive System, from problem solution functions, accomplished by modules separated strictly along functional lines. Each module is independent of all other modules in the sense that modification of a module, or addition of a new module, will not in general require modification of other modules. Even so, programming constraints on module development do exist but are minor. The essential restrictions are:

1. Modules may interface with other modules only through auxiliary storage files, as opposed to passing information between each other while in core.
2. Since the availability and allocation of auxiliary files for module execution interact with the execution of other modules, no module can specify or allocate files for its input or output data. All auxiliary storage allocation is reserved as an Executive function.
3. Modules operate as independent subprograms, and may not call, or be called by, other modules. They may be entered only from the Executive System.
4. Modules may interface with the Executive System through a parameter table that is maintained by the Executive System. User-specified options and parameters are communicated to modules in this way. The major line of communication is one-way, from user to Executive routine to module. However, in addition, an appreciable two way communication, from module back to executive routine (and therefore to other modules) is permitted via the parameter table.
5. Intra-module parameter communication is format-free in the sense that each module defines and orders its own local parameter set internally. Thus each module is independent of common data formatting by any other module.

No other constraints, except those imposed by the resident compilers and operating systems, are required for functional modules.

1.1.2 Program Organization

Because of the very large size of the NASTRAN program (more than 1500 decks and 800 individual overlay segments), execution as one physical program was not possible. However, to meet the stated design objectives, it was required that NASTRAN appear to the resident operating system as one program.

A program structure evolved which is basically computer independent, although the way in which the code structure is supported varies across the computers.

The NASTRAN program is divided into a series of logical pieces called links. Each link contains its own root segment (the set of subprograms which is always core resident for that link) and its own complete overlay structure. Each link is capable of performing a predefined subset of NASTRAN operations. Communication between links occurs through computer files. Control of the sequence of execution of the links is performed entirely by the NASTRAN program and requires no operator intervention. As a result of this approach, a NASTRAN program execution appears to the resident operating system as a normal batch job to be processed in the batch stream. Detailed descriptions of the way in which the link structure is implemented on each computer are given in section 5.

1.2 NASTRAN EXECUTIVE SYSTEM

1.2.1 Introduction

The essential functions of the Executive System are:

1. Establish and control the sequence of module executions according to options specified by the user.
2. Establish, protect, and communicate values of parameters for each module.
3. Allocate system files to all data blocks (a data block designates a set of data, matrix or table, occupying a file) generated during program execution. A file is "allocated" to a data block, and a data block is "assigned" to a file. The general data block I/O routine (GINØ) and the data card conversion routines (XRCARD and RCARD) are considered Input/Output utilities and are discussed separately in section 1.6.
4. Maintain a full restart capability for restoring a program execution after either a scheduled or unscheduled interruption.

The Executive System is open-ended in the sense that it can accommodate an essentially unlimited number of functional modules, files, and parameters. Modification of the Executive System necessary for change, addition, or extension of functional modules is restricted to changes in entries in control tables stored within the Executive routines.

Program execution is divided into two phases: 1) the Preface, in which modules XCSA, IFP1, XSØRT, IFP and XGPI are executed to: a) process the NASTRAN input data deck and b) perform general problem initialization; and 2) the program body itself, in which the sequence of program operations is controlled by the Operation Sequence Control Array (ØSCAR) Executive table, which was developed in the XGPI module of the Preface. A diagram of a sample NASTRAN input data deck is shown in Figure 1. Note that a NASTRAN input data deck consists of 3 separate decks: 1) the Executive Control Deck, 2) the Case Control Deck and 3) the Bulk Data Deck. A detailed description of the contents of the NASTRAN data deck is given in section 2 of the User's Manual. The flow of operations during the Preface is presented in Figure 2. The numbers in the blocks in Figure 2 refer to section numbers where more detailed explanations of the subroutines and modules can be found.

NASTRAN PROGRAMMING FUNDAMENTALS

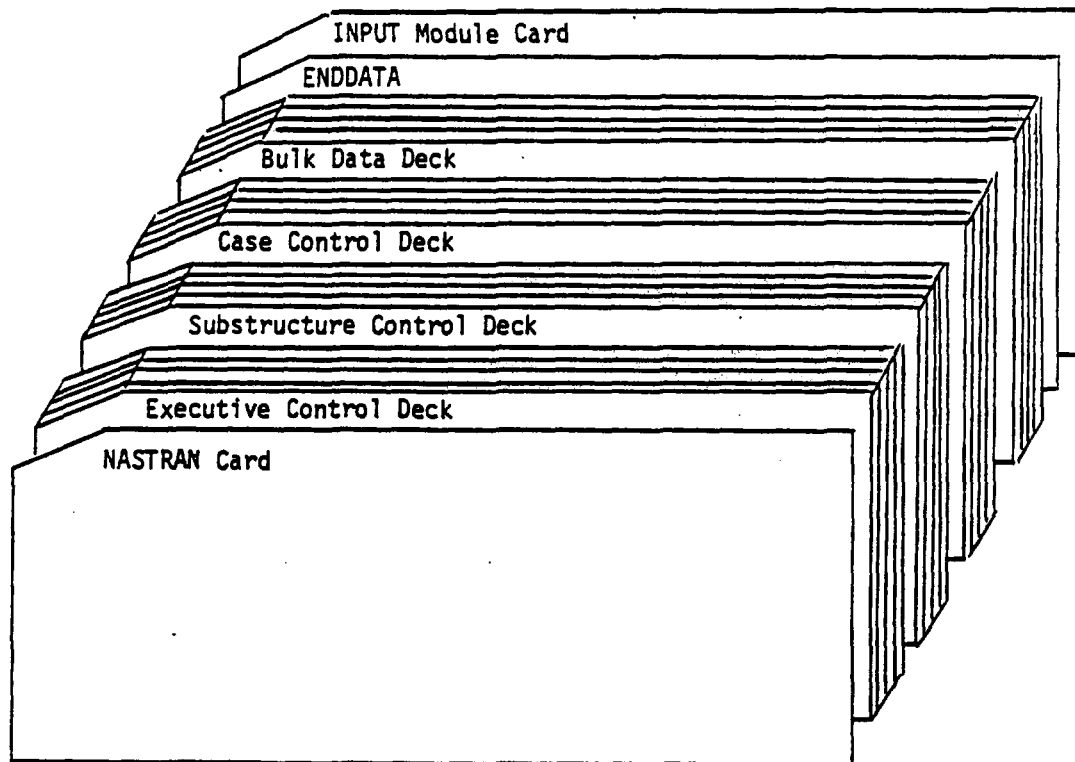


Figure 1. Sample NASTRAN Input Data Deck.

NASTRAN EXECUTIVE SYSTEM

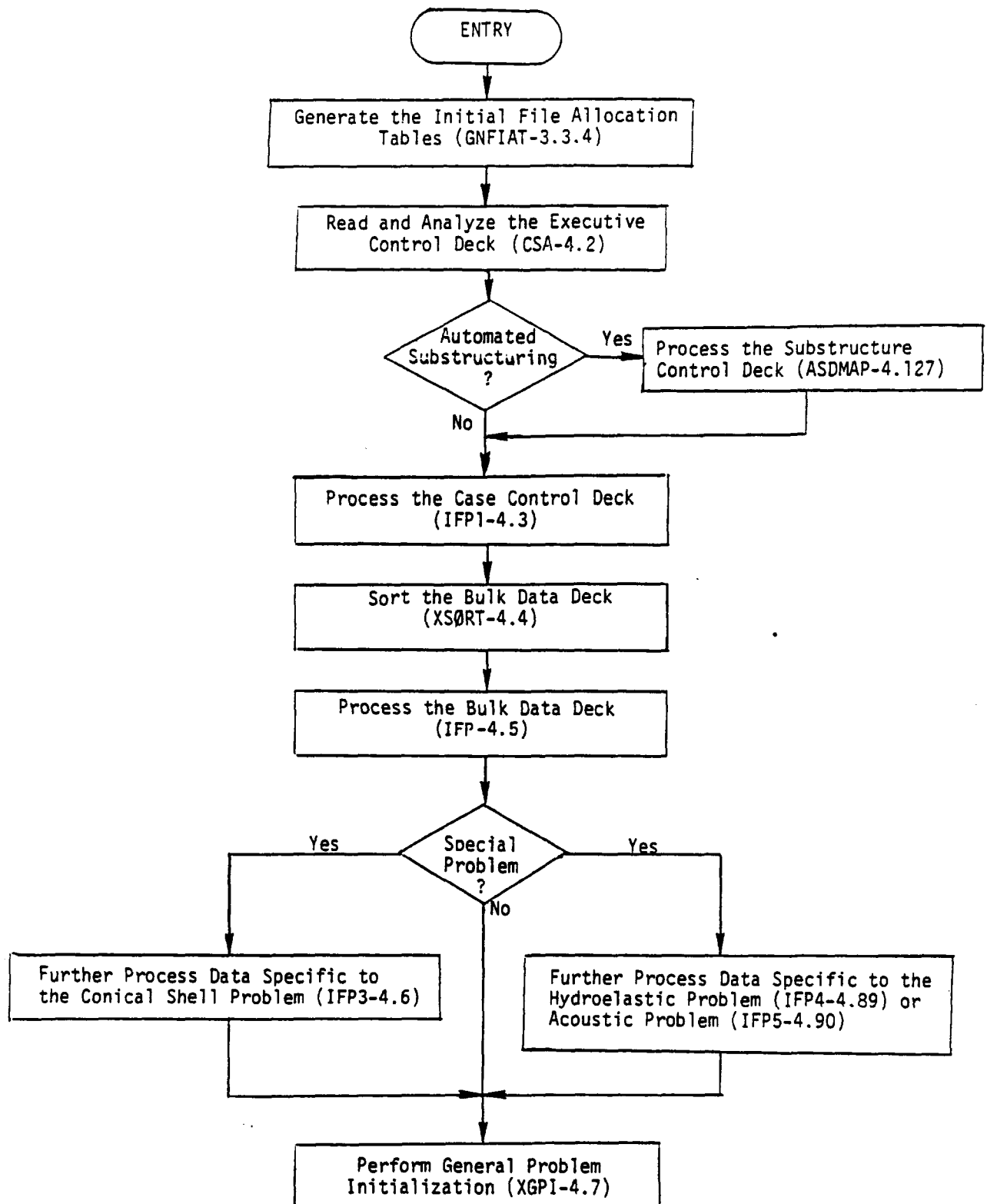


Figure 2. Flow of operations during the Preface.

1.2.2 Executive Operations During the Preface

The sequence of Preface operations shown in Figure 2 is controlled by the Sequence Monitor Initialization subroutine, SEMINT (see section 3.3.3). Each routine called by SEMINT is discussed in the following sections. The numbers in the section headings refer to section numbers where more detailed information on the subroutine or module can be found.

1.2.2.1 Generation of the Initial File Allocation Tables (GNFIAT section 3.3.4)

Two file allocation tables are maintained by the NASTRAN Executive System. One table, FIAT, (see section 2.4) defines the files to which data blocks generated during solution of the problem will be allocated. The second table, XFIAT, (see section 2.4) includes files to which permanent Executive data blocks, such as the New Problem Tape, the Old Problem Tape, plot tapes, and the User's Master File are assigned.

The New Problem Tape will contain those data blocks generated during the solution that are necessary for restarting the problem at any point. The Old Problem Tape contains the data blocks saved from some previous execution that may serve to bypass steps in the solution of the new problem. The User's Master File is a permanent collection of useful information, such as material properties, that may be used to generate input data.

The generation of the XFIAT and FIAT tables is a computer dependent operation since direct interface with the operating system of the computer must be made. The GNFIAT routine, which accomplishes this function, interrogates file tables in the nucleus of the operating system. Files which are available for use by the NASTRAN program are reserved, and the unit numbers for these files are stored in the NASTRAN file allocation tables. An indication of which units are physical tapes is also stored. If the number of files available is insufficient to run the problem, an error message is generated, and the run is aborted.

1.2.2.2 Analysis of the Executive Control Deck (XCSA Section 4.2)

The Executive Control Deck is processed and analyzed by the XCSA Executive Preface module. The Executive Control Deck includes cards which describe the nature and type of solution to be performed. This includes an identification of the problem, an estimated time for solution of the problem, the approach, a selection of the Rigid Format to be executed or an alternative sequence of NASTRAN operations (DMAP) to control the solution, a restart deck from a previous run if the

solution is to be restarted, an indication of any diagnostic printout to be made, a specification of whether the problem is to be checkpointed or not, and, if a Rigid Format is selected, any desired alterations to that format. Section 2 of the User's Manual should be consulted for the formats of, and restrictions on, each of the cards in the Executive Control Deck. The approach (APP) card, and the solution (SØL) card, which selects a particular solution (Rigid Format) to be executed, are worthy of special note. However, first some introductory definitions are required.

The sequence of operations to be executed during the program body is written in a data block oriented language called DMAP, an acronym for "Direct Matrix Abstraction Program". A DMAP instruction is a statement in the DMAP language, a DMAP sequence is a set of DMAP instructions, and a DMAP loop is a DMAP sequence to be repeated. A DMAP module is one which is "called" by means of a DMAP instruction.

A Rigid Format consists of: a) a fixed pre-stored DMAP sequence and b) its associated restart tables. A Rigid Format performs a specific (structural) problem solution. Section 3 of the User's Manual presents the DMAP sequence and the associated restart tables for each Rigid Format.

The APP card of the Executive Control Deck defines the problem solution approach. The APP card is required, and there are two options on the APP card: DISPLACEMENT or DMAP. The SØL card has the form

SØL n,m

where n = Rigid Format number, and m = a subset of the Rigid Format. The SØL card is required if the DISPLACEMENT option is chosen on the APP card. The SØL card must not be present in the deck if the DMAP option is chosen.

In addition to using the Rigid Formats provided automatically by NASTRAN, the user may wish either to execute a series of modules in a manner different from that provided by the Rigid Format, or to perform a series of matrix operations which are not contained in any existing Rigid Format. If the modifications to an existing Rigid Format are minor, the ALTER feature described in Section 2 of the User's Manual may be employed. Otherwise, a user-written Direct Matrix Abstraction Program (DMAP) should be used, in which case the card

APP DMAP

must be used. Chapter 5 of the User's Manual discusses DMAP.

Each of the cards comprising the Executive Control Deck is read via XRCARD (3.4.19) and analyzed. Depending on the card, information is either stored in various Executive tables maintained in core storage or written in the Executive Control Table (2.4.2.5) on the New Problem Tape for further processing during the general problem initialization phase (XGPI-4.7) of the Preface. Figure 3 presents the format of the Problem Tape. The formats of the New and the Old Problem Tapes are identical; only chronology defines their separate functions.

1.2.2.3 Processing of the Substructure Control Deck (ASDMAP, Section 4.127)

The substructure control deck is read and processed when the card APP=DISP,SUBS is defined in the Executive Control Deck, described above (Section 1.2.2.2). These data consist of substructure phase and operating file definitions, mode controls, and specific operational commands. This deck is read, decoded via XRCARD (Section 3.4.19), and processed to generate the following:

1. The Substructure Operating File (SØF) data which is used to initialize the system table /SØFCØM/ (Section 3.6.34) for subsequent I/O operations.
2. The parametric data for each substructure command which is written on the Case Control data block, CASECC, in the form of one record per command. Note that the header record is changed to read CASESS instead of CASECC.
3. DMAP ALTER card images for each substructure command which are copied to the New Problem Tape in the XALTER location shown in Figure 3. Any existing user-defined ALTER cards in the Executive Control deck are merged in sequence with the automatically-generated DMAP ALTERS.

In substructure analysis the user controls the restart of substructure operations by direct definition of the steps to be performed, and storage/retrievals of substructure data are processed internally. However, the user is allowed to execute a normal checkpoint/restart within the NASTRAN portion of a substructuring operation using the normal NPTP and ØPTP procedures.

1.2.2.4 Processing of the Case Control Deck (IFPI Section 4.3)

The Case Control Deck includes the following classes of cards: selection of specific sets of data from the Bulk Data Deck, selection of printed or punched output, definition of subcases definition of structural plots to be made, and definition of XY plots to be made. Section 2 of the User's Manual discusses in detail all cards of the Case Control Deck.

This deck is read via XRCARD (3.4.19) and processed. Information defining set selection, output selection and subcase definition is written into the Case Control data block, CASECC. Information defining plot requests is written in the Plot Control (PCDB) and XY Control (XYCDB) data blocks.

If the problem is a restart, a comparison with the Case Control Deck from the previous run is made. Differences are noted in an Executive restart table, which is used in the general problem initialization phase (XGPI-4.7) of the Preface.

1.2.2.5 Sorting of the Bulk Data Deck (XSORT Section 4.4)

The function of the XSORT routine is to prepare a file on the New Problem Tape (see section 1.2.2.1) which contains the sorted Bulk Data Deck (bulk data). Operation of the routine is influenced by the type of run. If the run is a cold start, the bulk data is read from the system input file (e.g. card reader) or the User's Master File, sorted, and written on the New Problem Tape. If the run is an unmodified restart, (restarts are discussed in section 1.10), the bulk data is copied from the Old Problem Tape (see section 1.2.2.1) to the New Problem Tape. If the run is a modified restart, the bulk data is read from the Old Problem Tape, and cards are deleted and/or added in accordance with cards in the system input stream. The modified bulk data is sorted and written on the New Problem Tape. Additionally, any changes in the data are noted in the Executive restart table.

A printed list of the unsorted bulk data is given if requested by an ECHO card in the Case Control Deck. Similarly, the sorted bulk data is echoed on request.

NASTRAN PROGRAMMING FUNDAMENTALS

THIS PAGE HAS BEEN LEFT BLANK INTENTIONALLY.

All files begin with an eight character (2 word) BCD header record.

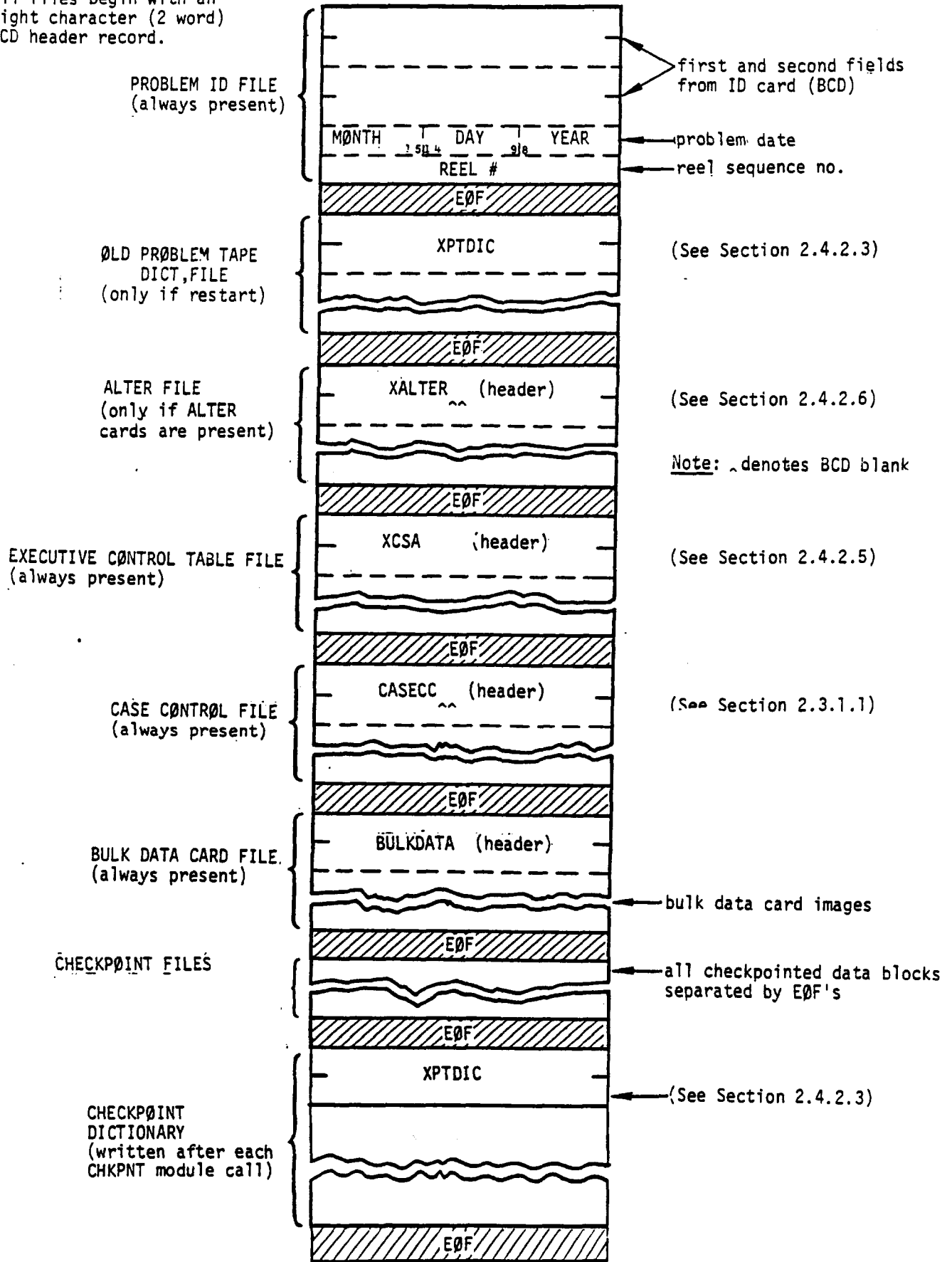


Figure 3. Problem tape format (same format for New Problem Tape and Old Problem Tape).

Since the collating sequence of alphanumeric characters varies from computer to computer, the sort routine converts all characters to an internal code prior to sorting. Following the sort, the characters are reconverted. In this way, the collating sequence is computer independent.

The algorithm used by the sort routine is biased toward the case where the data in sort or nearly in sort. Consequently, Bulk Data Decks which are nearly in sort will be processed efficiently by the routine.

1.2.2.5 Processing of the Bulk Data Deck (IFP Section 4.5)

The sorted Bulk Data Deck is read card-by-card from the New Problem Tape by the Input File Processor (IFP) and converted to internal binary form by RCARD (3.4.20). Each of the cards is checked for correctness of format. If any data errors are detected, a message is written, and a switch is set to terminate the run at the conclusion of the Preface. Section 2 of the User's Manual presents a detailed description of all cards of the Bulk Data Deck.

Processing of each bulk data card depends on the type of card. All bulk data cards of the same type are written into the logical record to which the card type has been assigned. These records are organized into data blocks classified according to general categories of use and written on prescribed preallocated files.

1.2.2.6 Processing of Conical Shell Data (IFP3 Section 4.6)

If the problem is a conical shell problem, further processing of the bulk data specific to the conical shell problem is accomplished. The nature of this processing is to convert data for the conical shell model into formats of a conventional statics problem. The result is that the conical shell problem can be described in a format convenient to the analyst and processed by NASTRAN in a format convenient to the program.

1.2.2.7 Processing of Hydroelastic Data (IFP4 Section 4.89)

If hydroelastic analysis data exists, this data must be converted to the data block formats and merged with existing data output from IFP. This module creates grid point, scalar point, element connection, and constraint data as well as producing a section in the MATP00L data block.

1.2.2.8 Processing of Acoustic Data (IFP5 Section 4.91)

If acoustic analysis data exists, the IFP5 module generates and merges grid points, scalar elements, and plotting elements with the existing data blocks.

1.2.2.9 General Problem Initialization (XGPI Section 4.7)

The Executive General Problem Initialization (XGPI) module is the heart of the Preface. Its principal function is to generate the Operation Sequence Control Array (ØSCAR-2.4.2.1), which defines the problem solution sequence. The ØSCAR consists of a sequence of entries, with each entry containing all of the information needed to execute one step of the problem solution. The ØSCAR is generated from information supplied by the user through his entries in the Executive Control Deck. This information is supplied by the SØL card, which points to a Rigid Format, or by a user supplied DMAP sequence.

The initial sequence of instructions was written in the Executive Control Table (2.4.2.5) on the New Problem Tape by the XCSA Preface module. This table is read to initiate assembly of the ØSCAR.

If the problem is a restart, the restart dictionary (contained in the Executive Control Table) and the Executive restart table are analyzed to determine which data blocks are needed to restart the solution and which operations in the ØSCAR need to be executed to complete the solution. Entries in the ØSCAR for operations not required for the current solution are flagged for no operation.

To aid in efficient assignment of data blocks to files, two attributes are computed and included with each data block in each entry of the ØSCAR. These attributes are: a) the ØSCAR sequence number when the data block is next used (NTU) and b) the ØSCAR sequence number when the data block is last used (LTU). Details of the file allocation are discussed in section 1.2.3.3.

When generation of the ØSCAR is complete, it is written on the Data Pool File (PØØL). If the problem is restart, data blocks needed for the current solution are copied from the Old Problem Tape to the Data Pool File.

1.2.3 Executive Operations During Problem Solution

1.2.3.1 Sequence Monitor (XSEMI Section 3.3.7)

When the Preface has been completed, solution of the problem is initiated. This solution is controlled by the sequence monitor. Figure 4 shows the flow for the sequence monitor. Note that there are i copies of XSEMI within NASTRAN, one controlling each link's operation. Section 1.1.2 defined the necessity for these divisions.

The sequence monitor reads an entry from the ØSCAR (2.4.2.1) which defines one step in the problem solution in terms of: the operation to be performed, data blocks required for input, data blocks to be output, scratch files required and parameters used. The File Status Table (FIST-2.4.1.3), which relates the internal data block reference numbers (see Section 1.6.4) to the file position in the File Allocation Table (FIAT-2.4.1.2), is created by the FIST generator, subroutine GNFIST. When the status table is complete, XSEMI moves the parameters required for the operation into blank common and calls the requested module (if within the current link) to begin the operation. If the requested module is not within the current link, ENSYS (see Section 3.3.5) is called and the Sequence Monitor within the new link is executed.

With the exception of XSFA, the seven routines described in the following subsections are Executive modules called directly by XSEMI to perform their specified functions.

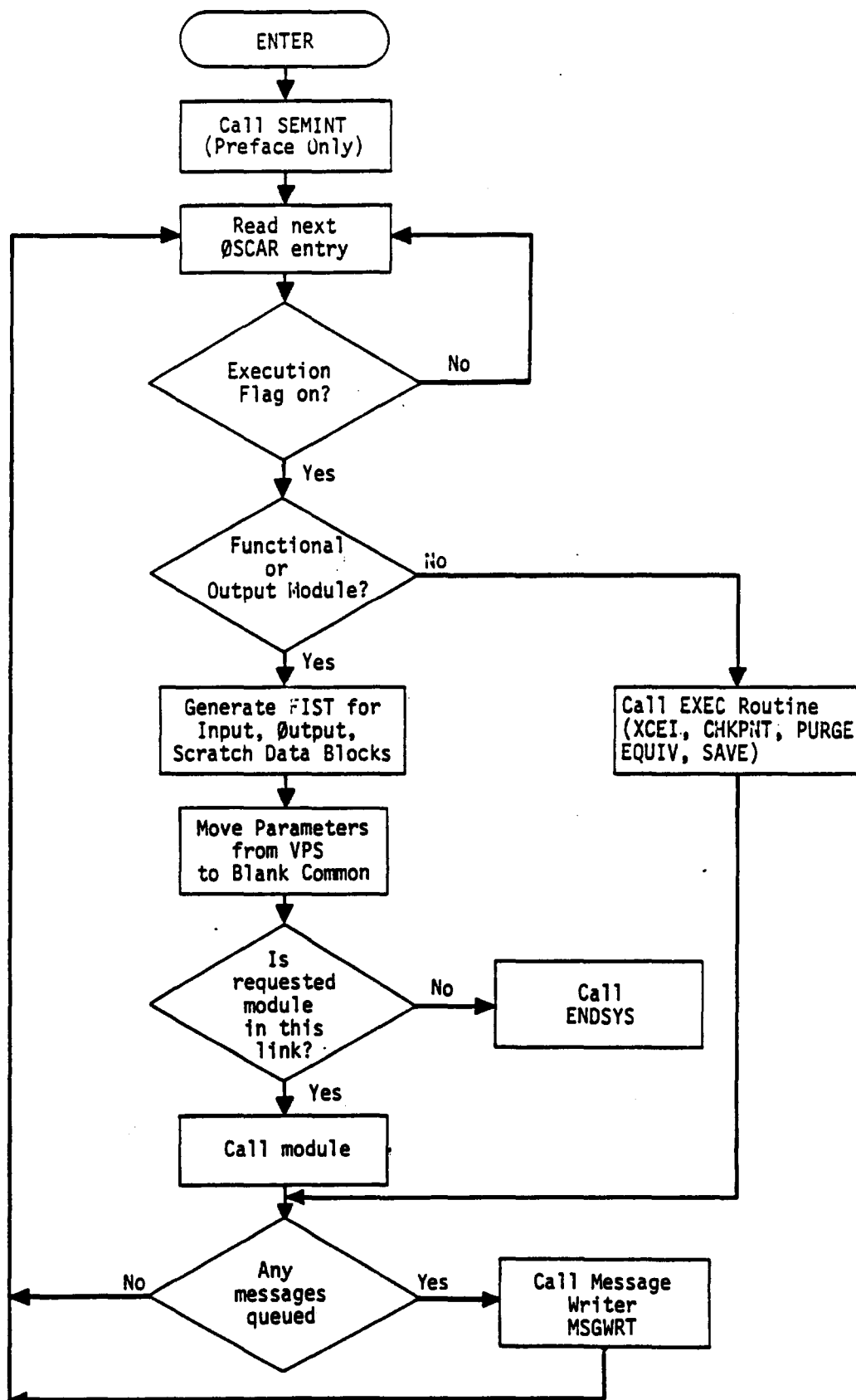


Figure 4. Flow diagram for the sequence monitor, XSEMI.

1.2.3.2 FIST Generator (GNFIST Section 3.3.9)

The FIST generator, subroutine GNFIST, creates the File Status Table (FIST), which contains the linkage between the internal data block reference numbers and the actual system files listed in the File Allocation Table (FIAT). Each input, output and scratch data block required by the forthcoming module is assigned an internal reference number if found to be active in FIAT. A data block found to be inactive, that is purged or not generated, will not be assigned a reference number. This missing reference number will cause the accessing module to be signaled regarding the inactive status. If, during the generation of the FIST, a data block is not found in the FIAT, active or inactive, the Executive Segment File Allocator (XSFA) module is called by GNFIST to make a file available to the subject data block.

1.2.3.3 Segment File Allocator (XSFA Section 4.9)

The Executive Segment File Allocator (XSFA) module, which is called exclusively by GNFIST, is the administrative manager of data blocks for NASTRAN. Since, in general, the number of data blocks required for solution of a problem far exceeds the number of files available, assignment of data blocks to files is a critical operation for efficient execution of NASTRAN.

The Executive Segment File Allocator module is called whenever a data block is required for execution of an operation but is not currently assigned to a file (i.e., does not appear in the FIAT). When the Segment File Allocator is called, it attempts to allocate not just for the data block initiating the call, but for as much of the remaining problem solution as possible. This allocation depends on the type of problem, the number of files available, and the range of use of the remaining data blocks.

The Segment File Allocator reads entries from the ØSCAR from the point of current operation to the end of the problem solution. The FIAT table entries are created in which attributes of the data blocks, including their next use (NTU) and last use (LTU), are stored. Data blocks which are currently assigned to files but are no longer required for problem solution are released. In certain cases, when the range of use of a data block is large, it may not be possible to allocate a file to the data block throughout its range of use. In this case, pooling of the data block is required so that the file to which the data block was assigned may be freed for another allocation. The next time used (NTU) attribute for a data block is used to efficiently pool data blocks. In general, the data block whose next use is the furthest from the current point is pooled, that is, copied onto the Data Pool File (PØØL). The format of the Data Pool File is shown in Figure 5.

One additional check is made with regard to pooling. The operation of the Segment File Allocator itself is less expensive than a pooling operation. Therefore, pooling occurs only when the module for which the allocation was required cannot be allocated without pooling.

When the Segment File Allocator is complete, a new File Allocation Table (FIAT) has been generated. This table is used until the solution again reaches a point where a data block is required to execute an operation but is not assigned to a file.

1.2.3.4 Interpretation of Executive Control Entries (XCEI Sections 4.11, 4.12, 4.13, 4.14)

Executive control entries include the DMAP instructions: REPT, JUMP, CØND and EXIT. Executive control entries in the ØSCAR are processed by the Executive Control Entry Interpreter (XCEI). When such an entry is encountered in the ØSCAR, the Control Entry Interpreter is called by XSEMi. If the operation is a jump, conditional jump or repeat, the ØSCAR is repositioned accordingly. If the operation is an exit, the NASTRAN termination routine PEXIT (3.4.22) is called.

1.2.3.5 Checkpointing Data Blocks (CHKPNT Section 4.10)

The checkpoint module (DMAP name: CHKPNT; entry point name: XCHK) copies specified data blocks required for problem restart onto the New Problem Tape and makes appropriate entries in the restart dictionary. This dictionary is also punched onto cards as each new entry is made. Thus, in the event of any unscheduled problem interruption, a restart from the last checkpoint

All files begin with an eight character (2 word) BCD header record.

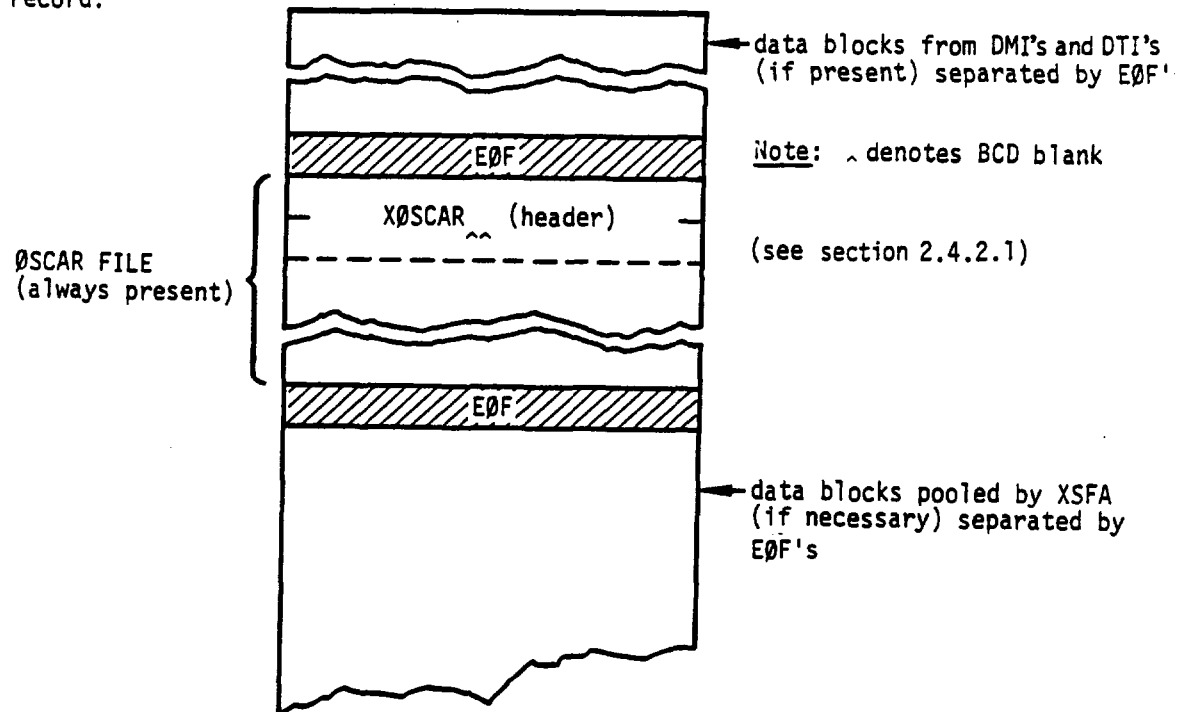


Figure 5. Format of the Data Pool File.

can be made using the Problem Tape and the restart dictionary from the interrupted run.

A checkpoint operation may also be requested implicitly by the declarative DMAP instruction PRECHK (Predefined Checkpoint), Section 4.1. Use of this facility allows shortened DMAP sequences.

1.2.3.6 Purging a Data Block (PURGE Section 4.16)

The purge routine (DMAP name: PURGE; entry point name: XPURGE) flags data blocks so that they will not be assigned to physical files. This special status provides a means for logically suppressing a segment of processing steps requiring the data block. Thus, if the function of a module is to multiply two matrices and add a third matrix to the product, the addition step might be deleted by purging the data block corresponding to the third matrix.

1.2.3.7 Equivalencing Data Blocks (EQUIV Section 4.17)

The equivalence routine (DMAP name: EQUIV; entry point name: XEQUIV) attaches one or more equivalent data block names to an existing data block. This special status provides a means of logically removing a module function by making a data block input to the module equivalent to a data block output from the module. Thus an entire module could be skipped, and an input data block "copied" to an output data block without physically moving the data from one file to another.

1.2.3.8 Saving Parameters (SAVE Section 4.15)

The save routine (DMAP name: SAVE; entry point name: XSAVE) provides a protection feature for the parameters communicated between, and used by, the functional modules. All variable parameters are stored within the VPS Executive table (see Section 2.4). Prior to each module's operation, the subset of parameters required by the module is moved to blank common. The module may use or modify this subset of parameters as desired. When the module terminates operation, only those parameters within the subset designated to be saved are restored to the Executive table. In addition to using the SAVE module explicitly to perform this function, the current DMAP syntax allows a SAVE instruction to be generated automatically. This user-directed option is activated in the actual DMAP statement as described in Section 5.2.1 of the User's Manual.

1.3 WORD SIZE AND COMPUTER HARDWARE CONSIDERATIONS

1.3.1 Introduction

Although NASTRAN is a FORTRAN oriented system, considerable effort was required to develop programming and word handling techniques applicable to three separate computer configurations. These computers exhibit wide differences in their binary word sizes and integer representation method. The current computer configurations considered and their significant differences follow:

1. Computer - IBM System 360/370 series

Word Size - 32 Bits

Character Capacity - 8 bits/character and 4 characters/word (character \equiv byte)

Integer Representation - twos complement for negative integers

2. Computer - UNIVAC 1108/1110

Word Size - 36 Bits

Character Capacity - 6 bits/character and 6 characters/word

Integer Representation - Ones complement for negative integers

3. Computer - CDC 6000/CYBER

Word Size - 60 Bits

Character Capacity - 6 bits/character and 10 characters/word

Integer Representation - Ones complement for negative integers

Various Executive routines (e.g., XSORT (4.4), XRCARD (3.4.19)) that deal directly with character strings from the input stream require some method of obtaining the above computer dependent information. Within the NASTRAN Preface, subroutine BTSTRP (3.3.2) solves an algorithm that determines which of the three computers is currently operating. This algorithm functions by inspecting the word length (by means of shifting and testing) and by checking the negative integer representation method. As a result of these tests, a word (MACH) within the SYSTEM Executive table (see section 2.4) is set to indicate the computer type. Since data within BTSTRP defines the number of bits-per-word (NBPW), the number of characters-per-word (NCPW), and the number of bits-per-character (NBPC) for each computer type, the correct values for these parameters are also stored into the SYSTEM table. This table resides within the NASTRAN root segment and is thus accessible to any module or subroutine.

1.3.2 Alphanumeric Data

Data stored within a computer as binary-coded-decimal (BCD) characters must be represented by the proper hardware defined bit codes. These character codes (and in the case of the IBM System/360, the number of bits representing the code) vary among the NASTRAN computer types. Although the number of characters-per-word could have been obtained from the SYSTEM table, various data blocks and buffers within NASTRAN required firm entry sizes, regardless of computer type, to facilitate indexing. For these reasons, the minimum number of characters-per-word (4) among the four computer types was chosen as a program design standard. Computer types with a word capacity of greater than four characters will have the unused low order character positions filled with BCD blanks.

1.3.3 Word Packing

Standard FORTRAN compilers do not provide the capability for storing or retrieving data that occupies less than a full computer word. Through the Machine Word Functions (MAPFNS, 3.4.1) routine some limited word packing (not to be confused with matrix packing) is performed within the Executive System and a few utility subroutines. Packing provides an efficient use of memory space at the expense of the additional operating time needed to combine or separate the elements of the packed words. The Machine Word Function ORF is generally used for combining elements, while ANDF with a suitable mask is used for separating them.

1.3.3.1 Examples of Machine Word Functions (MAPFNS) Usage

Assume three 10-bit items of data occupy the low order 10 bits of three separate 30-bit computer words (A, B, and C). To pack these three items into a single 30-bit word (X), perform the following steps using the individual functions available within MAPFNS:

- a) Left shift (LSHIFT) word A, twenty bits
- b) Left shift (LSHIFT) word B, ten bits
- c) Logically add (ORF) words A and B; store into X

WORD SIZE AND COMPUTER HARDWARE CONSIDERATIONS

d) Logically add (\oplus RF) words X and C; store into X.

Assume two 8-bit items of data are packed into the left and right halves of a 16-bit word (X). To unpack these two items into the low order 8 bits of two separate 16-bit words (A and B), perform the following steps using the individual functions available within MAPFNS:

- a) Create MASK containing 8 low order bits equal to 1 and the 8 high order bits equal to 0
- b) Right shift (RSHIFT) word X, eight bits; store into A
- c) Logically multiply (ANDF) word X by MASK; store into B.

In the preceding example, the word X remains unchanged since the functions return the requested modified result in a computer register.

1.4 SYSTEM BLOCK DATA SUBPROGRAM (SEMDBD)

NASTRAN contains a master block data program (SEMDBD) which is responsible for defining and initializing (root segment) common blocks. The common blocks referenced in SEMDBD are either Executive common blocks (XFIAT, XXFIAT, XFIST, etc.) which require initial values, or general information common blocks (SYSTEM, NAMES, TYPE, etc.) which are referenced by many modules. The source listing for SEMDBD identifies the common blocks, and it documents the data which are initialized, via comments. In addition, the Executive common blocks are documented in section 2.4 and the non-Executive common blocks in section 2.5. Certain parameters in these common blocks contain machine dependent values such as word size, number of BCD characters per word, etc. These values are set by subroutine BTSTRP (section 3.3.2) by identifying the machine on which the NASTRAN program is currently operating and setting the values accordingly.

1.5 THE OPEN CORE CONCEPT

1.5.1 Introduction

The design philosophy of the NASTRAN system dictated a completely open ended design whenever possible. NASTRAN was to have the flexibility to operate on a second generation machine with a 32K core (the IBM 7094/7040 DCS) as well as the largest of the IBM S/360 series of computers, and take complete advantage of the additional core storage without major program changes. The use of a fixed dimension for large arrays was outlawed since this automatically restricted the size of a problem that could be solved. Instead, modules were to be programmed to allocate space as required and to use spill logic to transfer data to scratch files if complete core allocation was impossible. In this manner, a problem might cause spill logic to be used on a computer with limited core storage, but not on a computer with a larger core storage capacity.

1.5.2 Definition of Open Core

The definition of open core is: a contiguous block of working storage defined by a labeled common block whose length is a variable determined by the NASTRAN Executive function CØRSZ. The implementation of this definition by the module writer consists of the originating of a labeled common block at the end of his overlay segment. This labeled common block contains a dimensioned variable of length 1. CØRSZ returns the number of words of core available between his open core origin and the end of core. The module writer can now write his program as if he had dimensioned his array by that number. In actuality, he is extending beyond the area reserved for the array into an area reserved for the job but not currently used by the segment. When implementing this concept, care must be taken to assure that the system does not use this area.

1.5.3 Example of an Application of Open Core

Figure 1 demonstrates the use of open core by two subroutines, A and B. By some means, which are machine dependent and are discussed in section 5, an end point is established for open core. The length of open core is then the difference between this end point and the labeled common block. In the example shown, subroutine A will have more open core available to it than B does.


```

SUBROUTINE A
COMMON // XX
COMMON /AX/ Z(1)
INTEGER CØRSZ
NZ = CØRSZ(Z(1),XX)
DØ 10 I = 1, NZ
10 Z(I) = I
RETURN
END
    
```

```

SUBROUTINE B
COMMON // XX
COMMON /BX/ Z(1)
INTEGER CØRSZ
NZ = CØRSZ(Z(1),XX)
DØ 10 I = 1,NZ
10 Z(I) = I
•
•
•
RETURN
END
    
```

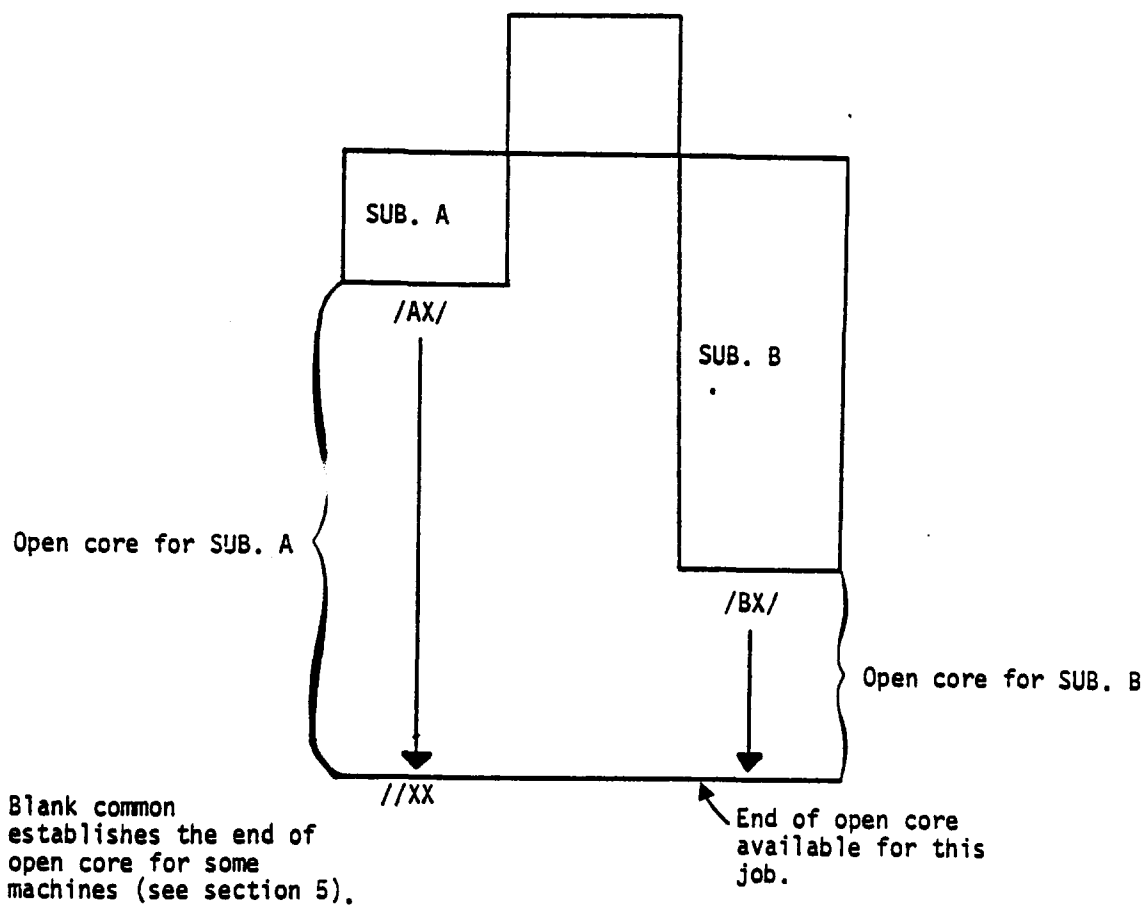


Figure 1. A example of the use of open core.

1.6 NASTRAN INPUT/OUTPUT

1.6.1 Introduction

The particular (IBM 7094, IBM S/360, Univac 1108, CDC 6600) operating system input and output files provide the required data connection between NASTRAN, the input data decks and the printed output. Utility subroutines XRCARD (section 3.4.19) and RCARD (section 3.4.20) convert special NASTRAN input card formats to standard FØRTRAN data words easily handled by all NASTRAN input processors. Printed output is generated through FØRTRAN formatted write statements. All internal data block input/output is handled by GINØ, the system of NASTRAN general purpose input/output routines. GINØ provides the required manipulation to tailor the variable length logical data records needed by most NASTRAN modules to fixed length records available on all direct access mass storage hardware.

1.6.2 Use of the Operating System Input File

The system input file is read only by the following routines within the NASTRAN Preface:

1. SEMINT (see section 3.3.3) reads the first card and processes it using utility XRCARD if it is the NASTRAN card (see section 6.3.1).
2. The Executive Control Deck containing free-field cards is read and processed by XCSA (section 4.2) using the XRCARD utility.
3. The Case Control Deck containing free-field cards is read and processed by IFP1 (section 4.3) using the XRCARD utility.
4. The Bulk Data Deck containing fixed-field cards is read by XSØRT (section 4.4). This data is subsequently processed by IFP (section 4.5) using the RCARD utility.

These card conversion utilities (XRCARD and RCARD) provide respectively all the free-field and fixed-field data card processing required by NASTRAN.

1.6.2.1 Use of the Subroutine XRCARD (See Section 3.4.19)

XRCARD interprets NASTRAN free-field data cards and processes the fields into a sequential buffer that can be easily handled by subsequent modules. Free-field data consist of series of data items separated by suitable delimiters and punched in non-specific card columns. Data items may include alphanumeric, integer, and various types of real variables. Field delimiters

may include the comma, slash, parenthesis, and blanks. For details regarding data and delimiter usage and the format of the sequential output buffer, see the XRCARD subroutine description in section 3.4.19.

1.6.2.2 Use of the Subroutine RCARD (See Section 3.4.20)

RCARD interprets NASTRAN fixed-field data cards and processes the fields into a sequential buffer that can be easily handled by subsequent modules. Fixed-field data consist of data items punched within specific card fields. Each eighty-column card is divided into an eight-column ID field (for the card mnemonic) followed by either eight eight-column data fields or four sixteen-column data fields. A special character (asterisk or plus) within the ID field determines when the card is to be interpreted as containing sixteen-column fields. The last eight columns of the card are for continuation mnemonics used by XSORT and are not processed by RCARD. The data item within the ID field must be alphanumeric. The data items within all other fields may include alphanumeric, integer, and various types of real variables. For details regarding data and the format of the sequential output buffer, see the RCARD subroutine description in section 3.4.20.

1.6.3 Use of the Operating System Output File

Although NASTRAN printed output is formed and placed onto the system output file through use of standard FORTRAN formatted write statements, two basic NASTRAN design concepts prohibit every operating module from generating printed output. Firstly, since the FORTRAN I/O package for output generation occupies a sizable block of computer memory, this package is generally positioned by loader directives within specific output oriented segments, rather than within the root segment of the overlay, to reduce the total memory requirement. Secondly, because many functional modules generate the same or similar diagnostic and information messages, a NASTRAN message writer (MSGWRT) was developed to centralize message text and thus prevent duplications within many separate modules.

For the reasons previously discussed, NASTRAN output generation is restricted to a specific class of modules which can reside within an output oriented segment below the link root segment. These segments will contain the output producing modules such as the Output File Processor (OFP - section 4.70), the Message Writer (MSGWRT - section 3.4.26), and the various table and matrix printers (TABPT - section 3.4.29, MATPRT - section 4.71, etc.) along with the output titling (PAGE - section 3.4.24) and necessary FORTRAN I/O packages.

1.6.4 GINØ

GINØ is a subroutine which provides for all input and output operations within NASTRAN except reading data from the resident system input file, writing data on the resident system output and punch files, writing plotter output, and transferring data to and from foreign programs. These latter operations are accomplished through FORTRAN formatted read/write statements. NASTRAN programs perform input/output operations by making the following calls to GINØ entry points (See Section 3.4.12):

1. ØPEN

ØPEN initiates activity for a file (unless the data block assigned to the file is purged, in which case an alternate return is given). A working storage area (GINØ buffer), for use by GINØ, is assigned (allocated) by the calling program thus providing optimum allocation or storage by the calling program. This working storage area is reserved for use by GINØ until activity on the file is terminated by a call to CLØSE (see paragraph 4 below).

2. WRITE

WRITE writes a specified (by the calling program) number of words on a file. The block of words to be written may comprise an entire logical record or portion of a logical record.

3. READ

READ returns to the calling program a specified (by the calling program) number of words from the logical record at which the file is currently positioned. READ may be used to transmit an entire logical record or portion of a logical record.

4. CLØSE

CLØSE terminates activity for a file. The working storage area assigned at ØPEN is released. The file is repositioned to the load point if requested.

5. REWIND

REWIND repositions the requested file to the load point. The file must be "open", i.e., a REWIND operation is requested subsequent to a call to ØPEN and prior to a call to CLØSE.

6. FWDREC

FWDREC repositions the requested file one logical record forward.

7. BCKREC

BCKREC repositions the requested file one logical record backwards. ;

8. SKPFIL

SKPFIL repositions the requested file forward or backward N logical files where N is specified by the calling program.

9. EOF

EOF writes a logical end-of-file on the requested file.

The basic unit of I/O in NASTRAN is a logical record. The length of a logical record is completely variable and may range from zero words to an arbitrarily large number of words. For NASTRAN matrix data blocks, the convention was adopted that each column of the matrix would comprise one logical record. For NASTRAN data blocks containing tables, no rigid convention exists. Typically each logical record contains one table of a specific type.

The logical record concept provides greatest ease in programming. However, since these records must be stored on a physical device such as a drum, disk or tape, the characteristics of the device must be taken into consideration. The bulk of NASTRAN data is stored on drums or disks. For both these devices the common unit of organization is a track, which stores a fixed number of words. Thus, there is a conflict between the variable length GINØ records and the fixed length tracks.

This conflict is resolved by blocking. GINØ acts as the interface between the device and the NASTRAN program. Using this technique, the program itself need not be concerned with device considerations (which would create machine dependent code). GINØ has been parameterized so that different devices may be easily accommodated.

Basically, blocking provides for the reading and writing of fixed-length blocks. The length of a block is a function of the device. It may be one track, one-half track or other integral division of a track (but never more than one track). Because of the relatively large time to access a given position on a track (due to the rotational speed of the device and/or mechanical movement of the head to the track), a block size equal to one full track is the most desirable. However, limitations in the amount of storage available to hold the blocks in core is a second consideration.

NASTRAN INPUT/OUTPUT

Since logical record lengths are variable but the length of records physically read or written is fixed, logic must be provided to accommodate this situation. This logic is provided in the GINØ routine, which allows for the following cases:

1. Multiple logical records per block
2. Multiple blocks per logical record

The method by which physical input and output of blocks is accomplished by GINO is machine dependent. On the IBM Systems 360/370 Series, BSAM macros are used. On the UNIVAC 1108 and 1110 Series, NTRAN is used. On the CDC 6000 and CYBER Series, CIØ macros are used. These implementation differences are transparent to the NASTRAN applications programmer (functional module writer). The systems programmer who is interested in implementation details on the various machines is referred to Section 5.

1.6.4.1 GINØ File Names

The names of files input as arguments to the GINØ routines listed above may be alphabetic (BCD, of the form 4HXXXX) or integer.

A GINØ file name is BCD if the file contains permanent Executive tables or data blocks. A list of these files for a particular NASTRAN run resides in the permanent portion of the FIST Executive table. The following list presents all current Executive files with their BCD file names:

<u>File</u>	<u>BCD File Name</u>
Data Pool File	PØØL
New Problem Tape	NPTP
Old Problem Tape	ØPTP
BCD Plot Tape	PLT1
Binary Plot Tape	PLT2
User's Master File	UMF
New User's Master File	NUMF
User Input File	INPT
User Input Files	INP1 - INP9
New Problem Tape Dictionary	XPTD

Functional modules should not access these permanent Executive files. Functional modules access the files on which their input, output and scratch data blocks reside by internal integer GINØ file names. Prior to calling a functional module, the link driver routine, XSEMI, calls subroutine GNFIST (GNFIST is called exclusively by XSEMI) to generate the FIST Executive table. For each input, output or scratch data block required for operation of a module (this information being contained in the ØSCAR entry), GNFIST searches the FIAT to find the data block. If the data block is in the FIAT and a file has been assigned to it, an internal GINØ file number denoting the data block and a pointer (index) to the entry in the FIAT is placed in the FIST. The following convention is used for internal GINØ file numbers: input data blocks -- 100 + position in the ØSCAR entry; output data blocks -- 200 + position in the ØSCAR entry; scratch data blocks -- 301 through 300 + n where n = number of scratch data blocks as defined in the MPL. (The position in the ØSCAR entry is the position in the DMAP instruction). If the data block is in the FIAT and is purged, no entry is placed in the FIST. For example, consider the following DMAP calling sequence for functional module XYZ:

```
XYZ      A,B,C/D,E,F,G/V,N,PARM1/V,N,PARM2 $
```

The data blocks input to the module are A, B and C; the data blocks output from the module are D, E, F and G; the module's parameters are PARM1 and PARM2. Note that internal scratch files are not mentioned in the DMAP calling sequence. The number of scratch files for a module is defined in the Module Property List (MPL) Executive table (see section 2.4) and is communicated to the Executive System via the ØSCAR. Details on the syntactical rules of DMAP are given in section 5 of the User's Manual.

In order to read the input data block B, the GINØ file number internal to XYZ is 102; in order to write data block D, the GINØ file number is 201. The third of, say, five scratch data blocks is referenced by XYZ through the GINØ file number 303.

1.6.5 SØF

The Substructure Operating File (SØF) is a permanent storage file for the user substructuring data. It is physically stored on a user diskpack, drum, or equivalent device, and is used to communicate data between different phases of a Multi-stage Substructuring problem.

In addition to the user data, the SØF contains tables created by the SØF utility subroutines which allow the full trace back of the substructuring process. The SØF input/output subroutines are:

1. SØFØPN

SØFØPN computes the addresses of the three in-core SØF buffers, and reads the SØF common blocks /SØF/ and /SYS/ into core.

2. SFETCH

SFETCH positions the SØF to read or write data to the SØF.

3. SUWRT

SUWRT copies a certain number (specified in the calling parameters) of data words belonging to an item from a given array onto the random access storage device.

4. MTRXØ

MTRXØ copies a matrix from a NASTRAN matrix file to the SØF. This is the only output interface routine between the NASTRAN data files and the SØF.

5. SUREAD

SUREAD reads a certain number of data words (specified in the calling parameters) belonging to an item into a given array.

6. MTRX1

MTRX1 copies a matrix from the SØF to a NASTRAN matrix file. This is the only input interface routine between the SØF files and the NASTRAN files.

7. SØFTRL

SØFTRL obtains the matrix control block of a matrix stored on the SØF.

8. SØFCLS

At the termination of a module, SØFCLS saves all of the in-core buffers and common blocks by writing them out to the direct access storage device.

9. SJUMP

SJUMP jumps over groups within an item, when in the read mode.

The SØF hierarchy is organized in a fashion analogous to the NASTRAN files. Equivalent to the GINØ record is the SØF group. The SØF item corresponds to the GINØ data block and is analogous to a GINØ file. An additional qualifier that exists for an SØF item which is not present for the GINØ file name is the substructure name. Each substructure-item combination identifies a specific SØF file. Further SØF details are discussed in Section 3.6.

The EXIØ module provides the capability to transfer selected SØF items to and from external user files. The EXIØ module performs these tasks in two modes. In one mode, all operations to and from external files are performed using GINØ for efficiency. In the second mode, all operations to and from external files are performed with FØRTRAN-formatted I/Ø. In this second mode, the external file may be used to transfer SØF data between different computer hardware systems. Subroutine EXFØRT performs this FØRTRAN-formatted input/output for the EXIØ module. Further details on the external-formatted I/Ø are discussed in Section 4.130.

1.7 NASTRAN MATRIX ROUTINES

1.7.1 Introduction

The requirement that NASTRAN handle large structural analysis problems implies that NASTRAN should be able to manipulate and store large matrices efficiently and effectively. In general, the matrices generated in the displacement approach tend to be sparse (i.e., the number of non-zero terms in any column of a matrix is small compared to the order of the matrix). The NASTRAN matrix routines, ADD, MPYAD, DECØMP, etc., which are described in section 3.5, are optimized as much as possible to take advantage of matrix sparsity and thus eliminate many unnecessary operation on zero elements. In order to aid in these operations and to make effective use of auxiliary storage, a packing scheme was devised to store only the non-zero terms in a column.

1.7.2 Matrix Packing and Unpacking

The need for a matrix packing routine can be seen by computing the auxiliary storage required to hold a 10,000 order matrix which is 1% dense (i.e., the average number of non-zero terms in a column is 100). With no packing technique, 10^8 words of storage are required to hold the matrix. Using the NASTRAN packing routines, a maximum of 2×10^6 words of storage are required if the terms are scattered, and 10^6 words are required if the terms occur in a band.

The routines BLDPK, INTPK, PACK, UNPACK, GETSTR, PUTSTR, GETSTB, along with their additional entry points, provide the matrix packing/unpacking capability of NASTRAN. The user should refer to the descriptions of these subroutines in sections 3.5.1 and 3.4.12 for a detailed description of the packing logic.

Matrices are stored by columns, and subroutines PACK/UNPACK provide the ability to pack/unpack a complete column. The capability is also provided to pack/unpack a column from the first non-zero element to the last.

An added feature of the packing routines is that subroutines BLDPK and INTPK provide the capability of packing/unpacking one element at a time. By use of INTPK, a matrix can be read element-by-element, such that an entire matrix can be processed without any appreciable core storage requirements. Likewise, by using BLDPK, a matrix can be built one element at a time. This is an extremely important feature to routines that must process matrices when storage is limited.

1.7.3 The Nested Vector Set Concept Used to Represent Components of Displacement

In constructing the matrices used in the Displacement Approach, each row and/or column of a matrix is associated closely with a grid point, a scalar point or an extra point. Every grid point has 6 degrees of freedom associated with it, and hence 6 rows and/or columns of the matrix. Scalar and extra points only have one degree of freedom. At each point (grid, scalar, extra) these degrees of freedom can be further classified into subsets, depending on the constraints or handling required for particular degrees of freedom. (For example in a two-dimensional problem all "z" degrees of freedom are constrained and hence belongs to the s (single-point constraint) set). Each degree of freedom can be considered as a "point", and the entire model is the collection of these one-dimensional points.

Nearly all of the matrix operations in displacement analysis are concerned with partitioning, merging, and transforming matrix arrays from one subset of displacement components to another. All the components of displacement of a given type (such as all points constrained by single-point constraints) form a vector set that is distinguished by a subscript from other sets. A given component of displacement can belong to several vector sets. The mutually exclusive vector sets, the sum of whose members are the set of all physical components of displacements, are as follows:

- u_m points eliminated by multipoint constraints and rigid elements,
- u_s points eliminated by single-point constraints,
- u_o points omitted by structural matrix partitioning,
- u_r points to which determinate reactions are applied in static analysis,
- u_l the remaining structural points used in static analysis (points left over),
- u_e extra degrees of freedom introduced in dynamic analysis to describe control systems etc.

The vector sets obtained by combining two or more of the above sets are (+ sign indicates the union of two sets):

- $u_a = u_r + u_l$, the set used in real eigenvalue analysis,
- $u_d = u_a + u_e$, the set used in dynamic analysis by the direct method,
- $u_f = u_a + u_o$, unconstrained (free) structural points,
- $u_n = u_f + u_s$, all structural points not constrained by multipoint constraints,
- $u_g = u_n + u_m$, all structural (grid) points including scalar points,

NASTRAN MATRIX ROUTINES

$u_p = u_g + u_e$, all physical points.

In dynamic analysis, additional vector sets are obtained by a modal transformation derived from real eigenvalue analysis of the set u_a . These are:

ξ_0 rigid body (zero frequency) modal coordinates,

ξ_f finite frequency modal coordinates,

$\xi_i = \xi_0 + \xi_f$, the set of all modal coordinates.

One vector set is defined that combines physical and modal coordinates. That set is

$u_h = \xi_i + u_e$, the set used in dynamic analysis by the modal method.

In aeroelastic analysis, additional vector sets are defined by the aerodynamic degrees of freedom. These are as follows:

u_k aerodynamic box and body coordinates

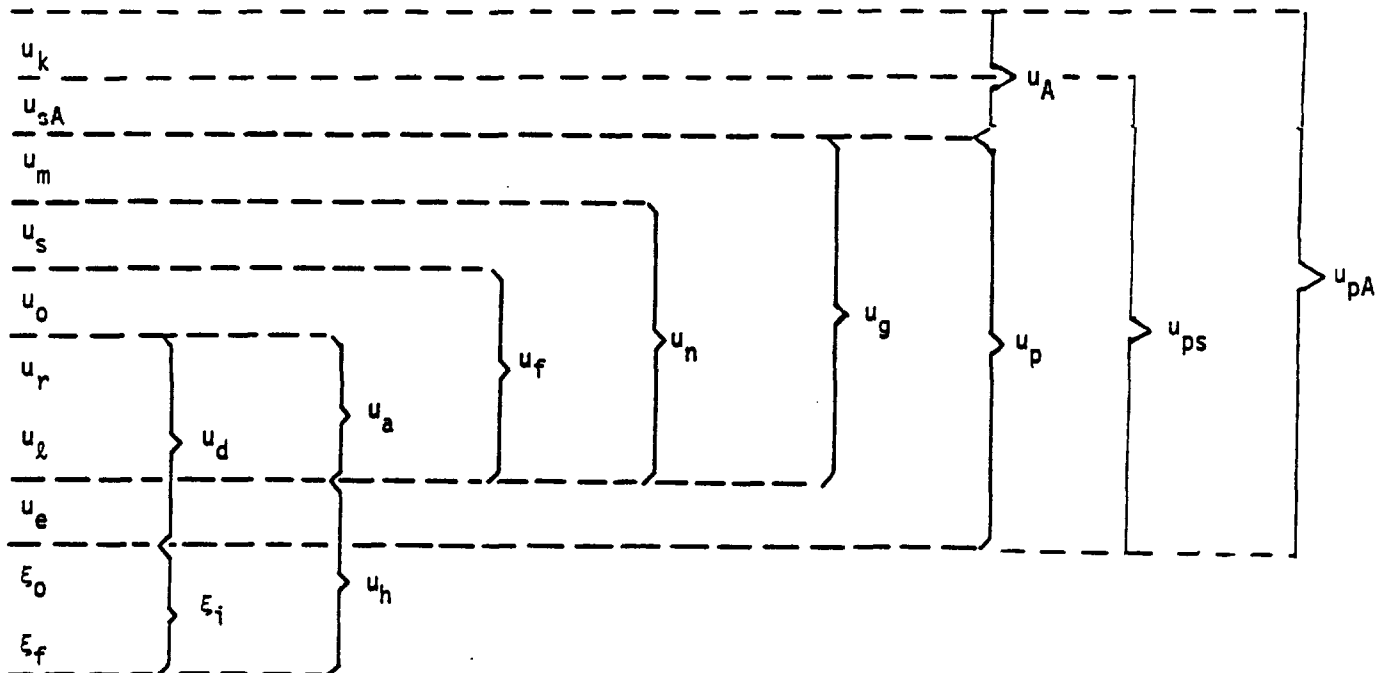
u_{sA} permanently constrained aerodynamic coordinates

$u_A = u_k + u_{sA}$, all aerodynamic coordinates

$u_{ps} = u_p + u_{sA}$

$u_{pA} = u_p + u_A$, all physical and aerodynamic coordinates

The nesting of vector sets is depicted by the following diagram:



The data block USET (USETD in dynamics and USETA in aeroelastic analysis) is central to this set classification. Each word of USET corresponds to a degree of freedom in the problem. Each set is assigned a bit in the word. If a degree of freedom belongs to a given set, the corresponding bit is on. Every degree of freedom can then be classified by analysis of USET. The common block /BITPOS/ relates the sets to bit numbers.

Two types of operations occur repeatedly. The first is the partitioning or sort operation. Examples are:

$$\{u_g\} \Rightarrow \begin{Bmatrix} u_n \\ u_m \end{Bmatrix} ; \quad (1)$$

and

$$[k_{nn}] \Rightarrow \begin{bmatrix} k_{ff} & k_{fs} \\ k_{sf} & k_{ss} \end{bmatrix} . \quad (2)$$

The second is the recombining (or merge) operation:

$$\{u_g\} \Leftarrow \begin{Bmatrix} u_n \\ u_m \end{Bmatrix} . \quad (3)$$

These operations can be completely described by a "partitioning" vector whose length corresponds to the length of the major set (the u_g set in Equation 1) and whose elements are zeros or ones depending on whether a degree of freedom belongs to the upper (the u_n set in Equation 1) subset or the lower (the u_m set in Equation 1) subset. Such a partitioning vector can be constructed by subroutine CALCV, which is described in section 3.5.5. This operation is described throughout the documentation by the notation USET (UG,UN,UM) where UG (u_g) is the major set, UN (u_n) is the zero set, and UM (u_m) is the one set. The partitioning vector generated by subroutine CALCV is input to the matrix routine PARTN (section 3.5.6) to perform operations similar to those in Equations 1 and 2 and is input to the matrix routine MERGE (section 3.5.6) to perform operations similar to that in Equation 3.

NASTRAN MATRIX ROUTINES

1.7.4 Processing of Matrices

Matrices in NASTRAN can be divided in two general types: core held matrices such as the 6x6's generated by the element routines and data block held matrices such as those output by functional modules. There are many routines to assist the programmer in the processing of both types of matrices. Incore matrices can be processed by GMMATD (Section 3.4.32), GMMATS (Section 3.4.33), INVERD (Section 3.4.34) and INVERS (Section 3.4.35). Data block held matrices can be processed at several levels. The most general is through the matrix packing and unpacking routines (BLDPK, PACK, INTPK and UNPACK). The next level of generality is provided by the matrix subroutines such as ADD, PARTN, MERGE, TRNSP, MPYAD, SDCOMP, DECOMP, CDCOMP, FBS, GFBS and INVTR. The functions provided by these routines can also be activated by a simple subroutine call through such routines as SSG2A, SDR1B, SSG2C, SSG2B, SSG3A, SOLVER, FACTOR and TRANP1. This third form is by far the most convenient and error free method for the novice NASTRAN applications programmer.

GENERATION OF MATRICES

1.8 GENERATION OF MATRICES

The Element Matrix Generator (EMG) module generates the stiffness, mass, and damping matrix partitions for each structural finite element. The EMA module assembles the element partitions into the NASTRAN format for each matrix type. If general elements are specified, the SMA3 module adds the general element stiffnesses to the EMA-generated matrix $[K_{gg}^x]$ to produce the complete stiffness matrix $[K_{gg}]$.

Other matrix generation modules are: 1) DSMG1, which generates the differential stiffness matrix, $[K_{gg}^d]$, for use in the Static Analysis with Differential Stiffness Rigid Format and in the Buckling Analysis Rigid Format; 2) PLA1, which generates the stiffness matrix for linear elements, $[K_{gg}^l]$, for use in the Piecewise Linear Analysis Rigid Format; 3) PLA4, which generates the stiffness matrix for nonlinear elements, $[K_{gg}^{nl}]$, for use in the Piecewise Linear Analysis Rigid Format; 4) MTRXIN, which provides a two-fold capability: a) to provide for direct input matrices such as control systems in the dynamics Rigid Formats, and b) to provide the DMAP user a capability of converting matrices input on DMIG bulk data cards to NASTRAN matrix format; and 5) the IFP module which provides the user the capability of converting matrices input on DMI bulk data cards to NASTRAN matrix format. Detailed information on each of these modules can be found in Section 4, Module Functional Descriptions.

1.8.1 The EST, EDPT, and GPECT Data BLOCKS

NASTRAN embodies a lumped element approach, i.e., the distributed physical properties of a structure are represented by a model consisting of a finite number of idealized substructures or elements that are interconnected at a finite number of points. An element will affect terms in the matrices only in rows and columns related to its interconnected points. Hence, each column of these matrices may be formed using only the matrix data for elements connected to the grid or scalar point associated with that column.

The Table Assembler (TA1) module generates the element tables used for the matrix assembly process. Alternate paths are taken in NASTRAN depending on which assembly modules will be executed. When EMG is used to generate matrices, the table assembler generates the Element Summary Table (EST) and the Grid Point-Element Connection Table (GPECT). The EST table groups the data by element type. It contains the connections, properties, grid point geometries, and a reference to the temperatures for every element. The GPECT table contains only the connection

data and is ordered by connected grid point indices. Each record of the GPECT contains all the connection data for one grid or scalar point in the model. This table is used in EMA to reallocate space for nonzero terms in the matrix.

When the DSMG1, PLA1, or PLA4 modules are used to generate stiffness matrices, the Element Connection and Property Table (ECPT) data block is generated instead of the GPECT. The ECPT data block is ordered exactly the same as the GPECT table, but contains all property, geometry, and temperature data for each element. The data for each element in the ECPT is exactly the same as in the EST.

1.8.2 Structural Elements

The basis for the structural matrices in NASTRAN are the finite structural and scalar elements. Each element generates matrix terms connecting and connected to the grid and scalar points given on its input connection card (e.g., CRØD). A structural element generates 6 by 6 matrix partitions related to the six degrees of freedom of each connecting grid point. A scalar element generates one term for each connection.

The stiffness matrix, $[K]$, for a structural element consists of a partition for each combination of the connected grid points. For example, a BAR or RØD element is connected to two grid points, "a" and "b". The stiffness matrix partitions are: $[K_{aa}]$, $[K_{ab}]$, $[K_{ba}]$ and $[K_{bb}]$. A triangular element (e.g., TRMEM) is connected to three points. It will generate nine partitions: $[K_{aa}]$, $[K_{ab}]$, $[K_{ac}]$, $[K_{ba}]$, $[K_{bb}]$, $[K_{bc}]$, $[K_{ca}]$, $[K_{cb}]$ and $[K_{cc}]$.

Although the actual equations for the element stiffness, mass and damping matrices are different for each element, the solutions follow a definite pattern. The element connection, orientation and property data are given in the EST and ECPT data blocks. The coordinate system data for orienting the global coordinates at each grid point are given in the CSTM data block. The material properties are given in the MPT and DIT data blocks. A utility routine, PRETRD, is available to fetch coordinate system data, and a utility routine, PREMAT, is available to fetch material properties.

1. An element coordinate system is calculated using the locations of the grid points. Using these data, a matrix $[E]$ is formed which transforms displacements from element coordinates to basic coordinates.

GENERATION OF MATRICES

2. The stiffness matrix may be formed in element coordinates using many methods. For the simple elements (e.g., RØD) the terms are direct functions of the geometry, properties and material coefficients of the element. For some elements the matrix is first formulated in terms of generalized coordinates, $\{q\}$, usually the coefficients of a power series. In general coordinates, the matrix is $[K_q]$. Transformation matrices, $[H_i]$, are generated to transform the displacements at the grid points in element coordinates $\{u\}$, to the general coordinates $\{q\}$.
3. The global coordinate system orientation matrix, $[T]$, for each grid point is calculated.
4. The stiffness matrix partition for the columns related to point j and the rows related to point i is $[K_{ij}]$. In general it is formed by the equation

$$[K_{ij}] = [T_i]^T [E] [H_i]^T [K_q] [H_j] [E]^T [T_j] \quad (1)$$

In the EMG/EMA method of matrix assembly, all partitions $[K_{ij}]$ are generated for each element defined in the EST table. These partitions are organized and stored on a direct access device so that the rows and columns are sequenced according to the grid point indices. The mass and damping matrices are generated and stored in a similar manner. The EMA module retrieves the element matrix partitions, independent of the type of elements, and assembles the complete, packed matrix.

In the PLA1, PLA4, DSMG1 method of matrix assembly, the ECPT table is used. For each grid point i , which is called the "pivot" point, the element routine is called to generate only the matrix partition $[K_{ij}]$. The remainder of the element matrix is generated in subsequent calls to the element routine with each new "pivot" point in the sequence of the ECPT.

TERMINATION PHILOSOPHY AND DIAGNOSTIC MESSAGES

1.9 TERMINATION PHILOSOPHY AND DIAGNOSTIC MESSAGES

Certain restrictions are placed upon the functional module writer with regard to run termination and error diagnostics. This is necessary in order to complete certain functions upon terminating and to maintain uniformity with regard to diagnostic messages.

A functional module writer is required to utilize a message writer (MSGWRT, section 3.4.26) to print all of his messages. In this manner similar message formats do not have to be duplicated in each module. Also, in order to avoid placing the I/O conversion routines and the lengthy format statements in the root segment, the message writer is restricted to its own overlay segment. Communication between the module and the message writer is via a queued message concept. Subroutine MESSAGE (section 3.4.25) is called to store the message parameters. In the case of a fatal message, a dump is taken if a DIAG 1 card is present in the Executive Control Deck, and PEXIT (section 3.4.22) is called. For non-fatal messages, the message is queued and control given back to the user. The message queue is printed after each module is executed.

In order for any routine to terminate the current execution, a call to PEXIT must be made. PEXIT handles all the functions necessary to wrap up the run: flushing output buffers, printing queued messages, and punching the last card of the checkpoint dictionary.

1.10 RESTARTS IN NASTRAN

NASTRAN is designed to run large problems with long running times. Even with the best of computer systems, a hardware, operator, or system failure is not uncommon for long running jobs. At the same time, the large volumes of data and the complexity of the structures that can be modeled and analyzed using NASTRAN make it highly likely that user input data errors will occur. Many of these errors are of a subtle type, meaning that they cannot be immediately detected in the NASTRAN Preface by the modules which process the data decks. To deal with these problems, and to save machine time on runs which abort because of data or system errors, NASTRAN has a sophisticated checkpoint and restart capability (see section 3 of the User's Manual for a discussion of restarts from the user point of view). The overall design philosophy for restart is twofold. A restart selectively executes only the modules necessary to accomplish a user-input data change. The user is able to change any part of his problem including structural model changes, additional cases, or more output requests. At the same time restarts are automatic as far as user interference is concerned. The user need only checkpoint (see section 1.2.3.5) his original run and submit changes to the original run on subsequent runs. The user does not have to analyze the effect of his changes. In addition the selective nature of restart allows the program to proceed with implicit errors (errors present in the data but not yet identified) until no further valid progress can be made. The work accomplished to this point is not lost, but rather only the table or matrix data block in error must be corrected to allow the program to proceed. Much error checking can be deferred until the actual module using the data is in control. The remainder of this section will explain the program mechanics by which restart is accomplished.

In NASTRAN there are four general types of restarts. Unmodified Restart (UMR), Psuedo Modified Restart (PMR), Modified Restart (MR), and Rigid Format Switch (RFS). Note that in the User's Manual UMR's and PMR's are described together as Unmodified Restarts. These classifications are for descriptive and internal purposes. The user need not know anything about which type is which. The basic characteristics of each type will be described below. An Unmodified Restart results when the user simply resubmits a problem with no data changes. It is used to continue a solution from the point of interruption. Presumably the problem aborted previously due to time expiring, machine error, system error, etc. The restart dictionary (created while checkpointing) is processed, and the solution is started again at the last re-entry point (after the last successful checkpoint). A Psuedo Modified Restart occurs when the user requests additional output from the program which simply requires the re-execution of an output module such as the Structure Plotter, the Grid Point

Weight Generator, or the Stress Data Recovery module etc. The numerical solution is not affected by these modules; only output is generated. Note that a PMR is the common case since printer output, plotter output, etc. is usually requested. A true UMR is rare. On a PMR, output modules are re-executed to display requested output, and then the problem is continued at the re-entry point. A true Modified Restart occurs when some numerically significant data change. The modules which process this type of data must be re-executed. These modules are re-executed to regenerate their output data blocks based on the new data, and the problem is continued at the re-entry point. A Rigid Format Switch is a special form of Modified Restart in which a problem changes from one solution type to another. One example would be: a user has solved for the static solution on Rigid Format 1 and now wants to find the normal modes by using Rigid Format 3. This may or may not require data changes. The key difference here is that the re-entry point cannot be used to determine the proper place to restart. The technique by which a RFS is accomplished is to re-execute the final modules on the new Rigid Format and let the File Name Table chain the solution back to the proper restart point.

To understand, in general, how the above types of restart are implemented, it is necessary to consider the Module Execution Decision Table (MEDT), which is associated with each Rigid Format. The Module Execution Decision Table is a table which has one entry for every DMAP instruction in the Rigid Format. Each entry is 5 words long; each word contains 31 bits. For convenience, these bits are numbered sequentially from left to right with the numbers 1 through 155. If the entry in the MEDT for a module has, say, bit 55 turned on, this module will be executed whenever a card or data block change associated with bit 55 occurs on a restart. The Card Name Table associates bits of the MEDT with selected bulk data card names, Case Control selections and parameter names. The File Name Table associates bits of the MEDT with selected data block names. For consistency, bits 1 through 62 for each entry in the MEDT are reserved for the Card Name Table, and bits 94 through 155 are reserved for the File Name Table. The following example illustrates the use of these tables in determining the effects of changing a bulk data card on a Modified Restart. Suppose the FØRCE bulk data card is to be changed when executing Rigid Format 1. The table in section 10.2.1 of the Programmer's Manual associates bit 60 with the FØRCE card. The decision table for bits 1 through 62 is shown in section 10.2.3 of the User's Manual. DMAP modules BEGIN, FILE, FILE, GP3, SAVE, PARAM, CHPNT, SSG1, CHPNT, EQUIV, etc., will be executed since bit 60 is on for each.

There is one more table, the Rigid Format Switch Table, which is constant for all Rigid Formats, and hence resides in Preface module XCSA. The Rigid Format Switch Table associates with

RESTARTS IN NASTRAN

each Rigid Format a bit for each entry in the MEDT: bit 63 is associated with Rigid Format 1, bit 64 is associated with Rigid Format 2, etc. If the restart involves a Rigid Format change, the bit in the decision table which is set is the bit corresponding to the Rigid Format of the previous execution.

Each part of the NASTRAN Preface contributes to processing the information for a restart. XCSA extracts and stores the Card Name Table, the File Name Table, the Module Execution Decision Table, the DMAP sequence and the Rigid Format Switch bit if any. These are written in the XCSA Executive Control Table (see section 2.4.2.5) on the New Problem Tape for later use by module XGF. IFP1 compares the current CASECC data block with the one submitted on the previous run (a copy of CASECC is stored on the Old Problem Tape for this purpose). Three types of changes are noted by IFP1:

1. Changes in data set selection such as Load set, SPC set, etc.;
2. The occurrence of output requests for printer, plotter, etc.;
3. Changes in the looping structure of the problem.

The results of this analysis are stored in common block /IFPX0/. Each bit in /IFPX0/ is associated with a key name. These names will appear in some Rigid Format's Card Name Table. Mnemonics with \$ appended are always associated with changes in the Case Control Deck. /IFPX0/ contains one bit for every unique entry in the Card Name Table. /IFPX1/ contains these names in order given by /IFPX0/. Thus, bit 135 in /IFPX0/ corresponds to the key word, LOAD\$. If bit 135 is on (non-zero), the status of LOAD\$ has changed on this restart. XSORT analyzes the bulk data card changes in a similar manner, setting the proper bits in /IFPX0/. IFP applies certain logical rules to combinations of the bits. XGPI then analyzes this information in the following manner. For each bit in /IFPX0/ the BCD equivalent is extracted from /IFPX1/. This mnemonic is searched for a match in the Card Name Table. If a match occurs, the appropriate bit in a master module execution mask is turned on. After all changes have been processed, the master mask is logically multiplied (logical and) with each module execution entry. A non-zero result indicates that this module is to be executed.

All bits in /IFPX0/ are classified as either significant to the solution or as only reflecting output requests. If only output request bits are on, a PMR is indicated. If the restart is a Modified Restart, one further table look-up may be necessary: the resulting DMAP sequence determined from the execution flags of the modules may not have the required input data blocks. (All

input data blocks must first appear as output data blocks). If this should be the case (most often caused by switching rigid formats), the File Name Table is consulted to determine which bits are on and hence which modules should be re-executed to generate the missing data blocks. The resulting DMAP sequence causes the selective execution of only those modules necessary to reflect the data changes and complete the requested solution.

2. DATA BLOCK AND TABLE DESCRIPTIONS

2.1 INTRODUCTION

This section contains descriptions of a) those NASTRAN data blocks which appear in one or more Rigid Formats (Section 2.3)*, b) Executive tables maintained by the NASTRAN Executive System (Section 2.4), and c) Miscellaneous tables used by more than one module (Section 2.5).

Data blocks that appear in Rigid Formats are structural problem oriented and reside on physical files. A file is "allocated" to a data block, and a data block is "assigned" to a file. The Executive Segment File Allocator (XSFA) Module is the "administrative manager" of files for NASTRAN.

Executive Tables have true executive functions in the sense that they are not oriented to a particular problem solution or even to structural analysis in general. They may be core resident or may reside on physical files.

Miscellaneous tables are common blocks which are used by the Executive System and/or a particular class of modules (e.g., /GPTA1/ is used by modules GP1, GP2, GP3, PLTSET and TA1). Common blocks that are used for intra-module communications are documented in Section 4, Module Functional Descriptions.

Section 2.2.1 contains an index for data block descriptions sorted on data block names, and Section 2.2.2 contains an index for data block descriptions sorted on the names of modules from which they are output. Alphabetical indexes are given for Executive table descriptions and miscellaneous table descriptions at the beginning of Sections 2.4 and 2.5 respectively.

*The names as listed in Section 2.3 are for the displacement approach rigid formats. For heat approach rigid formats, the names are preceded with the letter H.

2.2 DATA BLOCK DESCRIPTIONS - GENERAL COMMENTS AND INDEXES

Data block descriptions have been organized so that all data blocks output from the same module are grouped together. The name of each data block is given, and the data block is classified as a matrix or a table. A data block is classified as a matrix only if it is in NASTRAN matrix form, that is, generated by one of the NASTRAN matrix packing routines, PACK or BLDPK, the latter having secondary entry points BLDPKI, ZBLPKI and BLDPKN. All other data blocks are classified as tables.

Following a data block's name and classification is a brief description of its contents, followed by its format if it is a table. Since all matrices are in standard NASTRAN packed format a repeated description of the format is unnecessary for matrices. Each data block has a header record (consisting in general of two BCD words) which is the alphanumeric name of the data block as it appears in a Rigid Format, and this header record is designated "Record 0" in table formats. For those few data block which contain more than these two BCD words in their header record, e.g., SLT, GPTT, DLT, the contents are described explicitly. The conventions used for describing the types of words in records of tables are: R implies real; I implies integer; B implies BCD, four characters per computer word left adjusted with the remaining characters, if any, filled with BCD blanks; and L implies a "logical" -- not in the FORTRAN sense -- word which is a mask of bits, right adjusted.

There is associated with each data block a six word control block called a trailer. Trailer information is "written" by the module which outputs the corresponding data block and can be "read" by any module accessing the corresponding data block as input. If a module "writes" a zero trailer for a data block, this implies no data was written in the data block. If a module "writes" a non-zero trailer, this implies data was written in the data block. Non-zero trailer information is often used by modules to allocate core storage before reading the corresponding data block. Trailer information for each data block is stored in and retrieved from the FIAT Executive table (see section 2.4.1.2) by the utility routines WRTTRL (write trailer) and RDTRL (read trailer), which are described in section 3.4.16. While residing in the FIAT, a trailer is stored in 6 half-words; each half-word consists of 16 binary digits.

Trailer information is standardized for matrix data blocks, not standardized for table data blocks. The format of a matrix trailer is as follows:

DATA BLOCK AND TABLE DESCRIPTIONS

Word 1 = number of columns = N

Word 2 = number of rows = M

Word 3 = form = 1 square matrix

2 rectangular matrix

3 diagonal matrix $\begin{cases} N = 1 \\ M = \text{number of rows} \end{cases}$

4 lower triangular matrix

5 upper triangular matrix

6 symmetric matrix

7 row vector $\begin{cases} N = 1 \\ M = \text{number of rows} \end{cases}$

8 identity matrix

Word 4 = type = 1 elements of the matrix are real single precision

2 elements of the matrix are real double precision

3 elements of the matrix are complex single precision

4 elements of the matrix are complex double precision.

Word 5 = maximum number of non-zero words (rather than non-zero matrix elements) in any one column (e.g., if a real double precision matrix is diagonal and non-zero word 5 = 2)

Word 6 = (density of matrix) $\times 10^4$

Word 5 is dependent upon the structural model and the user's grid point sequencing rather than on any intrinsic property of the matrix and is therefore not described in this report.

The lower case letters, e.g., g, n, m, s, o, l, etc., used as subscripts designate the subsets of displacement to which the root symbol (e.g., [K], for a stiffness matrix) applies. The reader is referred to section 3 of the Theoretical Manual and to section 1.7 of the Programmer's Manual for further details.

DATA BLOCK DESCRIPTIONS - GENERAL COMMENTS AND INDEXES

2.2.1 Index for Data Block Descriptions Sorted on Data Block Names

<u>Section Number</u>	<u>Data Block Name</u>	<u>Output from Module</u>	<u>Page Number</u>
2.3.40.6	ABFL	MTRXIN	2.3-178
2.3.62.9	ACPT	APD	2.3-250
2.3.62.8	AERØ	APD	2.3-250
2.3.64.1	AJJL	AMG	2.3-254
2.3.47.2	AUTØ	RANDØM	2.3-223
2.3.2.11	AXIC	IFP	2.3-38
2.3.18.9	BAA	SMP1	2.3-94
2.3.41.2	BDD	GKAD	2.3-179
2.3.70.6	BDICT	EMG	2.3-267
2.3.54.1	BDPØØL	BMG	2.3-239
2.3.70.5	BELM	EMG	2.3-267
2.3.17.8	BFF	SCE1	2.3-89
2.3.69.1	BGG	EMA	2.3-264
2.3.10.2	BGG	SMA2	2.3-76
2.3.77.4	BGP	PLTMRG	2.3-282
2.3.62.7	BGPA	APD	2.3-249
2.3.76.6	BGPDΤ	SGEN	2.3-279
2.3.3.5	BGPDΤ	GP1	2.3-44
2.3.55.2	BGPDP	PLTTRAN	2.3-239
2.3.49.2	BHH	GKAM	2.3-225
2.3.16.5	BNN	MCE2	2.3-86
2.3.27.7	BQG	SDR1	2.3-112
2.3.66.3	BXHH	FA1	2.3-259
2.3.41.8	B2DD	GKAD	2.3-181
2.3.40.3	B2PP	MTRXIN	2.3-177
2.3.76.2	CASEC	SGEN	2.3-278
2.3.1.1	CASECC	IFP1	2.3-1

DATA BLOCK AND TABLE DESCRIPTIONS

<u>Section Number</u>	<u>Data Block Name</u>	<u>Output from Module</u>	<u>Page Number</u>
2.3.77.5	CASEP	PLTMRG	2.3-283
2.3.76.1	CASES	SGEN	2.3.278
2.3.71.1	CASESS	ASDMAP	2.3-269
2.3.39.1	CASEXX	CASE	2.3-176
2.3.67.3	CASEYY	FA2	2.3-260
2.3.78.3	CASEZZ	MØDACC	2.3-284
2.3.42.2	CLAMA	CEAD	2.3-183
2.3.67.2	CLAMAL	FA2	2.3-260
2.3.78.1	CLAMA1	MØDACC	2.3-284
2.3.50.1	CPHID	DDR1	2.3-226
2.3.78.2	CPHIH1	MØDACC	2.3-284
2.3.27.11	CPHIP	SDR1	2.3-113
2.3.3.4	CSTM	GP1	2.3-43
2.3.76.10	CSTM	SGEN	2.3-280
2.3.62.6	CSTMA	APD	2.3-248
2.3.61.1	CYCD	GPCYC	2.3-244
2.3.27.9	DELTAPG	SDR1	2.3-113
2.3.27.10	DELTAQG	SDR1	2.3-113
2.3.27.8	DELTAUGV	SDR1	2.3-112
2.3.2.7	DIT	IFP	2.3-28
2.3.29.7	DLT	DPD	2.3-150
2.3.21.1	DM	RBMG3	2.3-101
2.3.2.9	DYNAMICS	IFP	2.3-32
2.3.81.1	D1JE	INPUTT2	2.3-288
2.3.81.2	D2JE	INPUTT2	2.3-288
2.3.64.3	D1JK	AMG	2.3-255
2.3.64.4	D2JK	AMG	2.3-255
2.3.8.3	ECPT	TA1	2.3-71
2.3.34.4	ECPTNL	PLA1	2.3-169

DATA BLOCK DESCRIPTIONS - GENERAL COMMENTS AND INDEXES

<u>Section Number</u>	<u>Data Block Name</u>	<u>Output from Module</u>	<u>Page Number</u>
2.3.38.2	ECPTNL1	PLA4	2.3-175
2.3.4.1	ECT	GP2	2.3-46
2.3.62.5	ECTA	APD	2.3-247
2.3.2.8	EDT	IFP	2.3-30
2.3.29.4	EED	DPD	2.3-147
2.3.77.3	ELS	PLTMRG	2.3-282
2.3.5.4	ELSETS	PLTSET	2.3-48
2.3.2.5	EPT	IFP	2.3-23
2.3.62.4	EQAERØ	APD	2.3-247
2.3.29.5	EQDYN	DPD	2.3-149
2.3.77.6	EQEX	PLTNRG	2.3-283
2.3.3.2	EQEXIN	GP1	2.3-41
2.3.76.4	EQEXIN	SGEN	2.3-278
2.3.8.1	EST	TA1	2.3-56
2.3.34.2	ESTL	PLA1	2.3-165
2.3.34.3	ESTNL	PLA1	2.3-166
2.3.37.2	ESTNL1	PLA3	2.3-174
2.3.29.9	FRL	DPD	2.3-153
2.3.62.11	FLIST	APD	2.3-252
2.3.66.1	FSAVE	FA1	2.3-258
2.3.84.4	GCYCB	CYCT1	2.3-293
2.3.84.3	GCYCF	CYCT1	2.3-292
2.3.8.2	GEI	TA1	2.3-70
2.3.2.1	GEØM1	IFP	2.3-7
2.3.2.2	GEØM2	IFP	2.3-9
2.3.2.3	GEØM3	IFP	2.3-16
2.3.2.4	GEØM4	IFP	2.3-19
2.3.15.1	GM	MCE1	2.3-84
2.3.41.4	GMD	GKAD	2.3-180

DATA BLOCK AND TABLE DESCRIPTIONS

<u>Section Number</u>	<u>Data Block Name</u>	<u>Output from Module</u>	<u>Page Number</u>
2.3.18.1	GØ	SMP1	2.3-92
2.3.41.5	GØD	GKAD	2.3-180
2.3.8.4	GPCT	TA1	2.3-71
2.3.76.5	GPDT	SGEN	2.3-279
2.3.3.3	GPDT	GP1	2.3-42
2.3.8.7	GPECT	TA1	2.3-74
2.3.3.1	GPL	GP1	2.3-41
2.3.76.3	GPL	SGEN	2.3-278
2.3.62.1	GPLA	APD	2.3-245
2.3.29.1	GPLD	DPD	2.3-145
2.3.77.2	GPS	PLTMRG	2.3-281
2.3.5.3	GPSETS	PLTSET	2.3-47
2.3.9.3	GPST	SMA1	2.3-74
2.3.69.2	GPST	EMA	2.3-264
2.3.7.2	GPTT	GP3	2.3-54
2.3.76.8	GP3S	SGEN	2.3-279
2.3.76.9	GP4S	SGEN	2.3-280
2.3.63.1	GTKA	GI	2.3-253
2.3.32.3	HBAA	SMP2	2.3-162
2.3.41.10	HBDD	GKAD	2.3-181
2.3.70.10	HBDICT	EMG	2.3-268
2.3.70.9	HBELM	EMG	2.3-268
2.3.17.14	HBFF	SCE1	2.3-91
2.3.69.1	HGG	EMA	2.3-264
2.3.16.8	HBNN	MCE2	2.3-87
2.3.41.15	HB2DD	GKAD	2.3-182
2.3.40.5	HB2PP	MTRXIN	2.3-177
2.3.29.14	HDLT	DPD	2.3-156
2.3.21.2	HDM	RBMG3	2.3-101
2.3.29.17	HEQDYN	DPD	2.3-156

DATA BLOCK DESCRIPTIONS - GENERAL COMMENTS AND INDEXES

<u>Section Number</u>	<u>Data Block Name</u>	<u>Output from Module</u>	<u>Page Number</u>
2.3.3.8	HEQEXIN	GP1	2.3-45
2.3.8.5	HEST	TA1	2.3-72
2.3.18.11	HGØ	SMP1	2.3-94
2.3.41.12	HGØD	GKAD	2.3-182
2.3.8.6	HGPECT	TA1	2.3-72
2.3.18.12	HKAA	SMP1	2.3-95
2.3.41.9	HKDD	GKAD	2.3-181
2.3.70.8	HKDICT	EMG	2.3-267
2.3.70.7	HKELM	EMG	2.3-267
2.3.17.10	HKFF	SCE1	2.3-90
2.3.17.11	HKFS	SCE1	2.3-90
2.3.56.3	HKGG	RMG	2.3-240
2.3.69.1	HKGG	EMA	2.3-264
2.3.69.1	HKGGX	EMA	2.3-264
2.3.19.7	HKLL	RBMG1	2.3-97
2.3.19.8	HKLR	RBMG1	2.3-97
2.3.16.4	HKNN	MCE2	2.3-85
2.3.18.13	HKØØ	SMP1	2.3-95
2.3.19.9	HKRR	RBMG1	2.3-98
2.13.17.12	HKSS	SCE1	2.3-90
2.3.41.13	HK2DD	GKAD	2.3-182
2.3.40.4	HK2PP	MTRXIN	2.3-177
2.3.20.5	HLLL	RBMG2	2.3-100
2.3.18.14	HLØØ	SMP1	2.3-95
2.3.41.14	HM2DD	GKAD	2.3-182
2.3.29.15	HNLFT	DPD	2.3-156
2.3.60.1	HØEFIX	SDRHT	2.3-243
2.3.28.29	HØEF1	SDR2	2.3-143
2.3.45.29	HØEF2	SDR3	2.3-216

DATA BLOCK AND TABLE DESCRIPTIONS

<u>Section Number</u>	<u>Data Block Name</u>	<u>Output from Module</u>	<u>Page Number</u>
2.3.28.38	HØES1	SDR2	2.3-144
2.3.28.35	HØPG1	SDR2	2.3-144
2.3.43.8	HØPNL1	VDR	2.3-194
2.3.45.31	HØPNL2	SDR3	2.3-217
2.3.45.26	HØPP2	SDR3	2.3-216
2.3.28.36	HØQG1	SDR2	2.3-144
2.3.45.27	HØQP2	SDR3	2.3-216
2.3.43.9	HØUDV1	VDR	2.3-194
2.3.45.30	HØUDV2	SDR3	2.3-216
2.3.28.37	HØUGV1	SDR2	2.3-144
2.3.45.28	HØUPV2	SDR3	2.3-216
2.3.57.3	HPDØ	TRLG	2.3-241
2.3.57.4	HPDT	TRLG	2.3-241
2.3.24.9	HPF	SSG2	2.3-106
2.3.23.3	HPG	SSG1	2.3-103
2.3.27.18	HPGG	SDR1	2.3-115
2.3.24.8	HPL	SSG2	2.3-105
2.3.58.2	HPNLD	TRHT	2.3-242
2.3.24.6	HPØ	SSG2	2.3-105
2.3.57.1	HPPØ	TRLG	2.3-240
2.3.24.7	HPS	SSG2	2.3-105
2.3.57.2	HPSØ	TRLG	2.3-240
2.3.28.39	HPUGV1	SDR2	2.3-144
2.3.27.20	HQG	SDR1	2.3-115
2.3.59.2	HQG	SSGHT	2.3-243
2.3.56.2	HQGE	RMG	2.3-240
2.3.27.19	HQP	SDR1	2.3-115
2.3.24.5	HQR	SSG2	2.3-105
2.3.32.2	HRAA	SMP2	2.3-162

DATA BLOCK DESCRIPTIONS - GENERAL COMMENTS AND INDEXES

<u>Section Number</u>	<u>Data Block Name</u>	<u>Output from Module</u>	<u>Page Number</u>
2.3.41.11	HRDD	GKAD	2.3-181
2.3.17.13	HRFF	SCE1	2.3-91
2.3.56.1	HRGG	RMG	2.3-240
2.3.16.7	HRNN	MCE2	2.3-87
2.3.25.9	HRULV	SSG3	2.3-109
2.3.59.3	HRULV	SSGHT	2.3-243
2.3.25.10	HRUØV	SSG3	2.3-109
2.3.3.7	HSIL	GP1	2.3-45
2.3.29.12	HSILD	DPD	2.3-156
2.3.55.3	HSIP	PLTTRAN	2.3-239
2.3.7.3	HSLT	GP3	2.3-55
2.3.29.16	HTRL	DPD	2.3-156
2.3.58.1	HUDVT	TRHT	2.3-242
2.3.27.16	HUGV	SDR1	2.3-114
2.3.59.1	HUGV	SSGHT	2.3-242
2.3.25.7	HULV	SSG3	2.3-108
2.3.25.8	HUØØV	SSG3	2.3-108
2.3.27.17	HUPV	SDR1	2.3-115
2.3.13.4	HUSET	GP4	2.3-82
2.3.29.13	HUSETD	DPD	2.3-156
2.3.46.6	HXYPLTT.	XYTRAN	2.3-221
2.3.28.34	IEF1	SDR2	2.3-144
2.3.45.22	IEF2	SDR3	2.3-215
2.3.28.33	IES1	SDR2	2.3-144
2.3.45.21	IES2	SDR3	2.3-215
2.3.75.3	INX	REDUCE	2.3-277
2.3.28.32	IPHIP1	SDR2	2.3-143
2.3.45.20	IPHIP2	SDR3	2.3-215
2.3.28.31	IQP1	SDR2	2.3-143

DATA BLOCK AND TABLE DESCRIPTIONS

<u>Section Number</u>	<u>Data Block Name</u>	<u>Output from Module</u>	<u>Page Number</u>
2.3.45.19	IQP2	SDR3	2.3-215
2.3.45.25	IQP2	SDR3	2.3-216
2.3.18.2	KAA	SMP1	2.3-92
2.3.40.7	KBFL	MTRXIN	2.3-178
2.3.33.2	KBFS	DSMG2	2.3-163
2.3.33.1	KBLL	DSMG2	2.3-163
2.3.33.3	KBSS	DSMG2	2.3-163
2.3.32.1	KDAA	SMP2	2.3-162
2.3.35.3	KDAAM	ADD	2.3-172
2.3.41.1	KDD	GKAD	2.3-179
2.3.17.5	KDFF	SCE1	2.3-89
2.3.17.6	KDFS	SCE1	2.3-89
2.3.31.1	KDGG	DSMG1	2.3-161
2.3.70.2	KDICT	EMG	2.3-266
2.3.16.3	KDNN	MCE2	2.3-85
2.3.17.7	KDSS	SCE1	2.3-89
2.3.70.1	KELM	EMG	2.3-265
2.3.17.1	KFF	SCE1	2.3-88
2.3.17.2	KFS	SCE1	2.3-88
2.3.12.1	KGG	SMA3	2.3-78
2.3.69.1	KGG	EMA	2.3-264
2.3.12.2	KGGL	SMA3	2.3-78
2.3.38.1	KGGNL	PLA4	2.3-175
2.3.35.1	KGGSUM	ADD	2.3-172
2.3.9.1	KGGX	SMA1	2.3-74
2.3.69.1	KGGX	EMA	2.3-264
2.3.34.1	KGGXL	PLA1	2.3-165
2.3.49.3	KHH	GKAM	2.3-225
2.3.85.1	KKK	CYCT2	2.3-293

DATA BLOCK DESCRIPTIONS - GENERAL COMMENTS AND INDEXES

<u>Section Number</u>	<u>Data Block Name</u>	<u>Output from Module</u>	<u>Page Number</u>
2.3.19.1	KLL	RBMG1	2.3-96
2.3.19.2	KLR	RBMG1	2.3-96
2.3.16.1	KNN	MCE2	2.3-85
2.3.18.3	KØØB	SMP1	2.3-92
2.3.19.3	KRR	RBMG1	2.3-96
2.3.17.3	KSS	SCE1	2.3-88
2.3.66.2	KXHH	FA1	2.3-258
2.3.41.6	K2DD	GKAD	2.3-180
2.3.40.9	K2DPP	MTRXIN	2.3-178
2.3.40.1	K2PP	MTRXIN	2.3-177
2.3.18.10	K4AA	SMP1	2.3-94
2.3.17.9	K4FF	SCE1	2.3-90
2.3.69.1	K4GG	EMA	2.3-264
2.3.9.2	K4GG	SMA1	2.3-74
2.3.16.6	K4NN	MCE2	2.3-86
2.3.74.7	LAMA	RCOVR3	2.3-275
2.3.30.1	LAMA	READ	2.3-157
2.3.85.7	LAMA	CYCT2	2.3-294
2.3.20.3	LBLL	RBMG2	2.3-99
2.3.20.1	LLL	RBMG2	2.3-99
2.3.18.4	LØØ	SMP1	2.3-92
2.3.18.6	MAA	SMP1	2.3-93
2.3.72.1	MATC	CØMB2	2.3-271
2.3.2.10	MATPØØL	IFP	2.3-35
2.3.41.3	MDD	GKAD	2.3-179
2.3.70.4	MDICT	EMG	2.3-267
2.3.70.3	MELM	EMG	2.3-267
2.3.5.5	MESSAGE	PLTSET	2.3-49
2.3.17.4	MFF	SCE1	2.3-88

DATA BLOCK AND TABLE DESCRIPTIONS

<u>Section Number</u>	<u>Data Block Name</u>	<u>Output from Module</u>	<u>Page Number</u>
2.3.69.1	MGG	EMA	2.3-264
2.3.10.1	MGG	SMA2	2.3-76
2.3.49.1	MHH	GKAM	2.3-225
2.3.30.3	MI	READ	2.3-158
2.3.85.5	MKK	CYCT2	2.3-294
2.3.19.4	MLL	RBMG1	2.3-96
2.3.19.5	MLR	RBMG1	2.3-97
2.3.16.2	MNN	MCE2	2.3-85
2.3.18.8	MØA	SMP1	2.3-94
2.3.18.7	MØØ	SMP1	2.3-93
2.3.2.6	MPT	IFP	2.3-26
2.3.22.1	MR	RBMG4	2.3-102
2.3.19.6	MRR	RBMG1	2.3-97
2.3.66.4	MXHH	FA1	2.3-259
2.3.41.7	M2DD	GKAD	2.3-180
2.3.40.8	M2DPP	MTRXIN	2.3-178
2.3.40.2	M2PP	MTRXIN	2.3-177
2.3.29.10	NLFT	DPD	2.3-153
2.3.28.21	ØBEF1	SDR2	2.3-136
2.3.28.17	ØBES1	SDR2	2.3-132
2.3.28.10	ØBQG1	SDR2	2.3-125
2.3.42.3	ØCEIGS	CEAD	2.3-184
2.3.28.14	ØCPHIP	SDR2	2.3-129
2.3.28.20	ØEFB1	SDR2	2.3-135
2.3.28.22	ØEFC1	SDR2	2.3-137
2.3.45.13	ØEFC2	SDR3	2.3-209
2.3.28.19	ØEF1	SDR2	2.3-134
2.3.45.5	ØEF2	SDR3	2.3-201
2.3.30.4	ØEIGS	READ	2.3-158
2.3.28.16	ØESB1	SDR2	2.3-131
2.3.28.18	ØESC1	SDR2	2.3-133
2.3.45.12	ØESC2	SDR3	2.3-208

DATA BLOCK DESCRIPTIONS - GENERAL COMMENTS AND INDEXES

<u>Section Number</u>	<u>Data Block Name</u>	<u>Output from Module</u>	<u>Page Number</u>
2.3.28.15	ØES1	SDR2	2.3-130
2.3.28.40	ØES1A	SDR2	2.3-144a
2.3.82.4	ØES1AG	CURV	2.3-289c
2.3.82.3	ØES1AM	CURV	2.3-289b
2.3.82.2	ØES1G	CURV	2.3-289a
2.3.82.1	ØES1M	CURV	2.3-289
2.3.45.4	ØES2	SDR3	2.3-200
2.3.83.2	ØGPF1	GPFDR	2.3-291
2.3.14.1	ØGPST	GPSP	2.3-83
2.3.11.1	ØGPWG	GPWG	2.3-77
2.3.37.1	ØNLES	PLA3	2.3-174
2.3.83.1	ØNRGY1	GPFDR	2.3-290
2.3.73.3	ØPG1	RCØVR	2.3-272
2.3.28.5	ØPG1	SDR2	2.3-120
2.3.45.17	ØPG2	SDR3	2.3-213
2.3.43.1	ØPHID	VDR	2.3-186
2.3.28.13	ØPHIG	SDR2	2.3-128
2.3.73.1	ØPHIG	RCØVR	2.3-272
2.3.43.5	ØPHIH	VDR	2.3-190
2.3.43.4	ØPNL1	VDR	2.3-189
2.3.45.6	ØPNL2	SDR3	2.3-202
2.3.45.24	ØPPB	SDR3	2.3-216
2.3.28.30	ØPPCA	SDR2	2.3-143
2.3.45.23	ØPPCB	SDR3	2.3-215
2.3.28.7	ØPPC1	SDR2	2.3-122
2.3.45.9	ØPPC2	SDR3	2.3-205
2.3.28.6	ØPP1	SDR2	2.3-121
2.3.45.1	ØPP2	SDR3	2.3-197
2.3.68.1	ØPTP1	ØPTPR1	2.3-262
2.3.80.1	ØPTP2	ØPTPR2	2.3-287
2.3.28.9	ØQBG1	SDR2	2.3-124
2.3.73.4	ØQG1	RCØVR	2.3-272

DATA BLOCK AND TABLE DESCRIPTIONS

<u>Section Number</u>	<u>Data Block Name</u>	<u>Output from Module</u>	<u>Page Number</u>
2.3.28.8	ØQG1	SDR2	2.3-123
2.3.45.18	ØQG2	SDR3	2.3-214
2.3.28.12	ØQPC1	SDR2	2.3-127
2.3.45.10	ØQPC2	SDR3	2.3-206
2.3.28.11	ØQP1	SDR2	2.3-126
2.3.45.2	ØQP2	SDR3	2.3-198
2.3.28.2	ØUBGV1	SDR2	2.3-117
2.3.43.2	ØUDVC1	VDR	2.3-187
2.3.45.14	ØUDVC2	SDR3	2.3-210
2.3.43.3	ØUDV1	VDR	2.3-188
2.3.45.7	ØUDV2	SDR3	2.3-203
2.3.73.2	ØUGV1	RCOVR	2.3-272
2.3.28.1	ØUGV1	SDR2	2.3-116
2.3.45.16	ØUGV2	SDR3	2.3-212
2.3.43.6	ØUHVC1	VDR	2.3-191
2.3.45.15	ØUHVC2	SDR3	2.3-211
2.3.43.7	ØUHV1	VDR	2.3-193
2.3.45.8	ØUHV2	SDR3	2.3-204
2.3.28.4	ØUPVC1	SDR2	2.3-119
2.3.45.11	ØUPVC2	SDR3	2.3-207
2.3.28.3	ØUPV1	SDR2	2.3-118
2.3.45.3	ØUPV2	SDR3	2.3-199
2.3.67.4	ØVG	FA2	2.3-261
2.3.53.2	PAF	DDR2	2.3-237
2.3.53.5	PAT	DDR2	2.3-238
2.3.33.4	PBL	DSMG2	2.3-164
2.3.33.5	PBS	DSMG2	2.3-164
2.3.1.2	PCDB	IFP1	2.3-3
2.3.28.28	PCPHIP	SDR2	2.3-143

DATA BLOCK DESCRIPTIONS - GENERAL COMMENTS AND INDEXES

<u>Section Number</u>	<u>Data Block Name</u>	<u>Output from Module</u>	<u>Page Number</u>
2.3.57.4	PD	TRLG	2.3-241
2.3.44.3	PDF	FRRD	2.3-195
2.3.57.3	PDT	TRLG	2.3-241
2.3.23.1	PG	SSG1	2.3-103
2.3.35.2	PG	ADD	2.3-172
2.3.27.2	PGG	SDR1	2.3-111
2.3.74.3	PGS	RCØVR3	2.3-274
2.3.23.2	PG1	SSG1	2.3-103
2.3.36.2	PGV1	PLA2	2.3-173
2.3.57.5	PH	TRLG	2.3-241
2.3.30.2	PHIA	READ	2.3-157
2.3.42.1	PHID	CEAD	2.3-183
2.3.49.4	PHIDH	GKAM	2.3-225
2.3.85.6	PHIA	CYCT2	2.3-294
2.3.27.4	PHIG	SDR1	2.3-111
2.3.42.4	PHIH	CEAD	2.3-185
2.3.67.1	PHIHL	FA2	2.3-260
2.3.85.2	PK	CYCT2	2.3-293
2.3.24.4	PL	SSG2	2.3-104
2.3.26.1	PLI	SSG4	2.3-110
2.3.6.1	PLØTX1	PLØT	2.3-50
2.3.6.2	PLØTX2	PLØT	2.3-50
2.3.77.1	PLTP	PLTMRG	2.3-281
2.3.5.2	PLTPAR	PLTSET	2.3-47
2.3.5.1	PLTSETX	PLTSET	2.3-47
2.3.24.2	PØ	SSG2	2.3-104
2.3.26.2	PØI	SSG4	2.3-110
2.3.74.5	PØS	RCØVR3	2.3-275
2.3.44.4	PPF	FRRD	2.3-195

DATA BLOCK AND TABLE DESCRIPTIONS

<u>Section Number</u>	<u>Data Block Name</u>	<u>Output from Module</u>	<u>Page Number</u>
2.3.28.25	PPHIG	SDR2	2.3-140
2.3.57.1	PPT	TRLG	2.3-240
2.3.24.3	PS	SSG2	2.3-104
2.3.47.1	PSDF	RANDØM	2.3-222
2.3.29.8	PSDL	DPD	2.3-152
2.3.44.2	PSF	FRRD	2.3-195
2.3.74.4	PSS	RCØVR3	2.3-274
2.3.57.2	PST	TRLG	2.3-240
2.3.28.24	PUBGV1	SDR2	2.3-139
2.3.28.26	PUGV	SDR2	2.3-141
2.3.28.23	PUGV1	SDR2	2.3-138
2.3.28.27	PUPVC1	SDR2	2.3-142
2.3.75.1	PVX	REDUCE	2.3-276
2.3.84.1	PX	CYCT1	2.3-292
2.3.74.2	QAS	RCØVR3	2.3-274
2.3.27.6	QBG	SDR1	2.3-112
2.3.27.3	QG	SDR1	2.3-111
2.3.36.3	QG1	PLA2	2.3-173
2.3.65.1	QHHL	AMP	2.3-256
2.3.65.2	QJHL	AMP	2.3-256
2.3.27.15	QP	SDR1	2.3-114
2.3.27.12	QPC	SDR1	2.3-113
2.3.24.1	QR	SSG2	2.3-104
2.3.13.1	RG	GP4	2.3-79
2.3.25.6	RUBLV	SSG3	2.3-108
2.3.25.3	RULV	SSG3	2.3-107
2.3.25.4	RUØV	SSG3	2.3-107
2.3.85.4	RUXV	CYCT2	2.3-294
2.3.76.7	SIL	SGEN	2.3-279
2.3.3.6	SIL	GP1	2.3-45

DATA BLOCK DESCRIPTIONS - GENERAL COMMENTS AND INDEXES

<u>Section Number</u>	<u>Data Block Name</u>	<u>Output from Module</u>	<u>Page Number</u>
2.3.62.2	SILA	APD	2.3-245
2.3.29.2	SILD	DPD	2.3-145
2.3.62.12	SILGA	APD	2.3-252
2.3.55.1	SIP	PLTTRAN	2.3-239
2.3.64.2	SKJ	AMG	2.3-254
2.3.7.1	SLT	GP3	2.3-51
2.3.62.10	SPLINE	APD	2.3-251
2.3.29.6	TFPØØL	DPD	2.3-150
2.3.57.6	TØL	TRLG	2.3-241
2.3.29.11	TRL	DPD	2.3-155
2.3.74.1	UAS	RCØVR3	2.3-274
2.3.27.5	UBGV	SDR1	2.3-112
2.3.20.4	UBLL	RBMG2	2.3-100
2.3.25.5	UBLV	SSG3	2.3-108
2.3.33.7	UBØØV	DSMG2	2.3-164
2.3.44.1	UDVF	FRRD	2.3-195
2.3.50.2	UDV1F	DDR1	2.3-226
2.3.53.3	UDV2F	DDR2	2.3-237
2.3.48.1	UDVT	TRD	2.3-224
2.3.50.3	UDV1T	DDR1	2.3-226
2.3.53.6	UDV2T	DDR2	2.3-238
2.3.53.1	UEVF	DDR2	2.3-237
2.3.53.4	UEVT	DDR2	2.3-237
2.3.27.1	UGV	SDR1	2.3-111
2.3.36.1	UGV1	PLA2	2.3-173
2.3.44.5	UHVf	FRRD	2.3-196
2.3.48.2	UHVT	TRD	2.3-224
2.3.20.2	ULL	RBMG2	2.3-99
2.3.84.2	ULV	CYCT1	2.3-292
2.3.25.1	ULV	SSG3	2.3-107

DATA BLOCK AND TABLE DESCRIPTIONS

<u>Section Number</u>	<u>Data Block Name</u>	<u>Output from Module</u>	<u>Page Number</u>
2.3.27.14	UPV	SDR1	2.3-114
2.3.27.13	UPVC	SDR1	2.3-114
2.3.18.5	UØØ	SMP1	2.3-93
2.3.25.2	UØØV	SSG3	2.3-107
2.3.13.3	USET	GP4	2.3-80
2.3.62.3	USETA	APD	2.3-246
2.3.29.3	USETD	DPD	2.3-146
2.3.75.2	USX	REDUCE	2.3-276
2.3.85.3	UXV	CYCT2	2.3-293
2.3.82.1	VSET	PVEC05,10,20	2.3-289
2.3.69.1	XGG	EMA	2.3-264
2.3.1.3	XYCDB	IFP1	2.3-4
2.3.46.3	XYPLTF	XYTRAN	2.3-221
2.3.46.2	XYPLTFA	XYTRAN	2.3-221
2.3.46.4	XYPLTR	XYTRAN	2.3-221
2.3.46.1	XYPLTT	XYTRAN	2.3-218
2.3.46.5	XYPLTTA	XYTRAN	2.3-221
2.3.33.6	YBS	DSMG2	2.3-164
2.3.13.2	YS	GP4	2.3-80
2.3.74.6	YSS	RCØVR3	2.3-275
2.3.79.10	ZEF2	DDRMM	2.3-286
2.3.79.11	ZEFC1	DDRMM	2.3-286
2.3.79.12	ZEFC2	DDRMM	2.3-286
2.3.79.7	ZES2	DDRMM	2.3-285
2.3.79.8	ZESC1	DDRMM	2.3-286
2.3.79.9	ZESC2	DDRMM	2.3-286
2.3.79.4	ZQP2	DDRMM	2.3-285
2.3.79.5	ZQPC1	DDRMM	2.3-285
2.3.79.6	ZQPC2	DDRMM	2.3-285

DATA BLOCK DESCRIPTIONS - GENERAL COMMENTS AND INDEXES

<u>Section Number</u>	<u>Data Block Name</u>	<u>Output from Module</u>	<u>Page Number</u>
2.3.79.1	ZUPV2	DDRMM	2.3-285
2.3.79.2	ZUPVC1	DDRMM	2.3-285
2.3.79.3	ZUPVC2	DDRMM	2.3-285

DATA BLOCK AND TABLE DESCRIPTIONS

2.2.2 Index for Data Block Descriptions Sorted Alphabetically by Module

<u>Section Number</u>	<u>Module</u>	<u>Page Number</u>	<u>Section Number</u>	<u>Module</u>	<u>Page Number</u>
2.3.35	ADD	2.3-172	2.3.80	ØPTPR2	2.3-287
2.3.64	AMG	2.3-254	2.3.34	PLA1	2.3-165
2.3.65	AMP	2.3-256	2.3.36	PLA2	2.3-173
2.3.62	APD	2.3-245	2.3.37	PLA3	2.3-174
2.3.71	ASDMAP	2.3-269	2.3.38	PLA4	2.3-175
2.3.54	BMG	2.3-239	2.3.6	PLØT	2.3-50
2.3.39	CASE	2.3-176	2.3.77	PLTMRG	2.3-281
2.3.42	CEAD	2.3-183	2.3.5	PLTSET	2.3-47
2.3.72	CØMB2	2.3-271	2.3.55	PLTTRAN	2.3-239
2.3.84	CYCT1	2.3-292	2.3.82	CURV	2.3-289
2.3.85	CYCT2	2.3-293	2.3.47	RANDØM	2.3-222
2.3.79	DDRMM	2.3-285	2.3.19	RBMG1	2.3-96
2.3.50	DDR1	2.3-226	2.3.20	RBMG2	2.3-99
2.3.53	DDR2	2.3-237	2.3.21	RBMG3	2.3-101
2.3.29	DPD	2.3-145	2.3.22	RBMG4	2.3-102
2.3.31	DSMG1	2.3-161	2.3.73	RCØVR	2.3-272
2.3.33	DSMG2	2.3-163	2.3.74	RCØVR3	2.3-274
2.3.69	EMA	2.3-264	2.3.30	READ	2.3-157
2.3.70	EMG	2.3-265	2.3.75	REDUCE	2.3-276
2.3.66	FA1	2.3-258	2.3.56	RMG	2.3-240
2.3.67	FA2	2.3-260	2.3.17	SCE1	2.3-88
2.3.44	FRRD	2.3-195	2.3.60	SDRHT	2.3-243
2.3.63	GI	2.3-253	2.3.27	SDR1	2.3-111
2.3.41	GKAD	2.3-179	2.3.28	SDR2	2.3-116
2.3.49	GKAM	2.3-225	2.3.45	SDR3	2.3-197
2.3.51	GPCYC	2.3-244	2.3.76	SGEN	2.3-278
2.3.83	GPFDR	2.3-290	2.3.9	SMA1	2.3-74
2.3.3	GP1	2.3-41	2.3.10	SMA2	2.3-76
2.3.4	GP2	2.3-46	2.3.12	SMA3	2.3-78
2.3.7	GP3	2.3-51	2.3.18	SMP1	2.3-92
2.3.13	GP4	2.3-79	2.3.32	SMP2	2.3-162
2.3.14	GPSP	2.3-83	2.3.59	SSGHT	2.3-242
2.3.11	GPWG	2.3-77	2.3.23	SSG1	2.3-103
2.3.2	IFP	2.3-5	2.3.24	SSG2	2.3-104
2.3.1	IFP1	2.3-1	2.3.24	SSG3	2.3-107
2.3.81	INPUTT2	2.3-288	2.3.26	SSG4	2.3-110
2.3.15	MCE1	2.3-84	2.3.8	TA1	2.3-56
2.3.16	MCE2	2.3-85	2.3.48	TRD	2.3-224
2.3.78	MØDACC	2.3-284	2.3.58	TRHT	2.3-242
2.3.40	MTRXIN	2.3-177	2.3.57	TRLG	2.3-240
2.3.68	ØPTPR1	2.3-262	2.3.43	VDR	2.3-186
			2.3.46	XYTRAN	2.3-218

DATA BLOCK DESCRIPTIONS

2.3 DATA BLOCK DESCRIPTIONS

2.3.1 Data Blocks Output From Module IFP1

2.3.1.1 CASECC (TABLE)

Description

Case Control Data Table. Values typically zero if not set by the user.

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Subcase number
	2	I	Multipoint constraint set
	3	I	Single-point constraint set
	4	I	External static load set
	5	I	Real eigenvalue extraction set
	6	I	Element deformation set
	7	I	Thermal load set
	8	I	Thermal material set
	9	I	Transient initial conditions
	10	I	Non-linear load output set
	11	I	Output media selection - 1 = print, 2 = plot, 4 = punch, 5 = print and
	12	I	Format of output - 1 = real 2 = real/imag 3 = mag/phase If word 12<0, SORT2 is requested
	13	I	Dynamic load set
	14	I	Frequency response set
	15	I	Transfer function set
	16	I	Symmetry flag
	17	I	{ Same as 10-12 for load output
	18	I	
	19	I	
	20	I	{ Same as 10-12 for displacement output
	21	I	
	22	I	
	23	I	{ Same as 10-12 for element stress output
	24	I	
	25	I	
	26	I	{ Same as 10-12 for element force output
	27	I	
	28	I	
	29	I	{ Same as 10-12 for acceleration output
	30	I	
	31	I	
	32	I	{ Same as 10-12 for velocity output
	33	I	
	34	I	
	35	I	{ Same as 10-12 for forces of constraint output
	36	I	
	37	I	
	38	I	Time step set selection for transient problem

DATA BLOCK AND TABLE DESCRIPTIONS

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
	39	B	32 words of TITLE
	.	B	
	.	B	
	70	B	
	71	B	32 words of SUBTITLE
	.	B	
	.	B	
	102	B	
	103	B	32 words of LABEL
	.	B	
	.	B	
	134	B	
	135	I	Structure plotter flag
	136		Axisymmetric set - COSINE = 2 SINE = 1 FLUID = 2
	137	I	Number of harmonics to output
	138	I	Differential stiffness coefficient set
	139	B	Name of K2PP matrix
	140	B	
	141	B	Name of M2PP matrix
	142	B	
	143	B	Name of B2PP matrix
	144	B	
	145	I	OUTPUT frequency or time set selection
	146		Stress precision check (NCHECK)
	147		Not defined
	148	I	Complex eigenvalue extraction set
	149	I	Structural damping table set
	150	I	Inertia relief set (Force method only)
	151	I	Same as 10-12 for solution set displacements
	152	I	
	153	I	
	154	I	
	155	I	Same as 10-12 for solution set velocities
	156	I	
	157	I	
	158	I	
	159	I	Same as 10-12 for solution set accelerations
	160	I	
	161	I	
	162	I	
	163	I	Non-linear load set in transient problems
	164	I	Delete set (Force method only)
	165	I	Not defined
	166	I	Random analysis set
	167	I	Piecewise linear coefficient set
	168	I	Flutter set
	169	I	Length of Case Control (LCC)
	170	I	Same as 10-12 for grid point force output
	171	I	
	172	I	
	173	I	
	174	I	Same as 10-12 for MPCFØRCE output
	175	I	
	176	I	
	177	I	
	178	I	Same as 10-12 for aerodynamic force output

DATA BLOCK DESCRIPTIONS

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
	179	I	Aerodynamic gust load set
	180	I	} Same as 10-12 for element strain/curvature output
	181	I	
	182	I	
	:		} Not used
	LCC-1		
	LCC	I	Length of symmetry sequence (LSEM)
	LCC+1	R	} Coefficients for symmetry sequence
	LCC+LSEM	R	
	LCC+LSEM+1	I	Set ID
	LCC+LSEM+2	I	Length of the set (LSET)
	LCC+LSEM+3		} Repeated for each set. End of record terminates.
	:		
	:		
	LCC+LSEM+LSET	I	Set members

Note

The above record is repeated for each subcase and symmetry combination.

Table Trailer

Word 1 = number of records on CASECC
Word 2 = 0
Word 3 = maximum length of CASECC
Word 4 = 0
Word 5 = 0
Word 6 = 0

2.3.1.2 PCDB (TABLE)

Description

Plot Control Data Table for the structure plotter.

Table Format

<u>Record</u>	<u>Item</u>
0	Header record
1	The data here is the XRCARD translation of the Structure Plotter.
.	Packed cards in the Case Control Deck (See Subroutine Description for XRCARD). There is one record for each physical card.
.	
.	
N+1	End-of-file

Table Trailer

Words 1 through 3 are zero
Word 4 = 7777
Word 5 and Word 6 are zero

2.3.1.3 XYCDB (TABLE)

Description

XY Output Control Data Block.

Record one contains the subroutine IFP1XY interpretations of the XY plot output request case control data cards. Record two contains an N by 6 matrix stored by rows and sorted such that the columns are in sort left to right. N is the total number of combinations specified by the XY-plot-request case control data cards.

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1-Last	Mixed	IFP1XY interpretations of the XY-output-requests, translated for rapid processing by the XYTRAN module.
2	1	I	Subcase number (0 implies all)
	2	I	Vector request type
	3	I	Point or element ID
	4	I	Component
	5	I	XY output type
	6	I	Destination code

} repeats
for all
rows

Notes

1.

Vector request type =	{	1 = Displacement 2 = Velocity 3 = Acceleration 4 = Single-point forces of constraint 5 = Load or VG 6 = Stress 7 = Force 8 = Adisplacement 9 = Avelocity 10 = Aacceleration 11 = Nonlinear Force
-----------------------	---	--
2.

XY output type =	{	1 = Response 2 = PSDF 3 = AUTØ
------------------	---	--------------------------------------
3.

Destination code =	{	1 = Print 2 = Plot 3 = Print, plot 4 = Punch 5 = Print, punch 6 = Plot, punch 7 = Print, plot, punch
--------------------	---	--
4. Either of records 1 and 2 may be null, however they will always exist.

Table Trailer

Words 1-5 = nonzero
Word 6 = one

DATA BLOCK DESCRIPTIONS

2.3.2 Data Blocks Output From Module IFP.

Module IFP (Input File Processor, part of the NASTRAN Preface) processes the Bulk Data Deck sorted by module XSORT and writes the following data blocks that appear in one or more Rigid Formats: GEOM1, GEOM2, GEOM3, GEOM4, EPT, MPT, DIT, EDT, DYNAMICS, MATPDDL. Each of these data blocks has the usual 2-word BCD header record.

Each data block contains bulk data card images or modified card images of a subset of bulk data card types. Each logical record of each of the above data blocks contains all the data of a particular card type. If a card type is not present in the Bulk Data Deck, there is no record. For each card type present, 3 words are written as header information for the record. Then for every logical bulk data card of that type in the Bulk Data Deck, a card image or a modified card image is written sequentially in the record. Following the last data record, a final three word record is written; the data values are all 65535.

The first two words of the header information of each record are used by entry point LOCATE of subroutine PRELOC. The third word of the header information is the card number used by the IFP programmer to reference tables internal to the IFP module.

LOCATE is used by routines that wish to read data blocks output by the IFP. The second word of the header information portion of each record corresponds to a bit position in a 96-bit, 6-word data block trailer, each word containing 16 bits. If a routine requests LOCATE to locate (position the file to) a particular card type, LOCATE will determine if the card type is present by interrogating the corresponding bit in the trailer, the bit number having been supplied through the calling sequence to LOCATE. If the bit is zero, the card type is not present and LOCATE executes a non-standard RETURN. If the bit is one, the card type is present and LOCATE uses the first word of the header information, also supplied through the calling sequence, to find the proper logical record (card type).

Since the Bulk Data Deck is sorted alphabetically before IFP processes it, and since IFP processes the deck sequentially, the order of the card types in each data block is alphabetical. But it should be noted that in the case of "multi-entry" card types (more than one logical card on one physical card, e.g., CRD, CTUBE, etc.), the card types may or may not be sorted with respect to the primary field (element identification in the above examples). IFP sorts these card types and ensures that they are all fully sorted with respect to the primary field.

DATA BLOCK AND TABLE DESCRIPTIONS

Card type formats correspond to a typical card entry in the logical record allocated to a card type. A number (literal) in a card type format implied that the IFP has placed this number in its output buffer before writing the information on the file and that this number is not on the bulk data card itself. The mnemonics used in the card type formats correspond to the mnemonics in the bulk data card section of the NASTRAN User's Manual. It is advised that anyone using the information on the following pages secure a copy of this section of the User's Manual for cross reference purposes.

IFP also generates the AXIC and FØRCE data blocks, the Parameter Value Table (PVT) and writes Direct Matrix Input (DMI) and Direct Table Input (DTI) cards on the Data Pool File. The AXIC data block, whose presence implies a conical shell (a unique structural element) problem solution, a hydroelastic analysis problem, or an acoustic problem, is processed by the Preface modules IFP3, IFP4 and IFP5. The PVT, which is an Executive Table and is documented in Section 2.4, contains the names and values of all parameters input by means of the PARAM bulk data card. The PVT is written on the Problem Tape. Each DMI in the Bulk Data Deck is output on the Data Pool File in NASTRAN packed matrix format and is indistinguishable from any matrix data block pooled by the XSFA, that is, a matrix trailer is written on the last logical record of the data block. IFP also stores the name of each DMI on the DPL (see Section 2.4). Similarly, each DTI is output on the Data Pool File, a table trailer is written, and the name of the DTI is stored in the DPL.

The preface modules IFP3, IFP4, and IFP5 also will generate some of the data on the data blocks output from IFP. These modules are used to process data cards written by IFP and replace them with equivalent data blocks. For instance, data card CFLUID2 is initially placed on data block AXIC by the IFP module. The IFP3 module will generate CFLUID2 elements and add them to data block GEØM2.

DATA BLOCK DESCRIPTIONS

2.3.2.1 GEØM1 (TABLE)

Card Types and Header Information:

<u>Card Type</u>	<u>Header Word 1</u> <u>Card Type</u>	<u>Header Word 2</u> <u>Trailer Bit Position</u>	<u>Header Word 3</u> <u>Internal Card Number</u>
blank	0	0	89
ADUM1	0	0	3
ADUM2	0	0	32
ADUM3	0	0	51
ADUM4	0	0	88
ADUM5	0	0	99
ADUM6	0	0	100
ADUM7	0	0	101
ADUM8	0	0	103
ADUM9	0	0	106
CØRD1C	1701	17	6
CØRD1R	1801	18	5
CØRD1S	1901	19	7
CØRD2C	2001	20	9
CØRD2R	2101	21	8
CØRD2S	2201	22	10
GRDSET	0	0	2
GRID	4501	45	1
*PØINTAX	4501	45	1
*RINGAX	4501	45	1
*SECTAX	4501	45	1
SEQGP	5301	45	4

* See Section 4.6.2

Card Type Formats:

Blank cards are not written.

ADUMi cards are not written. Rather, the contents are coded and stored in words 46-54 of /SYSTEM/.

CØRD1C (6 words)	CID G1	2 G2	1 G3
CØRD1R (6 words)	CID G1	1 G2	1 G3
CØRD1S (6 words)	CID G1	3 G2	1 G3
CØRD2C (13 words)	CID RID A3 B3 C3	2 A1 B1 C1	2 A2 B2 C2
CØRD2R (13 words)	CID RID A3 B3 C3	1 A1 B1 C1	2 A2 B2 C2

DATA BLOCK AND TABLE DESCRIPTIONS

CORD2S (13 words)	CID	3	2
	RID	A1	A2
	A3	B1	B2
	B3	C1	C2
	C3		

The GRDSET card is not written. Rather, the contents are stored locally for use when processing the GRID cards.

GRID (8 words)	ID	CP	X1
	X2	X3	CD
	PS	F0	

If a GRDSET card is present, and if any of fields 3, 7, or 8 of any GRID card are blank, IFP will insert corresponding data fields from the GRDSET card. Only one GRDSET card may appear in the Bulk Data Deck.

P0INTAX (8 words)	ID	0	PHI
	0.0	0.0	0
	0	0	

See section 4.6.2 for additional information

RINGAX (8 words)	ID	0	R
	Z	0.0	0
	C	0	

See section 4.6.2 for additional information

SECTAX (8 words)	ID	0	R
	PHI1	PHI2	0
	0	0	

See section 4.6.2 for additional information

SEQGP (2 words)	ID	SEQID
-----------------	----	-------

DATA BLOCK DESCRIPTIONS

2.3.2.2 GEØM2 (TABLE)

Card Types and Header Information:

<u>Card Type</u>	<u>Header Word 1 Card Type</u>	<u>Header Word 2 Trailer Bit Position</u>	<u>Header Word 3 Internal Card Number</u>
BARØR	0	0	179
CAXIF2	2108	21	224
CAXIF3	2208	22	225
CAXIF4	2308	23	226
CBAR	2408	24	180
CCØNEAX	2315	23	146
CDAMP1	201	2	69
CDAMP2	301	3	70
CDAMP3	401	4	71
CDAMP4	501	5	72
CDUM1	6108	61	107
CDUM2	6208	62	108
CDUM3	6308	63	109
CDUM4	6408	64	110
CDUM5	6508	65	111
CDUM6	6608	66	112
CDUM7	6708	67	113
CDUM8	6808	68	114
CDUM9	6908	69	115
CELAS1	601	6	73
CELAS2	701	7	74
CELAS3	801	8	75
CELAS4	901	9	76
CFLUID2	7815	78	0*
CFLUID3	7915	79	0*
CFLUID4	8015	80	0*
CHBDY	4208	42	232
CHEXA1	5708	57	219
CHEXA2	5808	58	220
CIHEX1	7108	71	251
CIHEX2	7208	72	252
CIHEX3	7308	73	253
CMASS1	1001	10	65
CMASS2	1101	11	66
CMASS3	1301	12	67
CMASS4	1301	13	68
CNGRNT	5008	50	258
CØNM1	1401	14	63
CØNM2	1501	15	64
CØNRØD	1601	16	47
CQDMEM	2601	26	60
CQDMEM1	2008	20	249
CQDMEM2	5308	53	259
CQDMEM3	5408	54	261
CQDPLT	2701	27	59
CQUAD1	2801	28	57
CQUAD2	2901	29	58
CRØD	3001	30	48
CSHEAR	3101	31	61

*Generated by IFP3, IFP4 or IFP5.

DATA BLOCK AND TABLE DESCRIPTIONS

<u>Card Type</u>	<u>Header Word 1 Card Type</u>	<u>Header Word 2 Trailer Bit Position</u>	<u>Header Word 3 Internal Card Number</u>
CSLØT3	4408	44	227
CSLØT4	4508	45	228
CTETRA	5508	55	217
CTØRDRG	1908	19	104
CTRAPAX	7042	74	287
CTRAPRG	1808	18	80
CTRBSC	3201	32	54
CTRIAAX	7012	70	285
CTRIA1	3301	33	52
CTRIA2	3401	34	53
CTRIARG	1708	17	79
CTRMEM	3501	35	56
CTRPLT	3601	36	55
CTUBE	3701	37	49
CTWIST	3801	38	62
CVISC	3901	39	50
CWEDGE	5608	56	218
GENEL	4301	43	28
PLØTEL	5201	52	11
SPØINT	5551	49	105

DATA BLOCK DESCRIPTIONS

Card Type Formats:

The BARØR card is not written. Rather, the contents are stored locally for use when processing the CBAR cards.

CAXIF2 (6 words)	EID RHØ	G1 B	G2 N
CAXIF3 (7 words)	EID G3 N	G1 RHØ	G2 B
CAXIF4 (8 words)	EID G3 B	G1 G4 N	G2 RHØ
CBAR (16 words)	EID GB X3 PB Z3A Z3B	PID X1 F Z1A Z1B	GA X2 PA Z2A Z2B

If a BARØR card is present in the Bulk Data Deck, or if any of the fields 3, 6, 7, 8, 9 of a CBAR card are blank, IFP will insert the corresponding data fields from the BARØR card. Only one BARØR card may appear in the Bulk Data Deck.

CCONEAX (7 words)	EID RB	PID	RA
CDAMP1 (6 words)	EID G2	PID C1	G1 C2
CDAMP2 (6 words)	EID G2	B C1	G1 C2
CDAMP3 (4 words)	EID S2	PID	S1
CDAMP4 (4 words)	EID S2	B	S1
CDUMi (variable number of words, depending on the contents of the ADUMi card)			
CELAS1 (6 words)	EID G2	PID C1	G1 C2
CELAS2 (8 words)	EID G2 GE	K C1 S	G1 C2
CELAS3 (4 words)	EID S2	PID	S1
CELAS4 (4 words)	EID S2	K	S1
CFLUID2 (6 words)	EID P	S1 B	S2 N
CFLUID3 (7 words)	EID S3 N	S1 P	S2 B

DATA BLOCK AND TABLE DESCRIPTIONS

Card Type Formats Cont'd.:

CFLUID4 (8 words)	EID S3 B	S1 S4 N	S2 p
CHBDY (8 words)	EID AF G3	FLAG G1 G4	H G2
CHEXA1 (10 words)	EID G2 G5 G8	MID G3 G6	G1 G4 G7
CHEXA2 (10 words)	EID G2 G5 G8	MID G3 G6	G1 G4 G7
CIHEX1 (10 words)	EID G2 G5 G8	PID G3 G6	G1 G4 G7
CIHEX2 (22 words)	EID G2 G5 G8 G11 G14 G17 G20	PID G3 G6 G9 G12 G15 G18	G1 G4 G7 G10 G13 G16 G19
CIHEX3 (34 words)	EID G2 G5 G8 G11 G14 G17 G20 G23 G26 G29 G32	PID G3 G6 G9 G12 G15 G18 G21 G24 G27 G30	G1 G4 G7 G10 G13 G16 G19 G22 G25 G28 G31
CMASS1 (6 words)	EID G2	PID C1	G1 C2
CMASS2 (6 words)	EID G2	M C1	G1 C2
CMASS3 (4 words)	EID S2	PID	S1
CMASS4 (4 words)	EID S2	M	S1

DATA BLOCK DESCRIPTIONS

Card Type Formats Cont'd.:

CNGRNT (open ended)	E1 EN	E2 -1	...
CØNM1 (24 words)	EID M11 M31 M41 M44 M53 M61 M64	G M21 M32 M42 M51 M54 M62 M65	CID M22 M33 M43 M52 M55 M63 M66
CØNM2 (13 words)	EID M X3 I22 I33	G X1 I11 I31	CID X2 I21 I32
CØNRØD (8 words)	EID MID C	G1 A NSM	G2 J
CQDMEM (7 words)	EID G2 TH	PID G3	G1 G4
CQDMEM1 (7 words)	EID G2 TH	PID G3	G1 G4
CQDMEM2 (7 words)	EID G2 TH	PID G3	G1 G4
CQDMEM3	Not available.		
CQDPLT (7 words)	EID G2 TH	PID G3	G1 G4
CQUAD1 (7 words)	EID G2 TH	PID G3	G1 G4
CQUAD2 (7 words)	EID G2 TH	PID G3	G1 G4
CRØD (4 words)	EID G2	PID	G1
CSHEAR (6 words)	EID G2	PID G3	G1 G4
CSLØT3 (8 words)	EID G3 M	G1 RHØ N	G2 B
CSLØT4 (9 words)	EID G3 B	G1 G4 M	G2 RHØ N

DATA BLOCK AND TABLE DESCRIPTIONS

Card Type Formats Cont'd.:

CTETRA (6 words)	EID G2	MID G3	G1 G4
CTØRDRG (7 words)	EID G2 0	PID A1	G1 A2
CTRAPAX (7 words)	EID G2 TH	PID G3	G1 G4
CTRAPRG (7 words)	EID G3 MID	G1 G4	G2 TH
CTRBSC (6 words)	EID G2	PID G3	G1 TH
CTRIAAX (6 words)	EID R2	PID R3	R1 TH
CTRIA1 (6 words)	EID G2	PID G3	G1 TH
CTRIA2 (6 words)	EID G2	PID G3	G1 TH
CTRIARG (6 words)	EID G3	G1 TH	G2 MID
CTRMEM (6 words)	EID G2	PID G3	G1 TH
CTRPLT (6 words)	EID G2	PID G3	G1 TH
CTUBE (4 words)	EID G2	PID	G1
CTWIST (6 words)	EID G2	PID G3	G1 G4

DATA BLOCK DESCRIPTIONS

Card Type Formats Cont'd.:

CVISC (4 words)	EID G2	PID	G1
CWEDGE (8 words)	EID G2 G5	MID G3 G6	G1 G4
GENEL (Open Ended)	EID UI2 UIM M UD2 UDN N KZ21 KZM1 ... KZMM S12 S21 S2N SM2	UI1 CI2 CIM UD1 CD2 CDN I KZ31 KZ22 KZM2 N ... S22	CI1 ... -1 CD1 ... -1 KZ11 ... KZ32 ... S11 S1N ... SM1 SMN
PLØTEL (3 words)	EID	G1	G2
SPØINT (1 word)	ID		

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.2.3 GEOM3 (TABLE)

Card Types and Header Information:

<u>Card Type</u>	<u>Header Word 1 Card Type</u>	<u>Header Word 2 Trailer Bit Position</u>	<u>Header Word 3 Internal Card Number</u>
FØRCE	4201	42	18
FØRCEAX	2115	21	146
FØRCE1	4001	40	20
FØRCE2	4101	41	22
GRAV	4401	44	26
LØAD	4551	61	84
MØMAX	3815	38	157
MØMENT	4801	48	19
MØMENT1	4601	46	21
MØMENT2	4701	47	23
PLØAD	5101	51	24
PLØAD1*	6909	69	198
PLØAD2	6802	68	199
PLØAD3	7109	71	255
PRESAX	5215	52	154
QBDY1	4509	45	239
QBDY2	4909	49	240
QHBDY	4309	43	233
QVECT	5001	50	241
QVØL	5209	52	242
RFØRCE	5509	55	190
SLØAD	5401	54	25
TEMP	5701	57	27
TEMPAX	6815	68	155
TEMPD	5641	65	98
TEMPP1	8109	81	201
TEMPP2	8209	82	202
TEMPP3	8309	83	203
TEMPRB	8409	84	204

Card Type Formats:

FØRCE (7 words)	SID F N3	G N1	CID N2
FØRCEAX (7 words)	SID F FZ	RID FR	HID FT

See section 4.6.2 for additional information

FØRCE1 (5 words)	SID G1	G G2	F
FØRCE2 (7 words)	SID G1 G4	G G2	F G3
GRAV (6 words)	SID N1	CID N2	G N3

DATA BLOCK DESCRIPTIONS

Card Type Formats Cont'd.:

LØAD (Open Ended)	SID	S	S1
	L1	S2	L2
	...	S _n	L _n
	-1	-1	
MØMAX (7 words)	SID	RID	O
	F	MR	MT
	M _z		

See section 4.6.2 for additional information

MØMENT (7 words)	SID	G	CID
	M	N1	N2
	N3		
MØMENT1 (5 words)	SID	G	M
	G1	G2	
MØMENT2 (7 words)	SID	G	M
	G1	G2	G3
	G4		
PLØAD (6 words)	SID	P	G1
	G2	G3	G4
PLØAD1* Not available.			
PLØAD2 (3 words)	SID	P	EID
PLØAD3 (5 words)	SID	P	EID
	G1	G2	
PRESAX (7 words)	SID	V	RIDA
	RIDB	PHI1	PHI2
	N		

See section 4.6.2 for additional information

QBDY1 (3 words)	SID	Q	EID
QBDY2 (6 words)	SID	EID	Q1
	Q2	Q3	Q4
QHBDY (8 words)	SID	FLAG	Q0
	AF	G1	G2
	G3	G4	
QVECT (6 words)	SID	Q	E1
	E2	E3	EID
QVØL (Open Ended)	SID	QV	EID1
	EID2	...	EIDN
	-1		
RFØRCE (7 words)	SID	G	CID
	A	N1	N2
	N3		
SLØAD (3 words)	SID	S	F
TEMP (3 words)	SID	G	T

DATA BLOCK AND TABLE DESCRIPTIONS

Card Type Formats Cont'd.:

TEMPAX (3 words)	SID	RID	T
See section 4.6.2 for additional information			
TEMPD (2 words)	SID	T	
TEMPP1 (6 words)	SID T	EID T1	T T2
TEMPP2 (8 words)	SID MX T1	EID MY T2	T MX
TEMPP3 (24 words)	SID T0 Z2 T3 Z5 T6 Z8 T9	EID Z1 T2 Z4 T5 Z7 T8 Z10	Z0 T1 Z3 T4 Z6 T7 Z9 T10
TEMPRB (16 words)	SID TB T'2a TDa TCb TFb	EID T'1a T'2b TEa TDb	TA T'1b TCa TFa TEb

DATA BLOCK DESCRIPTIONS

2.3.2.4 GEOM4 (TABLE)

Card Types and Header Information:

<u>Card Type</u>	<u>Header Word 1 Card Type</u>	<u>Header Word 2 Trailer Bit Position</u>	<u>Header Word 3 Internal Card Number</u>
ASET	5561	76	215
ASET1	5571	77	216
BDYC	910	9	175
BDYS	1210	12	177
BDYS1	1310	13	178
CØNCT	210	2	168
CØNCT1	110	41	167
CRIGDR	8210	82	297
CRIGD1	5310	53	279
CRIGD2	5410	54	284
CRIGD3	8310	83	298
CYJØIN	5210	52	257
GTRAN	1510	15	187
LØADC	500	5	171
MPC	4901	49	17
MPCADD	4891	60	83
MPCAX	4015	40	149
MPCS	1110	11	176
ØMIT	5001	50	15
ØMIT1	4951	63	92
ØMITAX	4315	43	150
PØINTAX	4915	49	152
RELES	410	4	170
RINGAX	5615	56	145
SECTAX	6015	60	153
SPC	5501	55	16
SPC1	5481	58	12
SPCADD	5491	59	13
SPCAX	6215	62	148
SPCD	5110	51	256
SPCS	810	8	174
SPCS1	710	7	173
SPCSD	610	6	172
SUPAX	6415	64	151
SUPØRT	5601	56	14
TRANS	310	3	169

Card Type Formats:

ASET (2 words) ID C

The note below concerning the ØMIT card applies to the ASET card as well.

ASET1 (Open Ended)	C	G	G
	...	G	-1
BDYC (Open Ended)	ID	NAME1	SID1
	NAME2	SID2	...
	...	(blank)	-1

DATA BLOCK AND TABLE DESCRIPTIONS

Card Type Formats Cont'd.:

BDYS (Open Ended)	SID G2 ...	G1 C2 -1	C1 ... -1
BDYS1 (Open Ended)	SID G2	C ...	G1 -1
CØNCT (Open Ended)	SID SUBB GA ...	C GA GB -1	SUBA GB ... -1
CØNCT1 (Open Ended)	NSUB ... G11 C2 G2, NSUB	SID NAME _{NSUB} ... G21 ...	NAME1 C1 G1, NSUB ... -1
CRIGDR (4 words)	EID C1	G	G1
CRIGD1 (Open Ended) and CRIGD2 (Open Ended)	EID G11 G14 G2 G23 G26 GM1 GM4 -1 -1 -1	IG G12 G15 G21 G24 ... GM2 GM5 N -1	G1 G13 G16 G22 G25 GM GM3 GM6 -1 -1
CRIGD3 (Open Ended)	EID IG12 IG15 IG21 IG24 ... IGM1 IGM4 MSET DG12 DG15 DG21 DG24 ... DGN1 DGN4 -1 -1 -1	IG1 IG13 IG16 IG22 IG25 ... IGM2 IGM5 DG1 DG13 DG16 DG22 DG25 ... DGN2 DGN5 -K -1	IG11 IG14 IG2 IG23 IG26 IGM IGM3 IGM6 DG11 DG14 DG2 DG23 DG26 DGN DGN3 DGN6 -1 -1
CYJØIN (Open Ended)	SIDE G2	C ...	G1 -1
GTRAN (4 words)	TID TRAN	NAME	GID

DATA BLOCK DESCRIPTIONS

Card Type Formats Cont'd.:

L0ADC (Open Ended)	SID NAME2 -1	S ID1 ... (blank) -1	NAME1 S1 ... (blank)
MPC (Open Ended)	SID A A C -1	G G ... A -1	C C G -1
MPCADD (Open Ended)	SID ...	S1 S _n	S2 -1
MPCAX (Open Ended)	SID V ... -1	RID ... -1	C ... -1

See Section 4.6.2 for additional information.

MPCS (Open Ended)	SID C G1 -1	NAME A C1 ... (blank) -1	G NAME A1 ... -1
ØMIT (2 words)	ID	C	

Components can be input in any unique combination of digits 1-6. Output format will be ID and one digit, the digits for any one entry being in sort.

Example:	ID 12	C 3516	
Output as:	12 3 12	1 12 6	12 5
ØMIT1 (Open Ended)	C ...	G G	G -1
ØMITAX (2 words)	RID	C	

See Section 4.6.2 for additional information.

PØINTAX (8 words)	ID 0.0 0	0 0.0 0	PHI 0
-------------------	----------------	---------------	----------

See Section 4.6.2 for additional information.

RELES (Open Ended)	SID C1 ... -1	NAME G2 ...	G1 C2 -1
--------------------	------------------------	-------------------	----------------

DATA BLOCK AND TABLE DESCRIPTIONS

Card Type Formats Cont'd.:

RINGAX (8 words)	ID	0	R
	Z	0.0	0
	C	0	

See Section 4.6.2 for additional information.

SECTAX (8 words)	ID	0	R
	PHI1	PHI2	0
	0	0	

See Section 4.6.2 for additional information.

SPC (4 words)	SID	G	C
	D		
SPC1 (Open Ended)	SID	C	G1
	G2	...	G _n
	-1		
SPCADD (Open Ended)	SID	S1	S2
	...	S	-1
SPCAX (4 words)	SID	RID	C
	V		

See Section 4.6.2 for additional information.

SPCD (4 words)	SID	G	C
	D		
SPCS (Open Ended)	SID	NAME	G1
	C1	G2	C2
	-1
	-1		
SPCS1 (Open Ended)	SID	NAME	C
	G1	G2	...
	-1		
SPCSD (8 words)	SID	NAME	G1
	C1	Y1	G2
	C2	Y2	
SUPAX (2 words)	RID	C	

See Section 4.6.2 for additional information.

SUPØRT (2 words)	ID	C
------------------	----	---

The note above concerning the ØMIT card applies to the SUPØRT card as well.

TRANS (10 words)	CID	A1	A2
	A3	B1	B2
	B3	C1	C2
	C3		

DATA BLOCK DESCRIPTIONS

2.3.2.5 EPT (TABLE)

Card Types and Header Information:

<u>Card Type</u>	<u>Header Word 1 Card Type</u>	<u>Header Word 2 Trailer Bit Position</u>	<u>Header Word 3 Internal Card Number</u>
PBAR	52	20	181
PCONEAX	152	19	147
PDAMP	202	2	45
PDUM1	6102	61	116
PDUM2	6202	62	117
PDUM3	6302	63	118
PDUM4	6402	64	159
PDUM5	6502	65	160
PDUM6	6602	66	161
PDUM7	6702	67	163
PDUM8	6802	68	164
PDUM9	6902	69	165
PELAS	302	3	46
PHBDY	2502	25	236
PIHEX	7002	70	254
PMASS	402	4	44
PQDMEM	502	5	41
PQDMEM1	2202	22	250
PQDMEM2	5302	53	260
PQDMEM3	5402	54	262
PQDPLT	602	6	40
PQUAD1	702	7	38
PQUAD2	802	8	39
PROD	902	9	29
PSHEAR	1002	10	42
PTORDRG	2102	21	121
PTRAPAX	7052	95	288
PTRBSC	1102	11	35
PTRIA1	1202	12	33
PTRIA2	1302	13	34
PTRIAAX	7032	85	286
PTRMEM	1402	14	37
PTRPLT	1502	15	36
PTUBE	1602	16	30
PTWIST	1702	17	43
PVISC	1802	18	31
VIEW	2606	26	289

Card Type Formats:

PBAR (19 words)

PID
I1
NSM
C2
E1
F2
I12

MID
I2
FE
D1
E2
K1

A
J
C1
D2
F1
K2

DATA BLOCK AND TABLE DESCRIPTIONS

Card Type Formats Cont'd.:

PCONEAX (24 words)	ID MID2 T2 Z2 PHI3 PHI6 PHI9 PHI12	MID1 I NSM PHI1 PHI4 PHI7 PHI10 PHI13	T1 MID3 Z1 PHI2 PHI5 PHI8 PHI11 PHI14
PDAMP (2 words)	PID	B	
PDUMi (variable number of words, depending on the contents of the ADUMi card).			
PELAS (4 words)	PID S	K	GE
PHBDY (7 words)	PID E R2	MID A	AF R1
PIHEX (7 words)	PID NIP BETA	MID AR	CID ALFA
PMASS (2 words)	PID	M	
PQDMEM (4 words)	PID NSM	MID	T
PQDMEM1 (4 words)	PID NSM	MID	T
PQDMEM2 (4 words)	PID NSM	MID	T
PQDMEM3	Not Available		
PQDPLT (8 words)	PID MID2 Z1	MID1 T Z2	I NSM
PQUAD1 (10 words)	PID MID2 T3 Z2	MID1 I NSM	T1 MID3 Z1
PQUAD2 (4 words)	PID NSM	MID	T
PRØD (6 words)	PID J	MID C	A NSM
PSHEAR	PID NSM	MID	T

DATA BLOCK DESCRIPTIONS

Card Type Formats Cont'd.:

PTØRDRG (4 words)	PID TF	MID	TM
PTRAPAX (17 words)	PID PHI1 PHI4 PHI7 PHI10 PHI13	PHI2 PHI5 PHI8 PHI11 PHI14	MID PHI3 PHI6 PHI9 PHI12
PTRBSC (8 words)	PID MID2 Z1	MID1 T Z2	I NSM
PTRIA1 (10 words)	PID MID2 T3 Z2	MID1 I NSM	T1 MID3 Z1
PTRIA2 (4 words)	PID NSM	MID	T
PTRIAAX (17 words)	PID PHI1 PHI4 PHI7 PHI10	PHI2 PHI5 PHI8 PHI11	MID PHI3 PHI6 PHI9 PHI12
PTRMEM (4 words)	PID NSM	MID	T
PTRPLT (8 words)	PID MID2 Z1	MID1 T2 Z2	I NSM
PTUBE (5 words)	PID T	MID NSM	ØD
PTWIST (4 words)	PID NSM	MID	T
PVISC (3 words)	PID	C1	C2

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.2.6 MPT (TABLE)

Card Types and Header Information:

Card Type	Header Word 1 Card Type	Header Word 2 Trailer Bit Position	Header Word 3 Internal Card Number
DSFACT	53	10	143
MAT1	103	1	77
MAT2	203	2	78
MAT3	1403	14	122
MAT4	2103	21	234
MAT5	2203	22	235
MATS1	503	5	90
MATT1	703	7	91
MATT2	803	8	102
MATT3	1503	15	189
MATT4	2303	23	237
MATT5	2403	24	238
PLFACT	1103	11	185
PLIMIT	304	3	276
PØPT	404	4	277

Card Type Formats:

DSFACT (open ended)	SID	B1	B2
	...	B _n	-1
MAT1 (11 words)	MID	E	G
	NU	RHØ	A
	TREF	GE	ST
	SC	SS	

If any one of E, G or NU is blank, it will be computed to satisfy the identity $E = 2 (1+NU)G$; otherwise, values supplied by the user will be used.

MAT2 (16 words)	MID	G11	G12
	G13	G22	G23
	G33	RHØ	A1
	A2	A12	TØ
	GE	ST	SC
	SS		
MAT3 (16 words)	MID	EX	EY
	EZ	NUXY	NUYZ
	NUZX	RHØ	GXY
	GYZ	GZX	AX
	AY	AZ	TREF
	GE		
MAT4 (3 words)	MID	K	O
MAT5 (8 words)	MID	KXX	KXY
	KXZ	KYY	KYZ
	KZZ	O	
MATS1 (11 words)	MID	R1	R2
	R3	R4	...
	R10		
MATT1 (11 words)	MID	R1	R2
	R3	R4	...
	R10		

DATA BLOCK DESCRIPTIONS

Card Type Formats Cont'd.:

MATT2 (16 words)	MID R3 R15	R1 R4	R2 ...
MATT3 (16 words)	MID R3 R15	R1 R4	R2 ...
MATT4 (3 words)	MID	TK	TC
MATT5 (8 words)	MID T13 T33	T11 T22 TC	T12 T23
PLFACT (Open Ended)	SID ...	B1 B _n	B2 -1
PLIMIT (9 words)	ELTYP (2 words) PID1 PID4	PMIN PID2 PID5	PMAX PID3 0
alternate form:			
	ELTYP (2 words) PID1 0	PMIN THRU (2 words)	PMAX PIDi
PØPT (6 words)	MAX IPRN	EPS PUN1	GAMA PUN2

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.2.7 DIT (TABLE)

Card Types and Header Information:

<u>Card Type</u>	<u>Header Word 1</u> <u>Card Type</u>	<u>Header Word 2</u> <u>Trailer Bit Position</u>	<u>Header Word 3</u> <u>Internal Card Number</u>
GUST	1005	10	308
TABDMP1	15	21	162
TABLED1	1105	11	133
TABLED2	1205	12	134
TABLED3	1305	13	140
TABLED4	1405	14	141
TABLEM1	105	1	93
TABLEM2	205	2	94
TABLEM3	305	3	95
TABLEM4	405	4	96
TABLES1	3105	31	97
TABRND1	55	25	191
TABRNØG	56	26	188

Card Type Formats:

TABDMP1 (open ended)	ID 0 0 g ₁ ... -1	0 0 0 f ₂ f _n -1	0 0 f ₁ g ₂ g _n
TABLED1 (open ended)	ID 0 0 y ₁ ... -1	0 0 0 x ₂ x _n -1	0 0 x ₁ y ₂ y _n
TABLED2 (open ended)	ID 0 0 y ₁ ... -1	X1 0 0 x ₂ x _n -1	0 0 x ₁ y ₂ y _n
TABLED3 (open ended)	ID 0 0 y ₁ ... -1	X1 0 0 x ₂ x _n -1	X2 0 x ₁ y ₂ y _n
TABLED4 (open ended)	ID X3 0 A ₁ A _n	X1 X4 0 A ₂ -1	X2 0 A ₀ ...

DATA BLOCK DESCRIPTIONS

Card Type Formats Cont'd.:

TABLEM1 (open ended)	ID	0	0
	0	0	0
	0	0	x ₁
	y ₁	x ₂	y ₂
	...	x _n	y _n
	-1	-1	
TABLEM2 (open ended)	ID	X1	0
	0	0	0
	0	0	x ₁
	y ₁	x ₂	y ₂
	...	x _n	y _n
	-1	-1	
TABLEM3 (open ended)	ID	X1	X2
	0	0	0
	0	0	x ₁
	y ₁	x ₂	y ₂
	...	x _n	y _n
	-1	-1	
TABLEM4 (open ended)	ID	X1	X2
	X3	X4	0
	0	0	A ₀
	A ₁	A ₂	...
	A _n	-1	
TABLES1 (open ended)	ID	0	0
	0	0	0
	0	0	x ₁
	y ₁	x ₂	y ₂
	...	x _n	y _n
	-1	-1	
TABRND1 (open ended)	ID	0	0
	0	0	0
	0	0	f ₁
	g ₁	f ₂	g ₂
	...	f _n	g _n
	-1	-1	
TABRNDG (10 words)	ID	TYPE	LV
	WG	0	0
	0	0	-1
	-1		
GUST (5 words)	SID	TABLE	WG
	X0	V	

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.2.8 EDT (TABLE)

Card Types and Header Information:

<u>Card Type</u>	<u>Header Word 1 Card Type</u>	<u>Header Word 2 Trailer Bit Position</u>	<u>Header Word 3 Internal Card Number</u>
AEFACT	4002	40	273
AERØ	3202	32	265
CAERØ1	3002	30	263
CAERØ2	4301	43	301
CAERØ3	4401	44	302
CAERØ4	4501	45	303
CAERØ5	5001	50	309
DEFØRM	104	1	81
FLFACT	4102	41	274
FLUTTER	3902	39	272
MKAERØ1	3802	38	271
MKAERØ2	3702	37	270
PAERØ1	3102	31	264
PAERØ2	4601	46	304
PAERØ3	4701	47	305
PAERØ4	4801	48	306
PAERØ5	5101	51	310
SET1	3502	35	268
SET2	3602	36	269
SPLINE1	3302	33	266
SPLINE2	3402	34	267
SPLINE3	4901	49	307
VARIAN	4202	42	290

Card Type Formats:

AEFACT (Open Ended)	SID etc.	F1 -1	F2
AERØ (6 words)	ACSID RHØREF	VSØUND SVMXZ	BREF SYMXX
CAERØ1 (16 words)	PID NCHØRD O Z1 Y4	CP LSPAN X1 X12 Z4	NSPAN LCHØRD Y1 X4 X43
CAERØ2 (16 words)	EID NSB LDNT Y1	PID MINT IGID Z1	CP LSB X1 X12
CAERØ3 (16 words)	EID LISTW ... Y1 X4 X43	PID LISTC1 ... Z1 Y4	LP LISTC2 X1 X12 Z4

DATA BLOCK DESCRIPTIONS

Card Type Formats (Cont.):

CAERØ4 (16 words)	EID NSPAN ... Y1 X4 X43	PID LSPAN ... Z1 Y4	LP ... X1 X12 Z4
CAERØ5 (16 words)	EID NSPAN NTHICK Y1 X4 X43	PID LSPAN X Z1 Y4	CP NTHRY X1 X12 Z4
DEFØRM (3 words)	SID	ID	D
FLFACT (Open Ended)	SID etc.	F1 -1	F2
FLUTTER (9 words)	SID DENS IMETHØD	METHØD MACH blank	blank RFREQ NVALUE
MKAERØ1 (16 words)	M1 M4 M7 K2 or -1 K5 K8	M2 or -1 M5 M8 K3 K6	M3 M6 K1 K4 K7
Note: -1 ends the M or K list			
MKAERØ2 (8 words)	M1 K2 M4	K1 M3 K4	M2 K3
PAERØ1 (8 words)	PID B3 B6	B1 B4 B7	B2 B5
PAERØ2 (15 words)	PID AR LTH1 TMN1 TMI3	ØRIENT LRSB LTH2 TMI2 TMN3	WIDTH LRIB TND1 TMN2 ...
PAERØ3 (8,16,24 words)	PID ... X6	NBØX X5 Y6	NCTRL Y5 ...
PAERØ4 (Open Ended)	PID LIRC CADC1 DØCN -1	CLA LØRC GAPOC1 GABCN	LCLA DØC1 ... GAPOCN
PAERØ5 (Open Ended)	PID NXIS LTAUS CAOCN	NALPHA LXIS CAOC1 -1	LALPHA NTAUS ...
SET1 (Open Ended)	SID etc.	G1 -1	G2

DATA BLOCK DESCRIPTIONS

Card Type Formats (Cont.):

SET2 (8 words)	SID SP2 Z1	EID CH1 Z2	SP1 CH2
SPLINE1 (6 words)	EID BØX2	CAERØ SETG	BØX1 DZ
SPLINE2 (10 words)	EID BØX2 DTØR DTHY	CAERØ SETG CID	BØX1 DZ DTHX
SPLINE3 (Open Ended)	SID CØMP A1 CM	CAERØ G1 ... AM	UFID C1 GM -1
VARIAN (Open Ended)	DBℓ ₂	DBℓ ₂	etc.

DATA BLOCK DESCRIPTIONS

THIS PAGE HAS BEEN LEFT BLANK INTENTIONALLY.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.2.9 DYNAMICS (TABLE)

Card Types and Header Information:

<u>Card Type</u>	<u>Header Word 1 Card Type</u>	<u>Header Word 2 Trailer Bit Position</u>	<u>Header Word 3 Internal Card Number</u>
DAREA	27	17	182
DELAY	37	18	183
DLØAD	57	5	123
DPHASE	77	19	184
EIGB	107	1	86
EIGC	207	2	87
EIGP	257	4	158
EIGR	307	3	85
EPØINT	707	7	124
FREQ	1307	13	126
FREQ1	1007	10	125
FREQ2	1107	11	166
NØLIN1	3107	31	127
NØLIN2	3207	32	128
NØLIN3	3307	33	129
NØLIN4	3407	34	130
RANDPS	2107	21	195
RANDT1	2207	22	196
RANDT2*	2307	23	197
RLØAD1	5107	51	131
RLØAD2	5207	52	132
SEQEP	5707	57	135
TF	6207	62	136
TIC	6607	66	137
TLØAD1	7107	71	138
TLØAD2	7207	72	139
TSTEP	8307	83	142

Card Type Formats:

DAREA (4 words)	SID A	P	C
DELAY (4 words)	SID T	P	C
DLØAD (open ended)	SID L1 ... -1	S S2 S _n -1	S1 L2 L _n
PHASE (4 words)	SID TH	P	C
EIGB (18 words)	SID L2 NDN G 0 0	METHØD (2 words) NEP E C 0	L1 NDP NØRM (2 words) 0 0

DATA BLOCK DESCRIPTIONS

Card Type Formats Cont'd.:

EIGC (open ended)	SID G α_{a1} ω_{b1} N_{d1} α_{b2} N_{e2} α_{an} ω_{bn} N_{dn} -1 -1 -1	METHOD (2 words) C ω_{a1} ℓ_1 α_{a2} ω_{b2} N_{d2} ω_{an} ℓ_n -1 -1 -1	NORM (2 words) E α_{b1} N_{e1} ω_{a2} ℓ_2 ... α_{bn} N_{en} -1 -1 -1
EIGP (4 words)	SID M	α	ω
EIGR (18 words)	SID F2 NZ G O O	METHOD (2 words) NE E C O	F1 ND NORM (2 words) O O
EP0INT (1 word)	ID		
FREQ (open ended)	SID F -1	F ...	F F
FREQ1 (4 words)	SID NDF	F1	DF
FREQ2 (4 words)	SID NF	F1	F2
N0LIN1 (8 words)	SID S T	GI GJ O	CI CJ
N0LIN2 (8 words)	SID S GK	GI GJ CK	CI CJ
N0LIN3 (8 words)	SID S A	GI GJ O	CI CJ
N0LIN4 (8 words)	SID S A	GI GJ O	CI CJ
RANDPS (6 words)	SID X	J Y	K TID

DATA BLOCK AND TABLE DESCRIPTIONS

Card Type Formats Cont'd.:

RANDT1 (4 words)	SID TMAX	N	TO
RANDT2* Not available			
RLØAD1 (6 words)	SID N	L TC	M TD
RLØAD2 (6 words)	SID N	L TB	M TP
SEQEP (2 words)	ID	SEQID	
TF (open ended)	SID BO G(1) A1(1) C(2) A2(2) C(N) A2(N) -1	GD B1 C(1) A2(1) AO(2) ... AO(N) -1 -1	CD B2 AO(1) G(2) A1(2) G(N) A1(N) -1 -1
TIC (5 words)	SID UO	G VO	C
TLØAD1 (5 words)	SID O	L TF	M
TLØAD2 (10 words)	SID O F B	L T1 P	M T2 C
TSTEP (open ended)	SID NØ(1) NØ(2) DT(N) -1	N(1) N(2) ... NØ(N) -1	DT(1) DT(2) N(N) -1

DATA BLOCK DESCRIPTIONS

2.3.2.10 MATPØØL (TABLE)

Card Types and Header Information:

Card Type	Header Word 1 Card Type	Header Word 2 Trailer Bit Position	Header Word 3 Internal Card Number
DMIAX	214	2	221
DMIG	114	1	120
BNDFL	9614	96	0**
RADLST	2014	20	243
RADMTX	3014	30	244

Card Type Formats:

DMIAX (open ended)			NAME (2 words)		0	IFØ	TIN	TØUT	Header Informa- tion for the matrix (9 words)
			0		0	0			
GJ	CJ	NJ	GI	CI	NI	V*	}	Non-zero terms of the first non-zero column	
			GI	CI	NI	V*			
					
					
			GI	CI	NI	V*			
			-1	-1	-1		End of column indicators		
GJ	CJ	NJ	GI	CI	NI	V*	}	Non-zero terms of the second non-zero column	
			GI	CI	NI	V*			
					
					
			GI	CI	NI	V*			
			-1	-1	-1		End of column indicators		
.									
.									
GJ	CJ	NJ	GI	CI	NI	V*	}	Non-zero terms of the last non-zero column	
			GI	CI	NI	V*			
					
					
			GI	CI	NI	V*			
			-1	-1	-1		End of column indicators		
-1	-1	-1					End of matrix indicators		

*V may be 1, 2, or 4 words depending on TIN.

**Generated by IFP3, IFP4 or IFP5.

DATA BLOCK AND TABLE DESCRIPTIONS

Card Type Formats Cont'd.:

DMIG (open ended)	NAME (2 words)	O 0	IFØ 0	TIN	TØUT	Header Information for the matrix (9 words)
GJ	CJ	GI	CI	V*		Non-zero terms of the first non-zero column
		GI	CI	V*		
		.	.	.		
		.	.	.		
		GI	CI	V*		
		-1	-1			End of column indicators
GJ	CJ	GI	CI	V*		Non-zero terms of the second non-zero column
		GI	CI	V*		
		.	.	.		
		.	.	.		
		GI	CI	V*		
		-1	-1			End of column indicators
.						
.						
.						
GJ	CJ	GI	CI	V*		Non-zero terms of the last non-zero column
		GI	CI	V*		
		.	.	.		
		.	.	.		
		GI	CI	V*		
		-1	-1			End of column indicators
-1	-1					

*V may be 1, 2, or 4 words depending on TIN.

BNDFL (open ended)

CS _f	g	ρ	B	NØSYM	Header Information
M	S1	S2	NHARM	N1.....	
Id _{f1}	r	z	ℓ	c	Fluid point data
	s	ρ			
	G ₁	φ ₁			Surface grid point Id's and azimuth angles.
	G ₂	φ ₂			
	G ₃	φ ₃			
	.				
-1	-1	End of data for fluid point			
Id _{f2}	r	z	ℓ	c	Fluid point data
	s	ρ			
	.				
	etc.				
65535	65535	65535	End of Record		

DATA BLOCK DESCRIPTIONS

Card Type Formats Cont'd.:

RADLST (Open Ended)	E1	E2	E3
		-etc.-	
	-1	-1	-1
RADMTX (Open Ended)	I1	F11	F12
	F13	-etc.-	
	-1	-1	
	I2	F22	F23
	F24	-etc.-	
	-1	-1	
		-etc.-	
	-1	-1	-1

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.2.11 AXIC (TABLE)

Card Types and Header Information:

<u>Card Type</u>	<u>Header Word 1 Card Type</u>	<u>Header Word 2 Trailer Bit Position</u>	<u>Header Word 3 Internal Card Number</u>
<u>Conical Shell</u>			
AXIC	515	5	144
CCØNEAX	2315	23	146
FØRCEAX	2115	21	156
MØMAX	3815	38	157
MPCAX	4015	40	149
ØMITAX	4315	43	150
PØINTAX	4915	49	152
PRESAX	5215	52	154
RINGAX	5615	56	145
SECTAX	6015	60	153
SPCAX	6215	62	148
SUPAX	6415	64	151
TEMPAX	6815	68	155
<u>Hydroelastic</u>			
AXIF	8815	88	212
BDYLIST	8915	89	213
CFLUID2	8515	85	209
CFLUID3	8615	86	210
CFLUID4	8715	87	211
FLSYM	9115	91	222
FREET	9015	90	214
FSLIST	8215	82	206
GRIDB	8115	81	205
PRESPT	8415	84	208
RINGFL	8315	83	207
<u>Acoustic Cavity</u>			
AXSLØT	1115	11	223
GRIDF	1215	12	229
GRIDS	1315	13	230
SLBDY	1415	14	231
<u>Axisymmetric Solids</u>			
CTRAPAX	7042	74	287
CTRIAAX	7012	70	285

Card Type Formats:

Conical Shell

AXIC (2 words)	H	0	
CCØNEAX (4 words)	ID RB	PID	RA

DATA BLOCK DESCRIPTIONS

Card Type Formats Cont'd.:

FØRCEAX (8 words)	SID HID2 FP	RID S FZ	HID1 FR
MØMAX (8 words)	SID HID2 MP	RID S MZ	HID1 MR
MPCAX (Open Ended)	SID C HID ... C -1	RID A C RID A -1	HID RID A HID -1 -1
ØMITAX (3 words)	RID	HID	C
PØINTAX (3 words)	ID	RID	PH1
PRESAX (6 words)	SID RID2	P PHI1	RID1 PHI2
RINGAX (4 words)	ID C	R	Z
SECTAX (5 words)	ID PHI1	RID PHI2	R
SPCAX (5 words)	SID C	RID V	HID
SUPAX (3 words)	RID	HID	C
TEMPAX (4 words)	SID TEMP	RID	PHI

Hydroelastic

AXIF (open ended)	CSF B N1	G NØSYM	RHØ NHARM -1
BDYLIST (open ended)	RHØ IDF	IDF	IDF -1
CFLUID2 (5 words)	ID RHØ	IDF B	IDF
CFLUID3 (6 words)	ID IDF	IDF RHØ	IDF B
CFLUID4 (7 words)	ID IDF B	IDF IDF	IDF RHØ
FLSYM (3 words)	M	S1	S2
FREEPT (3 words)	IDF	ID	PHI

DATA BLOCK AND TABLE DESCRIPTIONS

Card Type Formats Cont'd.:

FSLIST (Open Ended)	RHØ IDF	IDF	IDF -1
GRIDB (5 words)	ID PS	PHI IDF	CID
PRESPT (3 words)	IDF	ID	PHI
RINGFL (4 words)	IDF X3	X1	X2

Acoustic Cavity

AXSLØT (5 words)	RHØ W	B M	N
GRIDF (3 words)	IDG	R	Z
GRIDS (5 words)	IDG W	R IDF	Z
SLBDY (Open Ended)	RHØ ID2	M -1	ID1 -1

Axisymmetric Solids

CTRAPAX (7 words)	ID G2 TH	PID G3	G1 G4
CTRIAAX (6 words)	ID G2	PID G3	G1 TH

DATA BLOCK DESCRIPTIONS

2.3.3 Data Blocks Output From Module GP1

2.3.3.1 GPL (TABLE)

Description

Grid Point List.

First logical record contains a list of external grid and scalar numbers in internal sort.
Second logical record contains pairs of external grid and scalar numbers and sequence numbers in internal sort.

Table Format

<u>Record</u>	<u>Word</u>	<u>Item</u>
0		Header record
1	1	External grid or scalar number
		} repeated for each grid or scalar point in model
2	1	External grid or scalar number
	2	Sequence number
		} repeated for each grid or scalar point in model
3		End-of-file

Notes

1. Internal is implied by word position in record one.
2. Sequence number = 1000 * external number unless replaced by a new sequence number on a SEQGP card.
3. All data words are integers.

Table Trailer

Word 1 = number of external grid points + number of scalar points.

Word 2-6 = zero.

2.3.3.2 EQEXIN (TABLE)

Description

Equivalence between external grid or scalar numbers and internal numbers.

First record contains pairs of external grid and scalar numbers and internal numbers in external sort. Second logical record contains pairs of external grid and scalar numbers and coded SIL numbers in external sort.

Table Format

<u>Record</u>	<u>Word</u>	<u>Item</u>
0		Header record
1	1 2	External grid or scalar number Internal number
2	1 2	External grid or scalar number 10*SIL number + code
3		End-of-file

} repeated for each
grid or scalar point
in model

} repeated for each
grid or scalar point
in model

Notes

1. All data words are integers.
2. Code = $\begin{cases} 1 & \text{for grid point} \\ 2 & \text{for scalar point} \end{cases}$

Table Trailer

Word 1 = number of grid points + number of scalar points.

Word 2-6 = zero.

2.3.3.3 GPDT (TABLE)

Description

Grid Point Definition Table.

One logical record contains list of all grid and scalar points with associated coordinate system and constraint information. List is in internal sort.

Table Format

<u>Record</u>	<u>Word</u>	<u>Item</u>
0		Header record
1	1 2 3 4 5 6 7	Internal number Coordinate system ID that defines x, y, z $\begin{pmatrix} x \\ y \\ z \end{pmatrix}$ or $\begin{pmatrix} R \\ \theta \\ z \end{pmatrix}$ or $\begin{pmatrix} \rho \\ \theta \\ \phi \end{pmatrix}$ depending on defining coordinate system Coordinate system ID for displacements Constraint code
2		End-of-file

} repeated for each
grid or scalar

Notes

1. Words 3-5 are single precision floating point; all other words are integer.
2. Scalar points are identified by coordinate system ID = -1, and words 3-7 are all zero.
3. See description of the GRID bulk data card in the User's Manual for a definition of the constraint code.
4. If a single degree of freedom, such as a hydroelastic fluid point, is desired, the integer -1, is used in position 6.

DATA BLOCK DESCRIPTIONS

Table Trailer

Word 1 = number of grid points + number of scalar points.
 Word 2-6 = zero.

2.3.3.4 CSTM (TABLE)

Description

Coordinate System Transformation Matrices.

One logical record contains all coordinate system transformations. Transformation is from global to basic by the following formulation:

(1) rectangular

$$\begin{pmatrix} x \\ y \\ z_B \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ z_g \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix}$$

(2) cylindrical

$$\begin{pmatrix} x \\ y \\ z_B \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \begin{pmatrix} R \cos \theta \\ R \sin \theta \\ z_g \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix}$$

(3) spherical

$$\begin{pmatrix} x \\ y \\ z_B \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \begin{pmatrix} \rho \sin \theta \cos \phi \\ \rho \sin \theta \sin \phi \\ \rho \cos \theta \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix}$$

Table Format

<u>Record</u>	<u>Word</u>	<u>Item</u>
0		Header record
1	1	Coordinate system ID
	2	Coordinate system type
	3-5	t_1, t_2, t_3
	6-14	$r_{11}, r_{12}, r_{13}, r_{21}, r_{22}, r_{23}, r_{31}, r_{32}, r_{33}$
2		End-of-file

$\left. \begin{matrix} 1 = \text{rectangular} \\ 2 = \text{cylindrical} \\ 3 = \text{spherical} \end{matrix} \right\}$ repeated for each coordinate system

Notes

1. Coordinate system ID and coordinate system type are integers.
2. t_i and r_{ij} are single precision floating point.

Table Trailer

Word 1 = number of grid points + number of scalar points.
 Word 2 = number of coordinate systems.
 Word 3-6 = zero.

2.3.3.5 BGPDT (TABLE)

Description

Basic Grid Point Definition Table.

One logical record contains a list of all grid and scalar points in internal sort, with (for grid points) their x, y, z coordinates in the basic system along with a coordinate system ID for displacement computations.

Table Format

<u>Record</u>	<u>Word</u>	<u>Item</u>
0		Header record
1	1 2-4	Coordinate system ID } repeated for each x, y, z in basic system } grid or scalar point
2		End-of-file

Notes

1. Coordinate system ID is integer; x, y, z are single precision, floating point.
2. Scalar points are identified by coordinate system ID = -1, and x, y, z = 0.

Table Trailer

Word 1 = number of grid points + number of scalar points.
 Word 2-6 = zero.

DATA BLOCK DESCRIPTIONS

2.3.3.6 SIL (TABLE)

Description

Scalar Index List.

One logical record that contains a list of SIL numbers for each grid or scalar point. The list is in internal sort, therefore, internal number is implied by word position in the record. Definition of SIL numbers is as follows:

Let i = internal number, then

$$SIL_1 = 1,$$

$$SIL_{i+1} = \begin{cases} SIL_i + 6 & \text{if } i = \text{grid point} \\ SIL_i + 1 & \text{if } i = \text{scalar point} \end{cases}$$

Table Format

<u>Record</u>	<u>Word</u>	<u>Item</u>
0		Header record
1	1	SIL_1
	\vdots	\vdots
	n	SIL_n
2		End-of-file

Notes

SIL numbers are integers.

Table Trailer

Word 1 = number of grid points + number of scalar points.

Word 2 = degrees of freedom in the g-displacement set.

Word 3-6 = zero.

2.3.3.7 HSIL (TABLE)

Description

See description and format of SIL table - section 2.3.3.6.

2.3.3.8 HEQEXIN (TABLE)

Description

See description and format of EQEXIN table - section 2.3.3.2.

2.3.4 Data Blocks Output From Module GP2

2.3.4.1 ECT (TABLE)

Description

Element Connection Table.

The ECT contains one logical record for each element connection card type that has been input. Additionally, the ECT contains one logical record for GENEL elements if they have been input.

Table Format

The ECT is identical in format to data block GEØM2, output from module IFP. All external grid or scalar numbers are replaced by internal numbers. SPØINT data is not copied on the ECT.

Table Trailer

Identical to trailer on GEØM2 data block.

2.3.5 Data Blocks Output From Module PLTSET

2.3.5.1 PLTSETX (TABLE)

Description

User error messages related to the definition of element plot sets for the structure plotter.

Table Format

See the description of the MESSAGE table, section 2.3.5.5.

Note

PLTSETX is generated in subroutine SETINP.

Table Trailer

Word 1-5 = 0
Word 6 = 1

2.3.5.2 PLTPAR (TABLE)

Description

Plot parameters and plot control table.

Table Format

<u>Record</u>	<u>Word</u>	<u>Item</u>
0		Header record
1		Duplicate of the plot control data block (PCDB) created in the IFP1 module except that all plot set definitions have been deleted.
2		
3		
etc.		
Last		End-of-file

Note

PLTPAR is generated in subroutine SETINP.

Table Trailer

Word 1-5 = 0
Word 6 = 1

2.3.5.3 GPSETS (TABLE)

Description

Grid point sets related to the element plot sets.

DATA BLOCK AND TABLE DESCRIPTIONS

Table Format

<u>Record</u>	<u>Word</u>	<u>Item</u>
0		Header record
1	1-NSETS	Element plot set ID's (integer)
2-(NSETS+1)	1 2-(NGP+1)	Number of grid points in an element set Pointers to the grid points in this element set (integers) 1 If = 0, the grid point is not in this set 2 If ≠ 0, this is an internal index relative to only the grid points in this element set (if negative, this grid point is to be excluded when used to draw deformed vectors and applying grid labels or symbols).
NSETS+2		End-of-file

Notes

1. NSETS = number of element sets (second module parameter)
2. NGP = total number of structural grid points (first module parameter)
3. GPSETS is generated in subroutines SETINP and CNSTRC

Table Trailer

Word 1-5 = 0
Word 6 = 1

2.3.5.4 ELSETS (TABLE)

Description

Element plot set connection tables.

Table Format

<u>Record</u>	<u>Group</u>	<u>Word</u>	<u>Item</u>	
0		0	Header Record	
1-NSETS	1-NTYPES	1	Hollerith element symbol (2 BCD characters)	} Repeated for all element types in the set
		2	NGPEL - Number of grid points per element of this type (Integer) 1. NGPEL < 3 - one dimensional element 2. $3 \leq \text{NGPEL} \leq 4$ and symbol not "TE" - the first and last grid are also connected 3. $4 < \text{NGPEL}$ or symbol is "TE" - special line connection pattern in LINEL is used.	
		3.	Element identification number (integer). If zero there are no more elements of this type; a new group or end-of-record follows.	
		4.	Index of this element type in ECT.	} Repeated for all elements of this type
		5-(NGPEL+3)	Grid point connection indices for the GPSETS array of the same record (Integer)	
NSETS+1			End-of-file	

DATA BLOCK DESCRIPTIONS

Notes

1. NSETS = number of element plot sets (second module parameter)
2. NTPS = number of element types represented in an element plot set (created in SETINP and modified in CNSTRC)
3. NGPEL and Hollerith element symbol are from /GPTA1/. Only elements that
 - a) have 2 to 20 grid points
 - b) do not have scalar connections possible
 - c) do not have an element symbol of "XX"
 are plottable. Word 4,5 of /GPTA1/ may be used to request the element by name (e.g., R0D).

Table Trailer

Word 1-5 = 0
Word 6 = 1

2.3.5.5 MESSAGE (TABLE)

Description

Messages to be processed by the message writer module (PRTMSG). Each message may either be a physical or logical record. This data block is never really created as such, but is included so as to explain other data blocks such as PL0TX1, PL0TX2, etc., and is referenced in the Table Formats of these data blocks.

Table Format

<u>Record</u>	<u>Word</u>	<u>Item</u>
0		Header record
A given logical record in a given physical record can be of two alternate forms		
A. <u>Record</u>	<u>Word</u>	<u>Item</u>
i	j	If = -1, -2, -3, -4, -5, or -6, then the next 32 words is a new title for the 1st, 2nd, 3rd, 4th, 5th, or 6th lines on all printed pages to follow from this message table (integer)
	(j+1)-(j+32)	The 32 4-character BCD words for this title
B. <u>Record</u>	<u>Word</u>	<u>Item</u>
i	j	NLIST = number of list items (integer)
	(j+1)-(j+NLIST)	List items (mixed mode)
	j+NLIST+1	NF = size of format to be used to print these list items (integer)
	(j+NLIST+2)-	Format to be used to print this list of consecutive
	(j+NLIST+NF+1)	BCD characters)

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.6 Data Blocks Output From Module PLØT

2.3.6.1 PLØTX1 (TABLE)

Description

User messages from the plot module relative to the undeformed structural shapes

Table Format

See the description of the MESSAGE table, section 2.3.5.5

Table Trailer

Word 1-5 = 0
Word 6 = 1

2.3.6.2 PLØTX2 (TABLE)

Description

User messages from the plot module relative to the deformed structural shanes generated in the statics analysis

Table Format

See the description of the MESSAGE table, section 2.3.5.5

Table Trailer

Word 1-5 = 0
Word 6 = 1

DATA BLOCK DESCRIPTIONS

2.3.7 Data Blocks Output From Module GP3

2.3.7.1 SLT (TABLE)

Description

Static Loads Table.

The header record of the SLT contains a sorted list of all unique load set ID's contained on static load cards except the LØAD card itself. The n logical records that follow the header record comprise all static loads data belonging to each of the n load sets, one logical record per load set. The (n+1)st logical record contains all LØAD cards (if any).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0	1-2	B	Data block name
	3	I	Load set ID ₁
	:		:
	2+n	I	Load set ID _n
1	1	I	Load card type
	2	I	No. of load data entries in the record
	3	}	Load data as function of load card type... repeated m times
	:		
	2+m		
:			} repeated for each different load card type belonging to the same load set ID
n			
			Same format as record 1
			Data belonging to the n-th load set
n+1	1,2	I,R	Combination load Id, Overall scale factor
	3,4	R,I	Scale factor, load set ID
	5,6	R,I	Scale factor, load set ID
	2k+3,2k+4	I	-1, -1
n+2			End-of-file

Notes

1. The SLT is generated in subroutine GP3A.
2. Card type ID's and format of data for each bulk data card type are as follows:

FØRCE card type

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	=1, (FORCE card type ID)
3	I	Internal grid number
4	I	Coordinate system ID
5	R	Signed scale factor for applied force
6-8	R	Force components

DATA BLOCK AND TABLE DESCRIPTIONS

MØMENT card type

See FØRCE, substituting "moment" for "force" and substitute "2" in word 1 for card type ID.

FØRCE1 card type

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	=3, (FØRCE1 card type ID)
3	I	Internal grid number
4	R	Signed magnitude of applied load
5-6	I	Internal grid numbers of grid points that define direction

MØMENT1 card type

See FØRCE1, substituting "moment" for "force" and substituting "4" in word "1" for card type ID.

FØRCE2 card type

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	=5, (FØRCE2 card type ID)
3	I	Internal grid number
4	R	Signed magnitude of applied load
5-8	I	Internal grid numbers of grid points that define direction

MØMENT2 card type

See FORCE2, substituting "moment" for "force" and substituting "6" in word "1" for card type ID.

SLØAD card type

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	=7, (SLOAD card type ID)
3	I	Internal scalar number
4	R	Applied load

GRAV card type

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	=8, (GRAV card type ID)
3	I	Coordinate system ID
4	R	Gravity vector scale factor
5-7	R	Gravity vector components

DATA BLOCK DESCRIPTIONS

PLØAD, PLØAD2, PLØAD3 card types

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	=9, (PLØAD, PLØAD2, and PLØAD3 card type ID)
3	R	Pressure
4-7	I	Internal grid numbers

RFØRCE card type

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	=10, (RFØRCE card type ID)
3	I	Internal grid number
4	I	Coordinate system ID
5	R	Scale factor
6-8	R	Components of rotation direction vector

PRESAX card type

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	=11, (PRESAX card type ID)
3	R	Pressure value
4-5	I	Ring ID's
6-7	R	Azimuthal angles
8	I	Number of harmonics

3. With the exception of GRAV and PLØAD card types, data for a given card type within a logical record is in sort on internal grid (or scalar) number at which the load is applied. Data for a PLØAD type is in sort on external equivalent of the grid ID corresponding to G1. Data for a GRAV type is a single entry record.
4. If no LØAD cards have been input, the (n+1) st record does not exist.

Table Trailer

Word 1 = number of load sets.

Word 2-6 = zero.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.7.2 GPTT (TABLE)

Description

Grid Point Temperature Table.

The header record of the GPTT contains sorted triples of temperature set ID, default temperature, and flag. For each temperature set for which temperature data is defined at the grid points or structural elements, a logical record of the GPTT is present.

Table Format

Record	Word	Item	
0	1-2	Data block name (BCD)	
	3	Temperature set ID, (integer)	
	4	Default temperature (floating point) -1 if no default temperature defined (integer)	} repeated for each temperature set
	5	0 if only default temperature for set (integer) >0 record number of temperature data for set (integer)	
1	1	Temperature set ID	
	2	Element type	
	3	Element type count of temperature data (number of values for element ID)Default = -1	} Repeats for all element types in problem.
	4	Element ID (nonexisting if word 3 = -1)	
	5	} Temperature values (nonexistent if element ID is neg- ative or nonexisting)	} Repeats for all elements of element type in problem.
	.		
	.		
	.		
	5+count-1		
	0	Flag indicating end of element data for element type.	
...			
k			
k+1		Same format as record 1 (Same data as record 1.)	
k+2	1	Temp set ID	
	2	GRID index	} Repeats for N grid point
	3	Temperature	
	-1		
	-1		
k+3		Same format as record k+2	
...			
k+k+1		End-of-file	

DATA BLOCK DESCRIPTIONS

Notes

1. The GPTT is generated in subroutine GP3D.
2. A temperature set may be defined as consisting only of a default temperature that applies to all grid points, and thus elements connecting those grid points.
3. A default temperature (if defined) is to apply to all grid points for which a temperature has not been defined.
4. The second set of records (k+1, etc.) are used only for heat transfer problems.

Table Trailer

Trailer contains no specific information.

2.3.7.3 HSLT (TABLE)

Description

See description and format of SLT table - section 2.3.7.1.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.8 Data Blocks Output from Module TA1

2.3.8.1 EST (TABLE)

Description

Element Summary Table.

The EST is a collection of data for all elements of the structural model. It contains one logical record for each element type. For each element: connection data, properties data, basic grid point data and the element temperature data are grouped. General elements and elements that belong to super elements are not included in the EST.

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>		
0			Header record		
1	1	I	Element type	} repeated for each element	} repeated for each element type
	2-i+1		ECT section		
	i+2-i+j+1		EPT section		
	i+j+2-i+j+k+1		BGPDT section		
	i+j+k+2-i+j+k+m+1		ETT section		
n+1			End-of-file		

Notes

1. i = number of words in ECT section.
2. j = number of words in EPT section.
3. k = number of words in BGPDT section.
4. m = number of words in ETT section.
2. The number of records in the EST corresponds to the number of separate element types in the model.
3. The EST is generated in subroutine TA1A.

Summary of EST Formats

		ECT Section EPT Section BGPDT Section			ETT Section	
<u>Element Type</u>	<u>Mnemonic</u>	<u>Number of words</u>	<u>Number of Words</u>	<u>Number of Words</u>	<u>Number of Words</u>	<u>Total Words Per Element</u>
1	RØD	3	5	8	1	17
2	BEAM	19	19	8	1	47
3	TUBE	3	4	8	1	16
4	SHEAR	5	3	16	1	25
5	TWIST	5	3	16	1	25
6	TRIA1	5	9	12	1	27
7	TRBSC	5	7	12	1	25
8	TRPLT	5	7	12	1	25
9	TRMEM	5	3	12	1	21
10	CØNRØD	8	0	8	1	17
11	ELAS1	5	3	0	0	8
12	ELAS2	8	0	0	0	8

DATA BLOCK DESCRIPTIONS

ECT Section EPT Section BGPDT Section

Element Type	Mnemonic	Number of Words	Number of Words	Number of Words	ETT Section Number of Words	Total Words Per Element
13	ELAS3	3	3	0	0	6
14	ELAS4	4	0	0	0	4
15	QDPLT	6	7	16	1	30
16	QDMEM	6	3	16	1	26
17	TRIA2	5	3	12	1	21
18	QUAD2	6	3	16	1	26
19	QUAD1	6	9	16	1	32
20	DAMP1	5	1	0	0	6
21	DAMP2	6	0	0	0	6
22	DAMP3	3	1	0	0	4
23	DAMP4	4	0	0	0	4
24	VISC	3	2	8	1	14
25	MASS1	5	1	0	0	6
26	MASS2	6	0	0	0	6
27	MASS3	3	1	0	0	4
28	MASS4	4	0	0	0	4
29	CØNM1	24	0	4	1	29
30	CØNM2	13	0	4	1	18
31	PLØTEL	3	0	8	1	12
34	BAR	15	18	8	1	42
35	CØNEAX	3	23	8	1	35
36	TRIARG	6	0	12	1	19
37	TRAPRG	7	0	16	1	24
38	TØRGRG	6	3	2	1	18
39	TETRA	5	0	17	1	23
40	WEDGE	7	0	25	1	33
41	HEXA1	9	0	33	1	43
42	HEXA2	9	0	33	1	43
43	FLUID2	6	0	8	0	14
44	FLUID3	7	0	12	0	19
45	FLUID4	8	0	16	0	24
46	Unused					
47	AXIF2	6	0	8	0	14
48	AXIF3	7	0	12	0	19
49	AXIF4	8	0	16	0	24
50	SLØT3	9	0	12	0	21
51	SLØT4	10	0	16	0	26
52	HBDY	7	0	17	1	25
53	DUM1	*	*	*	1	*
54	DUM2	*	*	*	1	*
55	DUM3	*	*	*	1	*
56	DUM4	*	*	*	1	*
57	DUM5	*	*	*	1	*
58	DUM6	*	*	*	1	*
59	DUM7	*	*	*	1	*
60	DUM8	*	*	*	1	*
61	DUM9	*	*	*	1	*
62	QDMEM1	6	3	16	1	26
63	QDMEM2	6	3	16	1	26
65	IHEX1	9	6	32	8	57
66	IHEX2	21	6	80	20	127
67	IHEX3	33	6	128	32	199
68	TRAIAX	6	17	12	1	34
69	TRAPAX	7	17	16	1	39

DATA BLOCK AND TABLE DESCRIPTIONS

*For the dummy elements, DUM1 thru DUM9, these values are determined at execution time upon presence of the respective ADUM1 thru ADUM9 bulk data cards.

We have from the ADUMi card thus:

GN = Number of grid points connected by PUMi

NC = Number of additional connection card values found on the CDUMi card in addition to the element ID, property or material ID, and GN = count of grid ID's.

NP = Number of additional property card values, found on the PDUMi card in addition to the property ID and material ID.

The number of words in the BGPDT section is then 4 times GN.

The number of words in the EPT section is then 1 + NP if NP is greater than zero, or is zero otherwise.

The number of words in the ECT section is then 1 + GN if NP is given greater than zero or is 2 + GN if NP is zero.

The total number of words is then the total of the ECT section plus the EPT section plus the BGPDT section plus 1 for the temperature.

Detailed EST Formats

ECT section for element type = 2:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Element ID
2-3	I	SIL numbers for grid points 1, 2
4-6	R	x, y, z (orientation vector)
7	I	Coordinate system ID for x, y, z
8-9	I	P_a, P_b
10-12	R	Z_a^1, Z_a^2, Z_a^3
13-15	R	Z_b^1, Z_b^2, Z_b^3
16-19	R	g_1, g_2, g_3, g_4

ECT Section for element type = 1, 3, 24, 31:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Element ID
2-3	I	SIL number for grid points 1, 2

ECT Section for element type = 4, 5:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Element ID
2-5	I	SIL numbers for grid points 1, 2, 3, 4

DATA BLOCK DESCRIPTIONS

ECT section for element type = 6, 7, 8, 9, 17:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Element ID
2-4	I	SIL numbers for grid points 1, 2, 3
5	R	θ (degrees)

ECT section for element type = 15, 16, 18, 19, 62, 63:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Element ID
2-5	I	SIL numbers for grid points 1, 2, 3, 4
6	R	θ (degrees)

ECT section for element type = 10:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Element ID
2-3	I	SIL numbers for grid points 1, 2
4	I	Material ID
5	R	A
6	R	J
7	R	C
8	R	non-structural mass (nsm)

ECT section for element type = 11, 20, 25:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Element ID
2-3	I	SIL numbers for grid points 1, 2
4-5	I	Component codes for grid points 1, 2

ECT section for element type = 12:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Element ID
2	R	Value
3-4	I	SIL numbers for grid points 1, 2
5-6	I	Component codes for grid points 1, 2
7-8	R	g_e , S

ECT section for element type = 13, 22, 27:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Element ID
2-3	I	SIL numbers for scalar points 1, 2

ECT section for element type = 14, 23, 28:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Element ID
2	R	Value
3-4	I	SIL numbers for scalar points 1, 2

DATA BLOCK AND TABLE DESCRIPTIONS

ECT section for element type = 21, 26:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Element ID
2	R	Value
3-4	I	SIL numbers for grid points 1, 2
5-6	I	Component codes for grid points 1, 2

ECT section for element type = 29:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Element ID
2	I	SIL number for grid point
3	I	Coordinate system ID
4-24	R	$m_{11}, m_{21}, m_{22}, m_{31}, \text{etc.}, (6 \times 6 \text{ symmetric matrix})$

ECT section for element type = 30:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Element ID
2	I	SIL number for grid point
3	I	Coordinate system ID
4	R	m
5-7	R	x_1, x_2, x_3
8-13	R	$I_{11}, I_{21}, I_{22}, I_{31}, I_{32}, I_{33}$

ECT section for element type = 34:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Element ID
2-3	I	SIL values for grid points 1, 2
4-6	R	X_1, X_2, X_3
7	I	Coordinate system ID for X_1, X_2, X_3
8-9	I	P_a, P_b
10-12	R	Z_a^1, Z_a^2, Z_a^3
13-15	R	Z_b^1, Z_b^2, Z_b^3

ECT section for element type = 35:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Element ID
2-3	I	SIL values for rings 1, 2

DATA BLOCK DESCRIPTIONS

ECT section for element type = 36:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Element ID
2-4	I	SIL values for grid points 1, 2, 3
5	R	θ (degrees)
6	I	Material ID

ECT section for element type = 37:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Element ID
2-5	I	SIL values for grid points 1, 2, 3, 4
6	R	θ (degrees)
7	I	Material ID

ECT section for element type = 38:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Element ID
2-3	I	SIL values for grid points 1, 2
4-5	R	A_1, A_2
6		Not defined

ECT section for element type = 39:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Element ID
2	I	Material ID
3-6	I	SIL values for grid points 1, 2, 3, 4

ECT section for element type = 40:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Element ID
2	I	Material ID
3-8	I	SIL values for grid points 1, 2, 3, 4, 5, 6

ECT section for element type = 41, 42:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Element ID
2	I	Material ID
3-10	I	SIL values for grid points 1, 2, 3, 4, 5, 6, 7, 8

ECT section for element type = 43:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Element ID
2,3	I	SIL values for grid points 1, 2
4	R	Density, ρ
5	R	Bulk modulus, B
6	I	Harmonic Index, N

DATA BLOCK AND TABLE DESCRIPTIONS

ECT section for element type = 44:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Element ID
2-4	I	SIL values for grid points 1, 2, 3
5	R	Density, ρ
6	R	Bulk modulus, B
7	I	Harmonic Index, N

ECT section for element type = 45:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Element ID
2-5	I	SIL values for grid points 1, 2, 3, 4
6	R	Density, ρ
7	R	Bulk modulus, B
8	I	Harmonic Index, N

ECT section for element type = 47:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Element ID
2,3	I	SIL values for grid points 1, 2
4	R	Density, ρ
5	R	Bulk modulus, B
6	I	Harmonic Index, N

ECT section for element type = 48:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Element ID
2,3,4	I	SIL values for grid points 1, 2, 3
5	R	Density, ρ
6	R	Bulk modulus, B
7	I	Harmonic Index, N

ECT section for element type = 49:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Element ID
2,3,4,5	I	SIL values for grid points 1, 2, 3, 4
6	R	Density, ρ
7	R	Bulk modulus, B
8	I	Harmonic Index, N

DATA BLOCK DESCRIPTIONS

ECT section for element type = 50:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Element ID
2,3,4	I	SIL values for grid points 1, 2, 3
5	R	Density, ρ
6	R	Bulk modulus, B
7	I	Number of Slots, M
8	I	Harmonic Index, N

ECT section for element type = 51:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Element ID
2,3,4,5	I	SIL values for grid points 1, 2, 3, 4
6	R	Density, ρ
7	R	Bulk modulus, B
8	I	Number of Slots, M
9	I	Harmonic Index, N

ECT section for element type = 52:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Element ID
2	B	FLAG
3	R	H
4	R	AF
5-8	I	SIL values for grid points 1, 2, 3, 4

ECT section for element type = 53 thru 61:
(Refer to the note under the Table Summary of EST Formats above)

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Element ID
2	I	Material ID (NP=0)
3 thru (GN+2)	I	SIL values for GN grid points. (NP=0)
(GN+3) thru (GN+2+NC)	Mixed	Additional connection data as determined by the user-programmer (Present only if NC is greater than zero)

Note that if NP is given greater than zero, the material ID will appear in the EPT section and thus words 3 thru (GN+2+NC) will be shifted up by the 1 word removed.

ECT section for element type = 65:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Element ID
2-9	I	SIL numbers for grid points 1-8

ECT section for element type = 66:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Element ID
2-21	I	SIL numbers for grid points 1-20

DATA BLOCK AND TABLE DESCRIPTIONS

ECT section for element type = 67:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Element ID
2-33	I	SIL numbers for grid points 1-32

ECT section for element type = 68:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Element ID
2-4	I	SIL values for grid points 1,2,3
5	R	θ (degrees)

ECT section for element type = 69:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Element ID
2-5	I	SIL values for grid points 1,2,3,4
6	R	θ (degrees)

DATA BLOCK DESCRIPTIONS

EPT section for element type = 1:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Material ID
2	R	A
3	R	J
4	R	C
5	R	nsm

EPT section for element type = 2:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Material ID
2	R	A
3-4	R	I_1, I_2
5	R	J
6	R	nsm
7	I	Force Element Code (FE)
8-9	R	C_1, C_2
10-11	R	D_1, D_2
12-13	R	E_1, E_2
14-15	R	F_1, F_2
16-17	R	K_1, K_2
18	R	I_{12}
19		Not defined

DATA BLOCK AND TABLE DESCRIPTIONS

EPT section for element type = 3:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Material ID
2	R	O.D.
3	R	t
4	R	nsm

EPT section for element type = 4, 5, 9, 16, 17, 18:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Material ID
2	R	t
3	R	nsm

EPT section for element type = 6, 19:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Material ID for membrane
2	R	t_1
3	I	Material ID for bending
4	R	I
5	I	Material ID for transverse shear
6	R	t_2
7	R	nsm
8-9	R	Z_1, Z_2

EPT section for element type = 7, 8, 15:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Material ID for bending
2	R	I
3	I	Material ID for transverse shear
4	R	t_2
5	R	nsm
6-7	R	Z_1, Z_2

EPT section for element type = 11, 13:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	R	K
2	R	g_e
3	R	s

EPT section for element type = 20, 22:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	R	B_e

DATA BLOCK DESCRIPTIONS

EPT section for element type = 24:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	R	C_1
2	R	C_2

EPT section for element type = 25, 27:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	R	M_e

EPT section for element type = 33:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Super element property ID

EPT section for element type = 34:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Material ID
2	R	A
3-4	R	I_1, I_2
5	R	J
6	R	nsm
7	I	FE (Force Method only)
8-9	R	C_1, C_2
10-11	R	D_1, D_2
12-13	R	E_1, E_2
14-15	R	F_1, F_2
16-17	R	K_1, K_2
18	R	I_{12}

EPT section for element type = 35:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Material ID for membrane
2R	R	T_1
3	I	Material ID for bending
4	R	I
5	I	Material ID for transverse shear
6	R	T_2
7	R	nsm
8-9	R	Z_1, Z_2
10-23	R	$\phi_i, i = 1, 14$

DATA BLOCK AND TABLE DESCRIPTIONS

EPT section for element type = 38:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Material Id
2	R	TM
3	R	TF

EPT section for element types = 53 thru 61:
(Refer to note under the table Summary of EST Formats above)

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Material ID
2 thru (1+NP)	Mixed	Property data determined by the user-programmer

The EPT section for element types 53 thru 61 is present only if NP is greater than zero as described above.

EPT section for element type = 68

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	Not Used	
2	I	Material ID
3-16	R	ϕ_I , I=1,14

EPT section for element type = 69

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	Not Used	
2	I	Material ID
3-16	R	ϕ_I , I=1,14

DATA BLOCK DESCRIPTIONS

ETT section for element type = 1-64:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	R	Average element temperature

ETT section for element type = 65:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1-8	R	Temperatures at grid points 1-8

ETT section for element type = 66:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1-20	R	Temperatures at grid points 1-20

ETT section for element type = 67:

<u>Word</u>	<u>Type</u>	<u>Item</u>
1-32	R	Temperature at grid points 1-32

Table Trailer

Word 1 = number of elements in model

Word 2-6 = are defined.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.8.2 GEI (TABLE)

Description

General Element Input.

The GEI contains one logical record for each general element in the model.

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	ID for general element
	2	I	n = number of elements in U_I list
	3	I	m = number of elements in U_D list
	4	I	SIL value for first U_I
	.		
	.		
	3+n	I	SIL value for n^{th} U_I
	4+n	I	SIL value for first U_D
	.		.
	.		.
	.		.
	3+n+m	I	SIL value for m^{th} U_D
	4+n+m	I	i = indicator of K or Z matrix
	5+n+m		
	.		
	.	R	Elements of K or Z matrix
	4+n+m+n ²		
	5+n+m+n ²		
	.		
	.	R	Elements of S matrix
	4+n+m+n ²		
	+nm		
2			Same format as record 1
.			
.			
.			
k			Same format as record 1
k+1			End-of-file

Table Trailer

Word 1 = number of general elements in the model.

Words 2-6 = zero.

DATA BLOCK DESCRIPTIONS

2.3.8.3 ECPT (TABLE)

Description

Element Connection and Properties Table.

The ECPT is a subset of the EST that is sorted by increasing internal grid point number. There is one logical record for each grid point. Each record contains a list of the elements attached to that grid point, and, for each element, a list of its grid points sorted by increasing grid point number.

Table Format

<u>Record</u>	<u>Word</u>	<u>Item</u>	
0		Header record	
1	1	SIL number for "pivot" grid or scalar point (integer)	
	2	= 6 for grid points = 1 for scalar points	
	3	Number of words for ECT section of element connected to the pivot (negative integer)	Repeated for each element connected to the pivot point
	4	Index to the ECT Table (see section 2.3.4.1) for the logical card within the logical record for this element type.	
	5	Element type (integer)	
	6+n-1	SIL numbers for grid or scalar points that define the element connection points (n = number of connection points).	
m+1		End of file (m = number of pivot points in the problem).	

Notes

1. If no elements are connected to a grid or scalar point, the record contains only one word.

Table Trailer

- Word 1 = Total number of elements in the GPTA1 common blocks.
- Word 2 = Number of grid scalar points in the model.
- Word 3 = Maximum number of elements connected to any grid or scalar point.
- Word 4 = m x n where
 m = maximum number of elements connected to any grid or scalar point
 n = maximum number of connection points for an element that is connected to a "pivot" point
- Word 5-6 = undefined

DATA BLOCK AND TABLE DESCRIPTIONS

Table Format

<u>Record</u>	<u>Word</u>	<u>Item</u>
0		Header record
1	1	+ SIL number for pivot grid or scalar point (integer)
	2	m = number of connected points (integer)
	3-2+m	Sorted list of SIL numbers of connected points
n+1		End-of-file

} repeated for each
grid or scalar
in the model

Notes

1. If the SIL number for the pivot (first word) < 0, then the pivot is a scalar point.
2. If no elements are connected to the pivot (and therefore no other grid or scalar points), the record contains only one word.
3. The pivot point is connected to itself.

Table Trailer

2.3.8.5 HEST (TABLE)

Description

See description and format of EST table - section 2.3.8.1.

2.3.8.6 HGPECT (TABLE)

Description

See description and format of ECPT - section 2.3.8.3.

DATA BLOCK DESCRIPTIONS

2.3.8.7 GPECT (TABLE)

Description

See description and format of ECPT table - section 2.3.8.3.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.9 Data Blocks Output From Module SMA1

2.3.9.1 KGGX (MATRIX)

Description

$[K_{gg}^x]$ - Partition of stiffness matrix exclusive of general elements - g set.

Matrix Trailer

Number of columns = g
 Number of rows = g
 Form = symmetric
 Type = real double precision

2.3.9.2 K4GG (MATRIX)

Description

$[K_{gg}^4]$ - Partition of structural damping matrix - g set.

Matrix Trailer

Number of columns = g
 Number of rows = g
 Form = symmetric
 Type = real double precision

2.3.9.3 GPST (TABLE)

Description

Grid Point Singularity Table

Table Format

<u>Record</u>	<u>Word</u>	<u>Item</u>	
0		Header record	
1	1	Order of singularity (1, 2, or 3)	} Repeated for each singularity
	2	N = number of SIL numbers that follow	
	3	SIL ₁	
	4	SIL ₂	
	.		
	.		
	2+N	SIL _N	
2		End-of-file	

Note

All entries are integers.

DATA BLOCK DESCRIPTIONS

Table Trailer

Word 1	=	undefined
Word 2	=	0
Word 3	=	1
Word 4	=	2
Word 5	=	1
Word 6	=	0

2.3.10 Data Blocks Output From Module SMA2

2.3.10.1 MGG (MATRIX)

Description

$[M_{gg}]$ - Partition of mass matrix - g set.

Matrix Trailer

Number of columns = g
 Number of rows = g
 Form = symmetric
 Type = real double precision

2.3.10.2 BGG (MATRIX)

Description

$[B_{gg}]$ - Partition of damping matrix - g set.

Matrix Trailer

Number of columns = g
 Number of rows = g
 Form = symmetric
 Type = real double precision

DATA BLOCK DESCRIPTIONS

2.3.11 Data Blocks Output From Module GPWG.

2.3.11.1 ØGPWG (TABLE)

Description

Grid Point Weight Generator Øutput Table.

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1			ØFP ID record
	1	I	1
	2	I	13
	3	I	External ID of grid point about which moments and interias were calculated. If External ID = 0 the basic origin was used.
	4-9		Not defined.
	10	I	98
	11-50		Not defined.
	51-146	B	96 words of title, subtitle, and label from /ØUTPUT/
2			ØFP data record
	1-36	R	[MO] 6x6 moment matrix
	37-45	R	[S] 3x3 matrix
	46-49	R	Mx, Xx, Yx, Zx
	50-53	R	My, Xy, Yy, Zy
	54-57	R	Mz, Xz, Yz, Zz
	58-66	R	Inertia matrix (3x3)
	67-69	R	Principal inertias
	70-78	R	Q matrix (3x3)
3			End-of-file

Table Trailer

Word 1 = 0

Word 2 = nonzero.

Words 3-6 = 0

2.3.12 Data Blocks Output From Module SMA3

2.3.12.1 KGG (MATRIX)

Description

$[K_{gg}]$ - Partition of stiffness matrix - g set. Contains contributions from all elements in the model, including general elements.

Matrix Trailer

Number of columns = g
 Number of rows = g
 Form = symmetric
 Type = real double precision

2.3.12.2 KGGL (MATRIX)

Description

$[K_{gg}^L]$ - Partition of the stiffness matrix of linear elements - g set. Contains contributions from all linear elements of the model including general elements. Used only in Piecewise Linear Analysis.

Matrix Trailer

Number of columns = g
 Number of rows = g
 Form = symmetric
 Type = real double precision

DATA BLOCK DESCRIPTIONS

2.3.13 Data Blocks Output From Module GP4

2.3.13.1 RG (Matrix)

Description

$[R_g]$ - Constraint equations matrix (due to multipoint constraints and rigid elements)

Matrix Trailer

Number of columns = g

Number of rows = m (number of degrees of freedom made dependent by multipoint constraints and rigid elements)

Form = rectangular

Type = real single precision

2.3.13.2 YS (MATRIX)

Description

$\{Y_s\}$ - Constrained displacement vector - s set.

Matrix Trailer

Number of columns = number of consecutive non-eigenvalue and non-differential stiffness subcases that have the same SPC and MPC set selections

Number of rows = s

Form = rectangular

Type = real single precision

2.3.13.3 USET (TABLE)

Description

Displacement set definitions table.

USET contains one logical record. Each word corresponds to each degree of freedom in the g-displacement set (in internal order) and contains ones in specified bit positions indicating the displacement sets to which the degree of freedom belongs.

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0	1-2	B	Data block name
	3	I	SPC set ID
	4	I	MPC set ID
1	1	L	Mask for first degree of freedom
	:		:
	n	L	Mask for n th degree of freedom
2			End-of-file

DATA BLOCK DESCRIPTIONS

Notes

1. Bit positions* for the various displacement sets are defined as follows:

s _b	s _g	l	a	f	n	g	r	o	s	m
22	23	24	25	26	27	28	29	30	31	32

*Bit positions are numbered 1-32 from left to right for the right-most 32 bits of the computer word.

Table Trailer

- Word 1 = zero.
- Word 2 = number of degrees of freedom in the g-displacement set (LUSET).
- Word 3 = logical "or" of all USET masks (left 16 bits).
- Word 4 = logical "or" of all USET masks (right 16 bits).
- Word 5 = zero.
- Word 6 = zero.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.13.4 HUSET (TABLE)

Description

Temperature set definitions table.

Table Format

See format of USET table - 2.3.13.3.

2.3.13.5 ASET and HASET

These data blocks are not used and are undefined.

DATA BLOCK DESCRIPTIONS

2.3.14 Data Blocks Output From Module GPSP.

2.3.14.1 ØGPST (TABLE)

Description

Unremoved Grid Point Singularities.

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1			ØFP ID record
	1	I	0
	2	I	8
	3	I	SPC set ID
	4	I	MPC set ID
	5-9		Not defined
	10	I	12
	11-50		Not defined
	51-146	B	96 words of title, subtitle, and label from /ØUTPUT/
2			ØFP data record
	1	I	External grid point ID
	2	I	Scalar point flag
	3	I	Singularity order
	4-6	I	Strongest singularity components
	7-9	I	Next strongest singularity components
	10-12	I	Weakest singularity components

*Note: The above 12 words are repeated in record 2 for each grid point with an unremoved singularity.

3 End-of-file

Table Trailer

Word 1 = 8

Word 2-6 = 0

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.15 Data Blocks Output From Module MCE1

2.3.15.1 GM (MATRIX)

Description

$[G_m]$ - Multipoint constraint transformation matrix - m set.

Matrix Header

Number of columns	=	n
Number of rows	=	m
Form	=	rectangular
Type	=	real double precision

2.3.16 Data Blocks Output From Module MCE2

2.3.16.1 KNN (MATRIX)

Description

$[K_{nn}]$ - Partition of stiffness matrix - n set.

Matrix Trailer

Number of columns = n
 Number of rows = n
 Form = symmetric
 Type = real double precision

2.3.16.2 MNN (MATRIX)

Description

$[M_{nn}]$ - Partition of mass matrix - n set.

Matrix Trailer

Number of columns = n
 Number of rows = n
 Form = symmetric
 Type = real double precision

2.3.16.3 KDNN (MATRIX)

Description

$[K_{nn}^d]$ - Partition of differential stiffness matrix - n set.

Matrix Trailer

Number of columns = n
 Number of rows = n
 Form = symmetric
 Type = real double precision

2.3.16.4 HKNN (MATRIX)

Description

$[HK_{nn}]$ - Position of conductivity matrix - n set

Matrix Trailer

Number of columns = n
 Number of rows = n
 Form = symmetric
 Type = real single/double precision

2.3.16.5 BNN (MATRIX)

Description

$[B_{nn}]$ - Partition of damping matrix - n set.

Matrix Trailer

Number of columns = n
Number of rows = n
Form = symmetric
Type = real double precision

2.3.16.6 K4NN (MATRIX)

Description

$[K_{nn}^4]$ - Partition of the structural damping matrix - n set.

Matrix Trailer

Number of columns = n
Number of rows = n
Form = symmetric
Type = real double precision

DATA BLOCK DESCRIPTIONS

2.3.16.7 HRNN (MATRIX)

Description

[HR_{nn}] - Position of radiation matrix - n set

Matrix Trailer

Number of columns = n
Number of rows = n
Form = symmetric
Type = real single/double precision

2.3.16.8 HBNN (MATRIX)

Description

[HB_{nn}] - Position of the capacity matrix - n set

Matrix Trailer

Number of columns = n
Number of rows = n
Form = symmetric
Type = real single/double precision

2.3.17 Data Blocks Output From Module SCE1

2.3.17.1 KFF (MATRIX)

Description

$[K_{ff}]$ - Partition of stiffness matrix after single-point constraints have been removed - f set.

Matrix Trailer

Number of columns = f
 Number of rows = f
 Form = symmetric
 Type = real double precision

2.3.17.2 KFS (MATRIX)

Description

$[K_{fs}]$ - Partition of stiffness matrix after single-point constraints have been removed.

Matrix Trailer

Number of columns = s
 Number of rows = f
 Form = rectangular
 Type = real double precision

2.3.17.3 KSS (MATRIX)

Description

$[K_{ss}]$ - Partition of stiffness matrix after single-point constraints have been removed - s set.

Matrix Trailer

Number of columns = s
 Number of rows = s
 Form = symmetric
 Type = real double precision

2.3.17.4 MFF (MATRIX)

Description

$[M_{ff}]$ - Partition of mass matrix after single-point constraints have been removed - f set.

Matrix Trailer

Number of columns = f
 Number of rows = f
 Form = symmetric
 Type = real double precision

DATA BLOCK DESCRIPTIONS

2.3.17.5 KDFF (MATRIX)

Description

$[K_{ff}^d]$ - Partition of differential stiffness matrix - f set.

Matrix Trailer

Number of columns = f
Number of rows = f
Form = symmetric
Type = real double precision

2.3.17.6 KDFS (MATRIX)

Description

$[K_{fs}^d]$ - Partition of differential stiffness matrix.

Matrix Trailer

Number of columns = s
Number of rows = f
Form = rectangular
Type = real double precision

2.3.17.7 KDSS (MATRIX)

Description

$[K_{ss}^d]$ - Partition of differential stiffness matrix - s set.

Matrix Trailer

Number of columns = s
Number of rows = s
Form = symmetric
Type = real double precision

2.3.17.8 BFF (MATRIX)

Description

$[B_{ff}]$ - Partition of damping matrix after single point constraints have been removed - f set.

Matrix Trailer

Number of columns = f
Number of rows = f
Form = symmetric
Type = real double precision

2.3.17.9 K4FF (MATRIX)

Description

$[K_{ff}^4]$ - Partition of structural damping matrix with single-point constraints removed - f set.

Matrix Trailer

Number of columns = f
 Number of rows = f
 Form = symmetric
 Type = real double precision

2.3.17.10 HKFF (MATRIX)

Description

$[HK_{ff}]$ - Partition of conductivity after single-point constraints have been removed - f set.

Matrix Trailer

Number of columns = f
 Number of rows = f
 Form = symmetric
 Type = real single/double precision

2.3.17.11 HKFS (MATRIX)

Description

$[HK_{fs}]$ - Partition of conductivity matrix after single-point constraints have been removed.

Matrix Trailer

Number of columns = s
 Number of rows = f
 Form = rectangular
 Type = real single/double precision

2.3.17.12 HKSS (MATRIX)

Description

$[HK_{ss}]$ - Position of stiffness matrix after single-point constraints have been removed - s set.

Matrix Trailer

Number of columns = s
 Number of rows = s
 Form = symmetric
 Type = real single/double precision

DATA BLOCK DESCRIPTIONS

2.3.17.13 HRFF

Description

[HR_{ff}] - Partition of radiation matrix after single-point constraints have been removed - f set.

Matrix Trailer

Number of columns = f
Number of rows = f
Form = symmetric
Type = real single/double precision

2.3.17.14 HBFF

Description

[HB_{ff}] - Partition of capacity matrix after single-point constraints have been removed.

Matrix Trailer

Number of columns = f
Number of rows = f
Form = symmetric
Type = real single/double precision

2.3.18 Data Blocks Output From Module SMP1

2.3.18.1 GØ (MATRIX)

Description

$[G_o]$ - Structural matrix partitioning transformation matrix.

Matrix Trailer

Number of columns = a
 Number of rows = o
 Form = rectangular
 Type = real double precision

2.3.18.2 KAA (MATRIX)

Description

$[K_{aa}]$ - Partition of stiffness matrix - a set.

Matrix Trailer

Number of columns = a
 Number of rows = a
 Form = symmetric
 Type = real double precision

2.3.18.3 KØØB (MATRIX)

Description

$[K_{oo}]$ - Partition of stiffness matrix - o set.

Matrix Trailer

Number of columns = o
 Number of rows = o
 Form = symmetric
 Type = real double precision

2.3.18.4 LØØ (MATRIX)

Description

$[L_{oo}]$ - Lower triangular factor of KØØB - o set.

Matrix Trailer

Number of columns = o
 Number of rows = o
 Form = lower triangular
 Type = real double precision

DATA BLOCK DESCRIPTIONS

2.3.18.5 U00 (MATRIX)

Description

[U₀₀] - Upper triangular factor of K00B - o set.

Matrix Trailer

Number of columns = o
Number of rows = o
Form = upper triangular
Type = real double precision

Note

This matrix is not a standard upper triangular factor. Its format is acceptable only to subroutine FBS.

2.3.18.6 MAA (MATRIX)

Description

[M_{aa}] - Partition of mass matrix - a set.

Matrix Trailer

Number of columns = a
Number of rows = a
Form = symmetric
Type = real double precision

2.3.18.7 M00 (MATRIX)

Description

[M₀₀] - Partition of mass matrix - o set.

Matrix Trailer

Number of columns = o
Number of rows = o
Form = symmetric
Type = real double precision

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.18.8 M0A (MATRIX)

Description

$[\bar{M}_{0a}]$ - Partition of mass matrix.

Matrix Trailer

Number of columns = a
 Number of rows = 0
 Form = rectangular
 Type = real double precision

2.3.18.9 BAA (MATRIX)

Description

$[B_{aa}]$ - Partition of damping matrix - a set.

Matrix Trailer

Number of columns = a
 Number of rows = a
 Form = symmetric
 Type = real double precision

2.3.18.10 K4AA (MATRIX)

Description

$[K_{aa}^4]$ - Partition of structural damping matrix - a set.

Matrix Trailer

Number of columns = a
 Number of rows = a
 Form = symmetric
 Type = real double precision

2.3.18.11 HG0 (MATRIX)

Description

$[HG_0]$ - Heat matrix partitioning transformation matrix.

Matrix Trailer

Number of columns = a
 Number of rows = 0
 Form = rectangular
 Type = real

DATA BLOCK DESCRIPTIONS

2.3.18.12 HKAA (MATRIX)

Description

$[HK_{aa}]$ - Partition of conductivity matrix - a set.

Matrix Trailer

Number of columns = a
Number of rows = a
Form = symmetric
Type = real

2.3.18.13 HK00 (MATRIX)

Description

$[HK_{00}]$ - Partition of conductivity matrix - o set.

Matrix Trailer

Number of columns = o
Number of rows = o
Form = symmetric
Type = real

2.3.18.14 HL00 (MATRIX)

Description

$[HL_{00}]$ - Lower triangular factor of HK_{00} - o set.

Matrix Trailer

Number of columns = o
Number of rows = o
Form = symmetric
Type = real

2.3.19 Data Blocks Output From Module RBMG1

2.3.19.1 KLL (MATRIX)

Description

$[K_{ll}]$ - Partition of stiffness matrix - l set.

Matrix Trailer

Number of columns = l
 Number of rows = l
 Form = symmetric
 Type = real double precision

2.3.19.2 KLR (MATRIX)

Description

$[K_{lr}]$ - Partition of stiffness matrix

Matrix Trailer

Number of columns = l
 Number of rows = r
 Form = rectangular
 Type = real double precision

2.3.19.3 KRR (MATRIX)

Description

$[K_{rr}]$ - Partition of stiffness matrix - r set.

Matrix Trailer

Number of columns = r
 Number of rows = r
 Form = symmetric
 Type = real double precision

2.3.19.4 MLL (MATRIX)

Description

$[M_{ll}]$ - Partition of mass matrix - l set.

Matrix Trailer

Number of columns = l
 Number of rows = l
 Form = symmetric
 Type = real double precision

DATA BLOCK DESCRIPTIONS

2.3.19.5 MLR (MATRIX)

Description

$[M_{lr}]$ - Partition of mass matrix.

Matrix Trailer

Number of columns = l
Number of rows = r
Form = rectangular
Type = real double precision

2.3.19.6 MRR (MATRIX)

Description

$[M_{rr}]$ - Partition of mass matrix - r set.

Matrix Trailer

Number of columns = r
Number of rows = r
Form = symmetric
Type = real double precision

2.3.19.7 HKLL (MATRIX)

Description

$[HK_{ll}]$ - Partition of conductivity matrix - l set.

Matrix Trailer

Number of columns = l
Number of rows = l
Form = symmetric
Type = real

2.3.19.8 HKLR (MATRIX)

Description

$[HK_{lr}]$ - Partition of conductivity matrix

Matrix Trailer

Number of columns = l
Number of rows = r
Form = rectangular
Type = real

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.19.9 HKRR (MATRIX)

Description

$[HK_{rr}]$ - Partition of conductivity matrix - r set.

Matrix Trailer

Number of columns = r
Number of rows = r
Form = symmetric
Type = real

DATA BLOCK DESCRIPTIONS

2.3.20 Data Blocks Output From Module RBMG2

2.3.20.1 LLL (MATRIX)

Description

$[L_{ll}]$ - Lower triangular factor of KLL - l set.

Matrix Trailer

Number of columns = l
Number of rows = l
Form = lower triangular
Type = real double precision

2.3.20.2 ULL (MATRIX)

Description

$[U_{ll}]$ - Upper triangular factor of KLL - l set.

Matrix Trailer

Number of columns = l
Number of rows = l
Form = upper triangular
Type = real double precision

Note

This matrix is not a standard upper triangular factor. Its format is acceptable only to subroutine FBS.

2.3.20.3 LBLL (MATRIX)

Description

$[L_{ll}^b]$ - Lower triangular factor of KBLL - l set.

Matrix Trailer

Number of columns = l
Number of rows = l
Form = lower triangular
Type = real double precision

2.3.20.4 UBLL (MATRIX)

Description

$[U_{ll}^b]$ - Upper triangular factor of KBLL - l set.

Matrix Trailer

Number of columns = l
 Number of rows = l
 Form = upper triangular
 Type = real double precision

Note

This matrix is not a standard upper triangular factor. Its format is acceptable only to subroutine FBS.

2.3.20.5 HLLL (MATRIX)

Description

$[HL_{ll}]$ - Lower triangular matrix of HKLL - l set.

Matrix Trailer

Number of columns = l
 Number of rows = l
 Form = lower triangular
 Type = real

2.3.21 Data Blocks Output From Module RBMG3

2.3.21.1 DM (MATRIX)

Description

[D] - Rigid body transformation matrix.

Matrix Trailer

Number of columns = l
Number of rows = r
Form = rectangular
Type = real double precision

2.3.21.2 HDM (MATRIX)

Description

[D] - Rigid Body transformation matrix.

Matrix Trailer

Number of columns = l
Number of rows = r
Form = rectangular
Type = real

2.3.22 Data Blocks Output From Module RBMG4

2.3.22.1 MR (MATRIX)

Description

$[m_r]$ - Rigid body mass matrix - r set.

Matrix Trailer

Number of columns = r
Number of rows = r
Form = symmetric
Type = real double precision

DATA BLOCK DESCRIPTIONS

2.3.23 Data Blocks Output From Module SSG1.

2.3.23.1 PG (MATRIX)

Description

$[P_g]$ - Static load vector matrix giving static loads - g set.

Matrix Trailer

Number of columns = number of subcases
Number of rows = g
Form = rectangular
Type = real single precision

2.3.23.2 PG1 (MATRIX)

Description

$[P_g^1]$ - Static load vector giving static loads for Piecewise Linear Analysis problem - g set.

Matrix Trailer

Number of columns = 1
Number of rows = g
Form = rectangular
Type = real single precision

2.3.23.3 HPG (MATRIX)

Description

$[HP_g]$ - Static load vector matrix giving static loads for heat problems - g set.

Matrix Trailer

Number of columns = number of subcase
Number of rows = g
Form = rectangular
Type = real single precision

2.3.24 Data Blocks Output From Module SSG2

2.3.24.1 QR (MATRIX)

Description

$[q_r]$ - Determinate support forces matrix - r set.

Matrix Trailer

Number of columns = number of subcases
 Number of rows = r
 Form = rectangular
 Type = real single precision

2.3.24.2 P0 (MATRIX)

Description

$[P_0]$ - Partition of the load vector matrix giving loads due to static force - o set.

Matrix Trailer

Number of columns = number of subcases
 Number of rows = o
 Form = rectangular
 Type = real single precision

2.3.24.3 PS (MATRIX)

Description

$[P_s]$ - Partition of load vector matrix giving loads in s set.

Matrix Trailer

Number of columns = number of subcases
 Number of rows = s
 Form = rectangular
 Type = real single precision

2.3.24.4 PL (MATRIX)

Description

$[P_\ell]$ - Partition of the load vector matrix giving static loads on ℓ set.

Matrix Trailer

Number of columns = number of subcases
 Number of rows = ℓ
 Form = rectangular
 Type = real single precision

DATA BLOCK DESCRIPTIONS

2.3.24.5 HQR (MATRIX)

Description

[hq_r] - Determinate support forces matrix - r set.

Matrix Trailer

Number of columns = number of subcases
Number of rows = r
Form = rectangular
Type = real single precision

2.3.24.6 HPØ (MATRIX)

Description

[hp₀] - Partition of the load vector matrix giving loads due to static force - o set.

Matrix Trailer

Number of columns = number of subcases
Number of rows = 0
Form = rectangular
Type = real single precision

2.3.24.7 HPS (MATRIX)

Description

[hp_s] - Partition of load vector matrix giving loads in s set.

Matrix Trailer

Number of columns = number of subcases
Number of rows = s
Form = rectangular
Type = real single precision

2.3.24.8 HPL (MATRIX)

Description

[hp_l] - Partition of the load vector matrix giving static loads on l set.

Matrix Trailer

Number of columns = number of subcases
Number of rows = l
Form = rectangular
Type = real single precision

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.24.9 HPF (MATRIX)

Description

[hp_f] - Partition of the load matrix giving static loads on ℓ set.

Matrix Trailer

Number of columns	=	number of subcases
Number of rows	=	ℓ
Form	=	rectangular
Type	=	real single precision

2.3.25 Data Blocks Output From Module SSG3

2.3.25.1 ULV (MATRIX)

Description

$[u_\ell]$ - Partition of the displacement vector matrix giving displacements - ℓ set.

Matrix Trailer

Number of columns = number of subcases
 Number of rows = ℓ
 Form = rectangular
 Type = real double precision

2.3.25.2 U00V (MATRIX)

Description

$[u_0^0]$ - Partition of the displacement vector matrix giving displacements in the 0 set.

Matrix Trailer

Number of columns = number of subcases
 Number of rows = 0
 Form = rectangular
 Type = real double precision

2.3.25.3 RULV (MATRIX)

Description

$[\delta P_\ell]$ - Residual vector matrix for the ℓ set.

Matrix Trailer

Number of columns = number of subcases
 Number of rows = ℓ
 Form = rectangular
 Type = real single precision

2.3.25.4 RU0V (MATRIX)

Description

$\{\delta P_0\}$ - Residual vector matrix for the 0 set.

Matrix Trailer

Number of columns = number of subcases
 Number of rows = 0
 Form = rectangular
 Type = real single precision

2.3.25.5 UBLV (MATRIX)

Description

$[u_{\ell}^b]$ - Partition of the differential stiffness displacement vector - ℓ set.

Matrix Trailer

Number of columns = 1
 Number of rows = ℓ
 Form = rectangular
 Type = real double precision

2.3.25.6 RUBLV (MATRIX)

Description

$[\delta p_{\ell}^b]$ - Differential stiffness residual vector - ℓ set.

Matrix Trailer

Number of columns = 1
 Number of rows = ℓ
 Form = rectangular
 Type = real single precision

2.3.25.7 HULV (MATRIX)

Description

$[hu_{\ell}]$ - Partition of the temperature vector matrix giving temperatures - ℓ set.

Matrix Trailer

Number of columns = number of subcases
 Number of rows = ℓ
 Form = rectangular
 Type = real double precision

2.3.25.8 HUØØV (MATRIX)

Description

$[hu_0^0]$ - Partition of the temperature vector matrix giving temperatures - 0 set.

Matrix Trailer

Number of columns = number of subcases
 Number of rows = 0
 Form = rectangular
 Type = real double precision

DATA BLOCK DESCRIPTIONS

2.3.25.9 HRULV (MATRIX)

Description

$[\delta h P_\ell]$ - Residual vector matrix for the ℓ set.

Matrix Trailer

Number of columns = number of subcases
Number of rows = ℓ
Form = rectangular
Type = real single precision

2.3.25.10 HRUØV (MATRIX)

Description

$[\delta h P_0]$ - Residual vector matrix for the 0 set.

Matrix Trailer

Number of columns = number of subcases
Number of rows = 0
Form = rectangular
Type = real single precision

2.3.26 Data Blocks Output From Module SSG4.

2.3.26.1 PLI (MATRIX)

Description

$[P_{\ell}^i]$ - Partition of load vector for inertia relief matrix giving loads due to static + inertial forces on ℓ set.

Matrix Trailer

Number of columns = number of subcases
 Number of rows = ℓ
 Form = rectangular
 Type = real single precision

2.3.26.2 P0I (MATRIX)

Description

$[P_0^i]$ - Partition of load vector for inertia relief matrix giving loads due to inertial force + static forces on 0 set.

Matrix Trailer

Number of columns = number of subcases
 Number of rows = 0
 Form = rectangular
 Type = real single precision

2.3.27 Data Blocks Output From Module SDR1

2.3.27.1 UGV (MATRIX)

Description

$[u_g]$ - Displacement vector matrix giving displacements in the g set.

Matrix Trailer

Number of columns = number of subcases in CASECC
 Number of rows = g
 Form = rectangular
 Type = real single precision

2.3.27.2 PGG (MATRIX)

Description

$[P_g]$ - Static load vector appended to include all boundary conditions - g set.

Matrix Trailer

Number of columns = number of subcases in CASECC
 Number of rows = g
 Form = rectangular
 Type = real single precision

2.3.27.3 QG (MATRIX)

Description

$[q_g]$ - Single-point constraint forces and determinate support forces matrix - g set.

Matrix Trailer

Number of columns = number of subcases in CASECC
 Number of rows = g
 Form = rectangular
 Type = real single precision

2.3.27.4 PHIG (MATRIX)

Description

$[\phi_g]$ - Eigenvector matrix giving eigenvectors (displacements) in the g set.

Matrix Trailer

Number of columns = number of eigenvalues found in READ
 Number of rows = g
 Form = rectangular
 Type = real single precision

2.3.27.5 UBGV (MATRIX)

Description

$[u_g^b]$ - Displacement vector matrix for differential stiffness giving displacements in the g set.

Matrix Trailer

Number of columns = number of factors on a DSFACT bulk data card
 Number of rows = g
 Form = rectangular
 Type = real single precision

2.3.27.6 QBG (MATRIX)

Description

$[q_g^b]$ - Single-point forces of constraint matrix for differential stiffness - g set.

Matrix Trailer

Number of columns = number of factors on a DSFACT bulk data card
 Number of rows = g
 Form = rectangular
 Type = real single precision

2.3.27.7 BQG (MATRIX)

Description

$[q_g^b]$ - Single-point forces of constraint matrix for a buckling analysis problem - g set.

Matrix Trailer

Number of columns = number of buckling modes found in READ
 Number of rows = g
 Form = rectangular
 Type = real single precision

2.3.27.8 DELTAUGV (MATRIX)

Description

$\{\delta u_g\}$ - Incremental displacement vector in piecewise linear analysis - g set.

Matrix Trailer

Number of columns = 1
 Number of rows = g
 Form = rectangular
 Type = real single precision

2.3.27.9 DELTAPG (MATRIX)

Description

$\{\delta P_g\}$ - Incremental load vector in piecewise linear analysis - g set.

Matrix Trailer

Number of columns = 1
 Number of rows = g
 Form = rectangular
 Type = real single precision

2.3.27.10 DELTAQG (MATRIX)

Description

$\{\delta q_g\}$ - Incremental vector of single-point forces of constraint in piecewise linear analysis - g set.

Matrix Trailer

Number of columns = 1
 Number of rows = g
 Form = rectangular
 Type = real single precision

2.3.27.11 CPHIP (MATRIX)

Description

$[\phi_p]$ - Complex eigenvectors in p set.

Matrix Trailer

Number of columns = number of eigenvalues found in CEAD
 Number of rows = p
 Form = rectangular
 Type = complex single precision

2.3.27.12 QPC (MATRIX)

Description

$[q_p^C]$ - Complex single-point forces of constraint - p set.

Matrix Trailer

Number of columns = number of eigenvalues found in CEAD
 Number of rows = p
 Form = rectangular
 Type = complex single precision

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.27.13 UPVC (MATRIX)

Description

$[u_p^C]$ - Frequency response solution vectors - p set.

Matrix Trailer

Number of columns = the product of the number of frequencies and number of loads
Number of rows = p
Form = rectangular
Type = complex single precision

2.3.27.14 UPV (MATRIX)

Description

$[u_p]$ - Transient solution vectors - p set.

Matrix Trailer

Number of columns = the number of output times multiplied by 3*
Number of rows = p
Form = rectangular
Type = real single precision

*Each triple is displacement, velocity and acceleration.

2.3.27.15 QP (MATRIX)

Description

$[q_p]$ - Transient single-point forces of constraint - p set.

Matrix Trailer

Number of columns = the number of output times
Number of rows = p
Form = rectangular
Type = real single precision

2.3.27.16 HUGV (MATRIX)

Description

$[HU_g]$ - Temperature vector matrix giving temperatures in g set.

Matrix Trailer

Number of columns = number of subcases in CASECC
Number of rows = g
Form = rectangular
Type = real

DATA BLOCK DESCRIPTIONS

2.3.27.17 HUPV (MATRIX)

Description

[HU_p] - Transient solution vectors - p set.

Matrix Trailers

Number of columns = number of output times multiplied by 3
Number of rows = p
Form = rectangular
Type = real

2.3.27.18 HPGG (MATRIX)

Description

[Hp_g] - Static load vector appended to include all boundary conditions - g set.

Matrix Trailer

Number of columns = number of subcases in CASECC
Number of rows = g
Form = rectangular
Type = real

2.3.27.19 HQP (MATRIX)

Description

[hq_p] - Transient single-point forces of constraint - p set.

Matrix Trailer

Number of columns = the number of output times
Number of rows = p
Form = rectangular
Type = real

2.3.27.20 HQG (MATRIX)

Description

[hq_g] - Single-point constraint forces and determinate support forces matrix - g set.

Matrix Trailer

Number of columns = number of subcases in CASECC
Number of rows = g
Form = rectangular
Type = real

2.3.28 Data Blocks Output From Module SDR2.

2.3.28.1 ØUGV1 (TABLE)

Description

Output displacement vector requests (g set, SØRT1, real).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code + 10*approach code
	2	I	1
	3	I	0
	4	I	Subcase number
	5	I	Load set ID
	6	I	0
	7	I	0
	8	I	0
	9	I	Format code
	10	I	Number of words per entry in next record = 8
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*point ID + device code
	2	I	Point type
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3)
			} repeat for each point

Notes

1. Records 1 and 2 are repeated for each vector to be output

2. Device code = $\begin{cases} 0 & \text{= x y output only} \\ 1 & \text{= print} \\ 4 & \text{= punch} \\ 5 & \text{= print and punch} \end{cases}$

3. Format code = $\begin{cases} 1 & \text{= real} \\ 2 & \text{= real/imaginary} \\ 3 & \text{= magnitude/phase} \end{cases}$

4. Approach code = 1, 3, 7, or 10

5. Point type = $\begin{cases} 1 & \text{= grid point} \\ 2 & \text{= scalar point} \\ 3 & \text{= extra point} \\ 4 & \text{= modal point} \end{cases}$

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK DESCRIPTIONS

2.3.28.2 ØUBGV1 (TABLE)

Description

Output displacement vector requests (g set, SØRT1, real)

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code + 10*approach code
	2	I	1
	3	I	0
	4	I	Subcase number
	5	I	Load set ID
	6	I	0
	7	I	0
	8	I	0
	9	I	Format code
	10	I	Number of words per entry in next record = 8
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*point ID + device code
	2	I	Point type
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3) } repeat point

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = $\begin{cases} 0 = x\ y\ \text{output only} \\ 1 = \text{print} \\ 4 = \text{punch} \\ 5 = \text{print and punch} \end{cases}$

3. Format code = $\begin{cases} 1 = \text{real} \\ 2 = \text{real/imaginary} \\ 3 = \text{magnitude/phase} \end{cases}$

4. Approach code = 4

5. Point type = $\begin{cases} 1 = \text{grid point} \\ 2 = \text{scalar point} \\ 3 = \text{extra point} \\ 4 = \text{modal point} \end{cases}$

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.28.3 QUPV1 (TABLE)

Description

Output displacement vector requests (p set, SQR1, real).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code + 10*approach code
	2	I	{ 1 = Displacement
	3	I	{ 10 = Velocity
	4	I	{ 11 = Acceleration
	5	R	0
	6	I	Subcase number
	7	I	Time
	8	I	0
	9	I	0
	10	I	Load set ID
	11-50		Format code
	51-82	B	Number of words per entry in next record = 8
	83-114	B	Not defined
	115-146	B	Title
			Subtitle
			Label
2	1	I	10*point ID + device code
	2	I	Point type
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3) } repeat point

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = { 0 = x y output only
1 = print
4 = punch
5 = print and punch

3. Format code = { 1 = real
2 = real/imaginary
3 = magnitude/phase

4. Approach code = 6

5. Point type = { 1 = grid point
2 = scalar point
3 = extra point
4 = modal point

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK DESCRIPTIONS

2.3.28.4 ØUPVC1 (TABLE).

Description

Output displacement vector requests (p set, SØRT1, complex).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code + 10*approach code
	2	I	{1001 = Displacement
	3	I	{1010 = Velocity
	4	I	{1011 = Acceleration
	5	R	0
	6	I	Subcase number
	7	I	Frequency
	8	I	0
	9	I	Load set ID
	10	I	Format code
	11-50		Number of words per entry in next record = 14
	51-82	B	Not defined
	83-114	B	Title
	115-146	B	Subtitle
			Label
2	1	I	10*point ID + device code
	2	I	Point type
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3)
	9-14	R	I(T1), I(T2), I(T3), I(R1), I(R2), I(R3)

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = {0 = x y output only
1 = print
4 = punch
5 = print and punch

3. Format code = {1 = real
2 = real/imaginary
3 = magnitude/phase

4. Approach code = 5

5. Point type = {1 = grid point
2 = scalar point
3 = extra point
4 = modal point

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.28.5 ØPG1 (TABLE).

Description

Output load vector requests (g set, SØRT1, real)

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code + 10*approach code
	2	I	2
	3	I	0
	4	I	Subcase number
	5	I	Load set ID
	6	I	0
	7	I	0
	8	I	0
	9	I	Format code
	10	I	Number of words per entry in next record = 8
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*point ID + device code
	2	I	Point type
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3) } Repeat point for each

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = $\begin{cases} 0 = x y \text{ output only} \\ 1 = \text{print} \\ 4 = \text{punch} \\ 5 = \text{print and punch} \end{cases}$

3. Format code = $\begin{cases} 1 = \text{real} \\ 2 = \text{real/imaginary} \\ 3 = \text{magnitude/phase} \end{cases}$

4. Approach code = 1, 3, 7, or 10

5. Point type = $\begin{cases} 1 = \text{grid point} \\ 2 = \text{scalar point} \\ 3 = \text{extra point} \\ 4 = \text{modal point} \end{cases}$

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK DESCRIPTIONS

2.3.28.5 ØPP1 (TABLE).

Description

Output load vector requests (p set, SØRT1, real).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code + 10*approach code
	2	I	2
	3	I	0
	4	I	Subcase number
	5	R	Time
	6	I	0
	7	I	0
	8	I	Load set ID
	9	I	Format code
	10	I	Number of words per entry in next record = 8
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*point ID + device code
	2	I	Point type
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3) } repeat point

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = $\begin{cases} 0 & = \text{x y output only} \\ 1 & = \text{print} \\ 4 & = \text{punch} \\ 5 & = \text{print and punch} \end{cases}$

3. Format code = $\begin{cases} 1 & = \text{real} \\ 2 & = \text{real/imaginary} \\ 3 & = \text{magnitude/phase} \end{cases}$

4. Approach code = 6

5. Point type = $\begin{cases} 1 & = \text{grid point} \\ 2 & = \text{scalar point} \\ 3 & = \text{extra point} \\ 4 & = \text{modal point} \end{cases}$

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.28.7 ØPPC1 (TABLE).

Description

Output load vector requests (p set, SØRT1, comolex).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code + 10*approach code
	2	I	1002
	3	I	0
	4	I	Subcase number
	5	R	Frequency
	6	I	0
	7	I	0
	8	I	Load set ID
	9	I	Format code
	10	I	Number of words per entry in next record = 14
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*point ID + device code
	2	I	Point type
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3)
	9-14	R	I(T1), I(T2), I(T3), I(R1), I(R2), I(R3)

} repeat
for
each
point

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = $\begin{cases} 0 = x y \text{ output only} \\ 1 = \text{print} \\ 4 = \text{punch} \\ 5 = \text{print and punch} \end{cases}$

3. Format code = $\begin{cases} 1 = \text{real} \\ 2 = \text{real/imaginary} \\ 3 = \text{magnitude/phase} \end{cases}$

4. Approach code = 5

5. Point type = $\begin{cases} 1 = \text{grid point} \\ 2 = \text{scalar point} \\ 3 = \text{extra point} \\ 4 = \text{modal point} \end{cases}$

Table Trailer

Words 1-6 contain no significant values.

2.3.28.8 ØQG1 (TABLE)

Description

Output forces of single-point constraint requests (g set, SØRT1, real).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code + 10*approach code
	2	I	3
	3	I	0
	4	I	Subcase number
	5	I	Load set ID
	6	I	0
	7	I	0
	8	I	0
	9	I	Format code
	10	I	Number of words per entry in next record = 8
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*point ID + device code
	2	I	Point type
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3) } repeat for each point

Notes

- Records 1 and 2 are repeated for each vector to be output.
- Device code = $\begin{cases} 0 & = \text{x y output only} \\ 1 & = \text{print} \\ 4 & = \text{punch} \\ 5 & = \text{print and punch} \end{cases}$
- Format code = $\begin{cases} 1 & = \text{real} \\ 2 & = \text{real/imaginary} \\ 3 & = \text{magnitude/phase} \end{cases}$
- Approach code = 1, 2, 3, 7, or 10
- Point type = $\begin{cases} 1 & = \text{grid point} \\ 2 & = \text{scalar point} \\ 3 & = \text{extra point} \\ 4 & = \text{modal point} \end{cases}$

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.28.9 ØQBG1 (TABLE)

Description

Output forces of single-point constraint requests (g set, SØRT1, real).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code + 10*approach code
	2	I	3
	3	I	0
	4	I	Subcase number
	5	I	Load set ID
	6	I	0
	7	I	0
	8	I	0
	9	I	Format code
	10	I	Number of words per entry in next record = 8
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*point ID + device code
	2	I	Point type
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3)} repeat for each point

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = $\begin{cases} 0 = x y \text{ output only} \\ 1 = \text{print} \\ 4 = \text{punch} \\ 5 = \text{print and punch} \end{cases}$

3. Format code = $\begin{cases} 1 = \text{real} \\ 2 = \text{real/imaginary} \\ 3 = \text{magnitude/phase} \end{cases}$

4. Approach code = 4

5. Point type = $\begin{cases} 1 = \text{grid point} \\ 2 = \text{scalar point} \\ 3 = \text{extra point} \\ 4 = \text{modal point} \end{cases}$

Table Trailer

Words 1-6 contain no significant values.

2.3.28.10 ØBQG1 (TABLE).

Description

Output forces of single-point constraint requests (g set, SQRT1, real).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code + 10*approach code
	2	I	3
	3	I	0
	4	I	Subcase number
	5	I	Mode number
	6	R	Eigenvalue
	7	I	0
	8	I	0
	9	I	Format code
	10	I	Number of words per entry in next record = 8.
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*point ID + device code
	2	I	Point type
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3) } repeat point } for each point

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = $\begin{cases} 0 = x y \text{ output only} \\ 1 = \text{print} \\ 4 = \text{punch} \\ 5 = \text{print and punch} \end{cases}$

3. Format code = $\begin{cases} 1 = \text{real} \\ 2 = \text{real/imaginary} \\ 3 = \text{magnitude/phase} \end{cases}$

4. Approach code = 8

5. Point type = $\begin{cases} 1 = \text{grid point} \\ 2 = \text{scalar point} \\ 3 = \text{extra point} \\ 4 = \text{modal point} \end{cases}$

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.28.11 ØQP1 (TABLE).

Description

Output forces of single-point constraint requests (p set, SØRT1, real).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code + 10*approach code
	2	I	3
	3	I	0
	4	I	Subcase number
	5	R	Time
	6	I	0
	7	I	C
	8	I	Load set ID
	9	I	Format code
	10	I	Number of words per entry in next record = 8
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*point ID + device code
	2	I	Point type
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3)}repeat for each point

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = $\begin{cases} 0 = x y \text{ output only} \\ 1 = \text{print} \\ 4 = \text{punch} \\ 5 = \text{print and punch} \end{cases}$

3. Format code = $\begin{cases} 1 = \text{real} \\ 2 = \text{real/imaginary} \\ 3 = \text{magnitude/phase} \end{cases}$

4. Approach code = 6

5. Point type = $\begin{cases} 1 = \text{grid point} \\ 2 = \text{scalar point} \\ 3 = \text{extra point} \\ 4 = \text{modal point} \end{cases}$

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK DESCRIPTIONS

2.3.28.12 ØQPC1 (TABLE).

Description

Output forces of single-point constraint requests (p set, SØRT1, complex).

Table Format

Record	Word	Type	Item
0			Header record
1	1	I	Device code + 10*approach code
	2	I	1003
	3	I	0
	4	I	Subcase number
	5	R/I	Frequency or mode number
	6	I/R	0 or eigenvalue (real part)
	7	I/R	0 or eigenvalue (imaginary part)
	8	I	Load set ID
	9	I	Format code
	10	I	Number of words per entry in next record = 14
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*point ID + device code
	2	I	Point type
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3)
	9-14	R	I(T1), I(T2), I(T3), I(R1), I(R2), I(R3)

repeat
for
each
point

Notes

- Records 1 and 2 are repeated for each vector to be output.
- Device code = $\begin{cases} 0 & = \text{x y output only} \\ 1 & = \text{print} \\ 4 & = \text{punch} \\ 5 & = \text{print and punch} \end{cases}$
- Format code = $\begin{cases} 1 & = \text{real} \\ 2 & = \text{real/imaginary} \\ 3 & = \text{magnitude/phase} \end{cases}$
- Approach code = 5, or 9
- Point type = $\begin{cases} 1 & = \text{grid point} \\ 2 & = \text{scalar point} \\ 3 & = \text{extra point} \\ 4 & = \text{modal point} \end{cases}$

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.28.13 ØPHIG (TABLE).

Description

Output eigenvector requests (g set, SØRT1, real).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code + 10*approach code
	2	I	7
	3	I	0
	4	I	Subcase number
	5	I	Mode number
	6	R	Eigenvalue
	7	I	0
	8	I	0
	9	I	Format code
	10	I	Number of words per entry in next record = 8
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*point ID + device code
	2	I	Point type
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3) } repeat point

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = $\begin{cases} 0 & = \text{x y output only} \\ 1 & = \text{print} \\ 4 & = \text{punch} \\ 5 & = \text{print and punch} \end{cases}$

3. Format code = $\begin{cases} 1 & = \text{real} \\ 2 & = \text{real/imaginary} \\ 3 & = \text{magnitude/phase} \end{cases}$

4. Approach code = 2, or 8

5. Point type = $\begin{cases} 1 & = \text{grid point} \\ 2 & = \text{scalar point} \\ 3 & = \text{extra point} \\ 4 & = \text{modal point} \end{cases}$

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK DESCRIPTIONS

2.3.28.14 ØCPHIP (TABLE).

Description

Output eigenvector requests (p set, SØRT1, complex).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code + 10*approach code
	2	I	1007
	3	I	0
	4	I	Subcase number
	5	I	Mode number
	6	R	Eigenvalue (real part)
	7	R	Eigenvalue (imaginary part)
	8	I	0
	9	I	Format code
	10	I	Number of words per entry in next record = 14
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*point ID + device code
	2	I	Point type
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3)
	9-14	R	I(T1), I(T2), I(T3), I(R1), I(R2), I(R3)

} repeat
for
each
point

Notes

- Records 1 and 2 are repeated for each vector to be output.
- Device code = $\begin{cases} 0 & = \text{x y output only} \\ 1 & = \text{print} \\ 4 & = \text{punch} \\ 5 & = \text{print and punch} \end{cases}$
- Format code = $\begin{cases} 1 & = \text{real} \\ 2 & = \text{real/imaginary} \\ 3 & = \text{magnitude/phase} \end{cases}$
- Approach code = 9
- Point type = $\begin{cases} 1 & = \text{grid point} \\ 2 & = \text{scalar point} \\ 3 & = \text{extra point} \\ 4 & = \text{modal point} \end{cases}$

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.28.15 ØES1 (TABLE).

Description

Output element stress requests (SQRT1, real).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code + 10*approach code
	2	I	5
	3	I	Element type
	4	I	Subcase number
	5	I/R	Time, Load set ID, or mode number
	6	R/I	Eigenvalue or 0
	7	I	0
	8	I	Load set ID or 0
	9	I	Format code
	10	I	Number of words per entry in next record = NWDS
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*element ID + device code
	2-NWDS	Mixed	Element stress data
			See 2.3.51 for details

} repeat
for each
element

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = $\begin{cases} 0 = x y \text{ output only} \\ 1 = \text{print} \\ 4 = \text{punch} \\ 5 = \text{print and punch} \end{cases}$

3. Format code = $\begin{cases} 1 = \text{real} \\ 2 = \text{real/imaginary} \\ 3 = \text{magnitude/phase} \end{cases}$

4. Approach code = 1, 2, 3, 6, 7, or 10

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK DESCRIPTIONS

2.3.28.16 ØESB1 (TABLE).

Description

Output element stress requests (SØRT1, real).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code + 10*approach code
	2	I	5
	3	I	Element type
	4	I	Subcase number
	5	I	Load set ID
	6	I	0
	7	I	0
	8	I	0
	9	I	Format code
	10	I	Number of words per entry in next record = NWDS
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*element ID + device code
	2-NWDS	Mixed	Element stress data
			See 2.3.51 for details
			} repeat } for each } element

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = $\begin{cases} 0 & = \text{x y output only} \\ 1 & = \text{print} \\ 4 & = \text{punch} \\ 5 & = \text{print and punch} \end{cases}$

3. Format code = $\begin{cases} 1 & = \text{real} \\ 2 & = \text{real/imaginary} \\ 3 & = \text{magnitude/phase} \end{cases}$

4. Approach code = 4

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.28.17 ØBES1 (TABLE).

Description

Output element stress requests (SØRT1, real).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code + 10*approach code
	2	I	5
	3	I	Element type
	4	I	Subcase number
	5	I	Mode number
	6	R	Eigenvalue
	7	I	0
	8	I	0
	9	I	Format code
	10	I	Number of words per entry in next record = NWDS
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*element ID + device code
	2-NWDS	Mixed	Element stress data
			See 2.3.51 for details
			} repeat for each element

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = $\begin{cases} 0 = x y \text{ output only} \\ 1 = \text{print} \\ 4 = \text{punch} \\ 5 = \text{print and punch} \end{cases}$

3. Format code = $\begin{cases} 1 = \text{real} \\ 2 = \text{real/imaginary} \\ 3 = \text{magnitude/phase} \end{cases}$

4. Approach code = 8

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK DESCRIPTIONS

2.3.28.18 ØESC1 (TABLE).

Description

Output element stress requests (SØRT1, complex).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code + 10*approach code
	2	I	1005
	3	I	Element type
	4	I	Subcase number
	5	R/I	Frequency or mode number
	6	I/R	0 or eigenvalue (real part)
	7	I/R	0 or eigenvalue (imaginary part)
	8	I	Load set ID
	9	I	Format code
	10	I	Number of words per entry in next record = NWDS
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*element ID + device code
	2-NWDS	Mixed	Element stress data
			See 2.3.51 for details
			} repeat for each element

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = $\begin{cases} 0 & = \text{x y output only} \\ 1 & = \text{print} \\ 4 & = \text{punch} \\ 5 & = \text{print and punch} \end{cases}$

3. Format code = $\begin{cases} 1 & = \text{real} \\ 2 & = \text{real/imaginary} \\ 3 & = \text{magnitude/phase} \end{cases}$

4. Approach code = 5, or 9

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.28.19 ØEF1 (TABLE).

Description

Output element force requests (SØRT1, real).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code + 10*approach code
	2	I	4
	3	I	Element type
	4	I	Subcase number
	5	I/R	Time, load set ID, or mode number
	6	I/R	0 or eigenvalue
	7	I	0
	8	I	Load set ID or 0
	9	I	Format code
	10	I	Number of words per entry in next record = NWDS
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*element ID + device code
	2-NWDS	Mixed	Element force data
			See 2.3.52 for details

} repeat
for each
element

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = $\begin{cases} 0 & \text{= x y output only} \\ 1 & \text{= print} \\ 4 & \text{= punch} \\ 5 & \text{= print and punch} \end{cases}$

3. Format code = $\begin{cases} 1 & \text{= real} \\ 2 & \text{= real/imaginary} \\ 3 & \text{= magnitude/phase} \end{cases}$

4. Approach code = 1, 2, 3, 6, 7, or 10

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK DESCRIPTIONS

2.3.28.20 ØEFB1 (TABLE).

Description

Output element force requests (SØRT1, real).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code + 10*approach code
	2	I	4
	3	I	Element type
	4	I	Subcase number
	5	I	Load set ID
	6	I	0
	7	I	0
	8	I	0
	9	I	Format code
	10	I	Number of words per entry in next record = NWDS
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*element ID + device code
	2-NWDS	Mixed	Element force data
			See 2.3.52 for details
			} repeat for each element

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = $\begin{cases} 0 & = \text{x y output only} \\ 1 & = \text{print} \\ 4 & = \text{punch} \\ 5 & = \text{print and punch} \end{cases}$

3. Format code = $\begin{cases} 1 & = \text{real} \\ 2 & = \text{real/imaginary} \\ 3 & = \text{magnitude/phase} \end{cases}$

4. Approach code = 4

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.28.21 ØBEF1 (TABLE).

Description

Output element force requests (SØRT1, real).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code + 10*approach code
	2	I	4
	3	I	Element type
	4	I	Subcase number
	5	I	Mode number
	6	R	Eigenvalue
	7	I	0
	8	I	0
	9	I	Format code
	10	I	Number of words per entry in next record = NWDS
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*element ID + device code
	2-NWDS	Mixed	Element force data
			See 2.3.52 for details

} repeat
for each
element

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = $\begin{cases} 0 = x y \text{ output only} \\ 1 = \text{print} \\ 4 = \text{punch} \\ 5 = \text{print and punch} \end{cases}$

3. Format code = $\begin{cases} 1 = \text{real} \\ 2 = \text{real/imaginary} \\ 3 = \text{magnitude/phase} \end{cases}$

4. Approach code = 8

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK DESCRIPTIONS

2.3.28.22 ØEFC1 (TABLE).

Description

Output element force requests (SØRT1, complex).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code + 10*approach code
	2	I	1004
	3	I	Element type
	4	I	Subcase number
	5	R/I	Frequency or mode number
	6	I/R	0 or eigenvalue (real part)
	7	I/R	0 or eigenvalue (imaginary part)
	8	I	Load set ID or 0
	9	I	Format code
	10	I	Number of words per entry in next record = NWDS
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*element ID + device code
	2-NWDS	Mixed	Element force data
			See 2.3.52 for details
			} repeat for each element

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = $\begin{cases} 0 = x y \text{ output only} \\ 1 = \text{print} \\ 4 = \text{punch} \\ 5 = \text{print and punch} \end{cases}$

3. Format code = $\begin{cases} 1 = \text{real} \\ 2 = \text{real/imaginary} \\ 3 = \text{magnitude/phase} \end{cases}$

4. Approach code = 5, or 9

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.28.23 PUGV1 (matrix - see note below)

Description

PUGV1 contains the translation components of UGV1 rotated to the basic coordinate system.

Note

The first four words of each logical record (column) contain identification data for the column. These words must be read prior to calling the appropriate unpacking routine.

Word 1 = subcase number

Word 2 = 1

Word 3 = static load set ID

Word 4 = 0.0

Matrix Trailer

Trailer is same as that for UGV1 except for word 5 and 6 (see Section 2.3.36.1).

DATA BLOCK DESCRIPTIONS

2.3.28.24 PUBGV1 (matrix - see note below)

Description

PUBGV1 contains the translation components of UBGV rotated to the basic coordinate system.

Note

The first four words of each logical record (column) contain identification data for the column. These words must be read prior to calling the appropriate unpacking routine.

Word 1 = subcase number

Word 2 = 1

Word 3 = static load set ID

Word 4 = 0.0

Matrix Trailer

Trailer is same as that for UBGV except for word 5 and 6. (See Section 2.3.27.5)

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.28.25 PPHIG (matrix - see note below)

Description

PPHIG contains the translation components of PHIG rotated to the basic coordinate system.

Note

The first four words of each logical record (column) contain identification data for the column. These words must be read prior to calling the appropriate unpacking routine.

Modal Rigid Formats

Word 1 = subcase number

Word 2 = 4

Word 3 = mode number

Word 4 = frequency (f)

Buckling Rigid Format

1 = subcase number

2 = 4

3 = -(mode number)

4 = eigenvalue (λ)

Matrix Trailer

Trailer is same as that for PHIG except for word 5 and 6 (see Section 2.3.27.4).

DATA BLOCK DESCRIPTIONS

2.3.28.26 PUGV (matrix - see note below)

Description

PUGV contains the translation components of UPV (excluding extra points) rotated to the basic coordinate system.

Note

The first four words of each logical record (column) contain identification data for the column. These words must be read prior to calling the appropriate unpacking routine.

Word 1 = subcase number

Word 2 = 3

Word 3 = dynamic load set ID

Word 4 = time

Matrix Trailer

Trailer is same as that for UGV except for word 5 and 6, (see Section 2.3.27.1).

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.28.27 PUPVC1 (matrix - see note below)

Description

PUPVC1 contains the translation components of UPVC rotated to the basic coordinate system.

Note

The first four words of each logical record (column) contain identification data for the column. These words must be read prior to calling the appropriate unpacking routine.

Word 1 = subcase number

Word 2 = 2

Word 3 = dynamic load set ID

Word 4 = frequency

Matrix Trailer

Trailer is same as that for UPVC except for word 5 and 6 (see Section 2.3.27.13).

DATA BLOCK DESCRIPTIONS

2.3.28.28 PCPHIP (matrix - see note below)

Description

PCPHIP contains the translation components of CPHIP rotated to the basic coordinate system.

Note

The first four words of each logical record (column) contain identification data for the column. These words must be read prior to calling the appropriate unpacking routine.

Word 1 = subcase number

Word 2 = 5

Word 3 = mode number

Word 4 = frequency ($|\text{Im}(\alpha)|/2\pi$)

Matrix Trailer

Trailer is same as that for CPHIP except for word 5 and 6 (see Section 2.3.27.11).

2.3.28.29 HØEF1 (TABLE)

Description

See description and format of ØEF1 table - section 2.3.28.19.

2.3.28.30 ØPPCA (TABLE)

Description

See description and format of ØPPC1 table - section 2.3.28.7.

2.3.28.31 IQP1 (TABLE)

Description

IQP1 contains modal forces of single-point constraint (p set, SØRT1)

Table Format

See format of ØQPC1 - section 2.3.28.12.

2.3.28.32 IPHIP1 (TABLE)

Description

IPHIP1 contains modal displacements (p set, SØRT1).

Table Format

See format of ØUPVC1 table - section 2.3.28.4.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.28.33 IES1 (TABLE)

Description

IES1 contains the modal element stresses (SØRT1)

Table Format

See format of ØESC1 table - section 2.3.28.18.

2.3.28.34 IEF1 (TABLE)

Description

IEF1 contains the modal element forces (SØRT1).

Table Format

See format of ØEFC1 table - section 2.3.28.22.

2.3.28.35 HØPG1 (TABLE)

Description

See description and format of ØPG1 table - section 2.3.28.5.

2.3.28.36 HØQG1 (TABLE)

Description

See description and format of ØQG1 table - section 2.3.28.8.

2.3.28.37 HØUGV1 (TABLE)

Description

Output temperature vector requests (g set, SØRT1, real).

Table Format

See format of ØUGV1 table - section 2.3.28.1.

2.3.28.38 HØES1 (TABLE)

Description

See description and format of ØES1 table - section 2.3.28.15.

2.3.28.39 HPUGV1 (TABLE)

Description

See description and format of PUGV1 table - section 2.3.28.23.

DATA BLOCK DESCRIPTIONS

2.3.28.40 ØES1A (TABLE).

Description

Output element strain/curvature requests (SØRT1, real).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code + 10*approach code
	2	I	21
	3	I	Element type
	4	I	Subcase number
	5	I/R	Time, Load set ID, or mode number
	6	R/I	Eigenvalue or 0
	7	I	0
	8	I	Load set ID or 0
	9	I	Format code
	10	I	Number of words per entry in next record = NWDS
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*element ID + device code
	2-NWDS	Mixed	Element strain/curvature data
			See 2.3.51 for details

} repeat
for each
element

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = $\begin{cases} 0 = x y \text{ output only} \\ 1 = \text{print} \\ 4 = \text{punch} \\ 5 = \text{print and punch} \end{cases}$

3. Format code = $\begin{cases} 1 = \text{real} \\ 2 = \text{real/imaginary} \\ 3 = \text{magnitude/phase} \end{cases}$

4. Approach code = 1, 2, 3, 6, 7, or 10

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK AND TABLE DESCRIPTIONS

THIS PAGE HAS BEEN LEFT BLANK INTENTIONALLY.

DATA BLOCK DESCRIPTIONS

2.3.29 Data Blocks Output From Module DPD

2.3.29.1 GPLD (TABLE)

Description

Grid Point List Dynamics.

One logical record which contains a list of all grid points, scalar points and extra points in the model in internal sort.

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	ID for first point
	:		:
	n	I	ID for n th point
2			End-of-file

Table Trailer

Word 1 = number of grid points + number of scalar points + number of extra points.

Word 2-6 = zero.

2.3.29.2 SILD (TABLE)

Description

Scalar Index List Dynamics.

Two logical records. First logical record contains scalar index values in the p-displacement set for each point in the dynamics model (internal order). These values are defined as follows:

$$SILD_1 = 1$$

$$SILD_{i+1} = \begin{cases} SILD_i + 6 & \text{if } i \text{ corresponds to a grid point} \\ SILD_i + 1 & \text{if } i \text{ corresponds to a scalar or an extra point} \end{cases}$$

The second logical record contains an equivalence between scalar index values in the g-displacement set and scalar index values in the p-displacement set.

DATA BLOCK AND TABLE DESCRIPTIONS

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Scalar index for first point
	:	:	:
	n	I	Scalar index for n th point
2	1, 2	I	SIL value, SILD value
	:	:	:
	2m-1, 2m	I	SIL value, SILD value
3			End-of-file

Table Trailer

Word 1 = number of degrees of freedom in the p-displacement set (LUSETD).
 Word 2 = number of extra points.
 Word 3-6 = zero.

2.3.29.3 USETD (TABLE)

Description

Displacement set definitions table dynamics.

USETD contains one logical record. Each word corresponds to each degree of freedom in the p-displacement set (in internal order) and contains ones in specified bit positions indicating the displacement sets to which the point belongs.

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	L	Mask for first degree of freedom
	:	:	:
	n	L	Mask for n th degree of freedom
2			End-of-file

Notes

Bit positions* for the various displacement sets are defined as follows:

d	f	e	n	p	e	s	b	s	g	l	a	f	n	g	r	o	s	m
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32			

*Bit positions are numbered 1-32 from left to right for the right-most 32 bits of the computer word.

DATA BLOCK DESCRIPTIONS

Table Trailer

Word 1 = number of degrees of freedom in the p-displacement set (LUSEDT).
 Word 2 = number of extra points.
 Word 3 = zero.
 Word 4 = logical "or" of all USETD masks.
 Word 5 = zero.
 Word 6 = zero.

2.3.29.4 EED (TABLE)

Description

Eigenvalue Extraction Data.

The EED contains one logical record for each eigenvalue extraction card type (EIGB, EIGC, EIGP, EIGR). Each logical record contains data from all cards of a given type.

Table Format

<u>Record</u>	<u>Word</u>	<u>Item</u>
0		Header record
1		EIGB data (if EIGB cards in bulk data)
2		EIGC data (if EIGC cards in bulk data)
3		EIGP data (if EIGP cards in bulk data)
4		EIGR data (if EIGR cards in bulk data)
5		End-of-file

Detailed format for EIGB data:

<u>Word</u>	<u>Type</u>	<u>Item</u>	
1-3	I	107, 1, 0	
4	I	Set ID	
5-6	B	Method	
7-8	R	F_1, F_2	
9-11	I	N_e, N_d, N_z	
12	R	E	
13-14	B	Norm	
15	I	If norm = "POINT", SIL value in a-set of normalization point	} repeated for each EIGB card in bulk data
16-21		Not defined	

DATA BLOCK AND TABLE DESCRIPTIONS

Detailed format for EIGC card:

<u>Word</u>	<u>Type</u>	<u>Item</u>	
1-3	I	207, 2, 0	
4	I	Set ID	
5-6	B	Method	
7-8	B	Norm	
9	I	If Norm = "PØINT", SIL value in <u>analysis</u> set of normalization point	
10		Not defined	
11	R	E	
12-13		Not defined	
14-15	R	α_a, ω_a	} repeated for each EIGC card in bulk data
16-17	R	α_b, ω_b	
18	R	ℓ	
19-20	I	N_e, N_d	
21		Not defined	
14+8k-21+8k	I	-1 (k = number of regions)	

Detailed format for EIGP card:

<u>Word</u>	<u>Type</u>	<u>Item</u>	
1-3	I	257, 4, 0	
5	I	Set ID	} repeated for each EIGP card in bulk data
6-7	R	α, ω	
8	I	M	

Detailed format for EIGR card:

<u>Word</u>	<u>Type</u>	<u>Item</u>	
1-3	I	307, 3, 0	
4	I	Set ID	
5-6	B	Method	
7-8	R	F_1, F_2	
9-11	I	N_e, N_d, N_z	} repeated for each EIGR card in bulk data
12	R	E	
13-14	B	Norm	
15	I	If norm = "PØINT", SIL value in a-set of normalization point	
16-21		Not defined	

Table Trailer

Word 1 =

bit 17 = 1 if EIGB record exists
 bit 18 = 1 if EIGC record exists
 bit 19 = 1 if EIGP record exists
 bit 20 = 1 if EIGR record exists
 other bits = 0

Word 2-6 = zero.

DATA BLOCK DESCRIPTIONS

2.3.29.5 EQDYN (TABLE)

Description

Equivalence between external points and scalar index values - dynamics.

EQDYN contains two logical records. The first record contains pairs of external point numbers and scalar index values in the p-displacement set for the points in external order. The second record is essentially the same as the first except that the type of point (grid, scalar, extra) is coded in the second word of the pair.

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1,2	I	ID for first point, scalar index for first point
	:	:	:
	2n-1,2n	I	ID for n th point, scalar index for n th point
2	1,2	I	ID for first point, 10*scalar index + type
	:	:	:
	2n-1,2n	I	ID for n th point, 10*scalar index + type
3			End-of-file

Note

Type = $\begin{cases} 1 & \text{for grid point} \\ 2 & \text{for scalar point} \\ 3 & \text{for extra point} \end{cases}$

Table Trailer

Word 1 = number of grid points + number of scalar points + number of extra points in dynamics model.

Word 2 = number of extra points.

Word 3-6 = zero.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.29.6 TFP00L (TABLE).

Description

Transfer Function Pool.

The TFP00L data block contains one logical record for each transfer function set defined in the bulk data on a TF bulk data card. Point and component values are converted to row and column numbers in the p-displacement set.

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Set ID
	2	I	65536*column number + row number
	3-5	R	Coefficients
			} repeated for each set of non-zero terms
n			Same format as first record
n+1			End-of-file

Table Trailer

Word 1 = number of transfer function sets.

Word 2-6 = zero.

2.3.29.7 DLT (TABLE).

Description

Dynamic Loads Table.

The header record of the DLT contains a summary of all dynamic load sets defined in the problem. The first record of the DLT contains all DL0AD cards (if DL0AD cards have been input). Each succeeding record contains all data for one dynamic load set.

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0	1-2	B	Data block name
	3	I	m = number of DL0AD set ID's
	4-3+m	I	Set ID's on DL0AD cards
	4+m-3+m+n	I	Set ID's on RL0AD1, 2 and TL0AD1, 2 cards
1	1	I	Set ID
	2	R	Scale factor
	3-4	R,I	Scale factor, set ID
	:		} repeated for each DL0AD card
	4+l,5+l	I	-1, -1

DATA BLOCK DESCRIPTIONS

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
k	1	I	Dynamic load card type
	2	I	{ = 0 no time delays ≠ 0 time delays
	3-8		See Notes
	9	I	SIL number } repeated for each
	10-12	R	A, τ , θ } dynamic load set
n+2			End-of-file

Notes

1. If no DLØAD cards have been input, the third word of the header record is zero and the DLØAD record does not exist. Therefore, record 1 of the DLT corresponds to the load set defined in word 4 of the header record.
2. DLØAD-set ID's are in sort by set ID. In record 1, set ID's within a DLØAD card are in sort.
3. Within other records, data is in sort by SIL number.
4. Formats by dynamic load card type are as follows:

1 = RLØAD1

<u>Word</u>	<u>Type</u>	<u>Item</u>
3	I	Table ID for C(f)
4	I	Table ID for D(f)
5-8		Not defined

2 = RLØAD2

<u>Word</u>	<u>Type</u>	<u>Item</u>
3	I	Table ID for B(f)
4	I	Table ID for ϕ (f)
5-8		Not defined

3 = TLØAD1

<u>Word</u>	<u>Type</u>	<u>Item</u>
3	I	Table ID for F(t)
4-8		Not defined

4 = TLØAD2

<u>Word</u>	<u>Type</u>	<u>Item</u>
3-4	R	T_{K1} , T_{K2}
5-6	R	ω_K , ϕ_K
7-8	R	n_K , α_K

Table Trailer

Word 1 = GINØ file name of DLT.
Word 2-6 = undefined.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.29.8 PSDL (TABLE)

Description

Power Spectral Density List.

The first logical record of the PSDL contains RANDPS data. Subsequent logical records contain RANDT data, one set per logical record. Each RANDT logical record contains a sorted list of unique time lags defined in the set.

Table Format

Record	Word	Type	Item
0	1,2	B	Data block name
	3	I	RANDT set ID ₁
	⋮		⋮
	2+n	I	RANDT set ID _n
1	1	I	RANDPS set ID
	2	I	Load set ID
	3	I	Load set ID
	4,5	R	Complex number
	6	I	Table ID
			} repeated for each RANDPS card in bulk data
2	1-m	R	Time lags
⋮			
n+1			Same format as record 2 Data belongs to RANDT set ID _n
n+2			End-of-file

Notes

1. RANDPS cards must be present for data block to exist. Therefore, record one always contains RANDPS data.
2. If no RANDT1 or RANDT2 cards are present in the bulk data, the header record will contain exactly two words and record two will be an end-of-file.

Table Trailer

Word 1 = {number of RANDT sets, or
65535 if no RANDT sets exist.

Word 2-6 = zero.

DATA BLOCK DESCRIPTIONS

2.3.29.9 FRL (TABLE)

Description

Frequency Response List.

The FRL contains one logical record for each different frequency set defined in the bulk data. Each record contains a sorted list of the unique frequencies defined in the set.

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0	1,2	B	Data block name
	3	I	Set ID ₁
	:		:
	2+n	I	Set ID _n
1	1-m	R	Radian frequencies belonging to set ID ₁ ($w = 2\pi f$)
:			:
n	1-k	R	Radian frequencies belonging to set ID _n ($w = 2\pi f$)
n+1			End-of-file

Table Trailer

Word 1 = number of frequency sets.

Word 2-6 = zero.

2.3.29.10 NLFT (TABLE)

Description

Non-Linear Forcing Table.

The header record of the NLFT contains a sorted list of set identification numbers for all NØLIN sets defined in the bulk data. Each logical record of the NLFT contains all data for a single set. Point and component numbers on the NØLIN cards are converted to scalar index values in both the d- and e-displacement sets.

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0	1,2	B	Data block name
	3	I	Set ID ₁
	:		:
	2+n	I	Set ID _n

DATA BLOCK AND TABLE DESCRIPTIONS

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
1	1	I	Type of nonlinear load ($1 \leq \text{type} \leq 10$) (see Note 2 below)
	2	I	SIL value in d-set
	3	I	SIL value in e-set
	4	R	Scale factor
	5	I	SIL value in d-set
	6	I	SIL value in e-set
		type = 1,5	= Table ID (integer)
		type = 2,6,9,10	= SIL value in d-set (integer)
		type = 3,7	= Scale factor (real)
		type = 4,8	= Scale factor (real)
		type = 1,5	= Not defined
		type = 2,6,9,10	= SIL value in e-set (integer)
		type = 3,7	= Not defined
		type = 4,8	= Not defined
:			
n			Same format as record 1. Data belongs to set ID _n .
n+1			End of file

Notes:

1. Within each record, the data is sorted on word 2 of each 8-word entry in the record.
2. Nonlinear load types are as follows:

<u>Type</u>	<u>Nonlinear Load Description</u>
1	Displacement-dependent NØLIN1 load
2	Displacement-dependent/displacement-dependent NØLIN2 load*
3	Displacement-dependent NØLIN3 load
4	Displacement-dependent NØLIN4 load
5	Velocity-dependent NØLIN1 load
6	Velocity-dependent/displacement-dependent NØLIN2 load*
7	Velocity-dependent NØLIN3 load
8	Velocity-dependent NØLIN4 load
9	Velocity-dependent/velocity-dependent NØLIN2 load*
10	Displacement-dependent/velocity dependent NØLIN2 load*

* Note that NØLIN2 load is a function of displacements and/or velocities at two points.

Table Trailer

Word 1 = number of NOLIN sets.
Word 2-6 = zero.

DATA BLOCK DESCRIPTIONS

2.3.29.11 TRL (TABLE).

Description

Transient Response List.

The header record of the TRL contains a list of all transient initial condition set identifications in the bulk data. Subsequent logical records contain TIC data for each set (one set per logical record). If TSTEP cards are present in the bulk data, this data follows the TIC data, one logical record for each TSTEP set.

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0	1,2	B	Data block name
	3	I	Number of TIC sets
	4	I	Set ID ₁
	:		:
	3+n	I	Set ID _n
	4+n	I	Degrees of freedom in the d-displacement set
1	1	I	SIL value in d-set
	2,3	R	U ₀ , V ₀ } repeated for each initial condition in set
:			
n			Same format as record 1
			Data belongs to set ID _n
n+1	1	I	TSTEP set ID
	2	I	N } repeated for
	3	R	Δt } each interval
	4	I	NO } in set
:			
n+m			Same format as record n+1
n+m+1			End-of-file

Notes

1. Data within each TIC record is sorted on word 1 of each 3-word entry.
2. If word 3 of the header record = 0, then the first logical record of the TRL contains TSTEP data.
3. If TSTEP data is not present in the bulk data, and end-of-file will follow the last TIC record.

Table Trailer

Word 1 = number of TIC sets.
 Word 2 = number of TSTEP sets.
 Word 3-6 = zero.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.29.12 HSILD (TABLE)

Description

See description and format of SILD table - section 2.3.29.2.

2.3.29.13 HUSETD (TABLE)

Description

Temperature set definitions table dynamics.

Table Format

See format of USETD table - section 2.3.29.3.

2.3.29.14 HDLT (TABLE)

Description

See description and format of DLT table - section 2.3.29.7.

2.3.29.15 HNLFT (TABLE)

Description

See description and format of NLFT table - section 2.3.29.10.

2.3.29.16 HTRL (TABLE)

Description

See description and format of TRL table - section 2.3.29.11.

2.3.29.17 HEQDYN (TABLE)

Description

See description and format of EQDYN table - section 2.3.29.5.

DATA BLOCK DESCRIPTIONS

2.3.30 Data Blocks Output from Module READ

2.3.30.1 LAMA (TABLE)

Description

a - Real Eigenvalue Table

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0	1	I	Header record
1			ØFP ID record
	1	I	21
	2	I	9
	3	I	0
	4-9	I	0
	10	I	7
	11-50		Not defined
	51-146	B	Title, subtitle, and label from /ØOUTPUT/
2			ØFP data record
	1	I	Mode number
	2	I	Extraction order
	3	R	λ - eigenvalue
	4	R	$\omega = \sqrt{ \lambda }$
	5	R	$f = \omega/2\pi$
	6	R	Generalized mass
	7	R	Generalized stiffness
3			End-of-file

Notes

1. The seven data words in record 2 repeat for each eigenvalue found in READ.

Table Trailer

Word 1 = 0

Word 2-6 = 0

2.3.30.2 PHIA (MATRIX)

Description

$[\Phi_a]$ - Eigenvectors matrix giving the eigenvectors (displacements) in the a set.

Matrix Trailer

Number of columns = number of eigenvalues found in READ
 Number of rows = a
 Form = rectangular
 Type = real single precision

2.3.30.3 MI (MATRIX)

Description

$[m_i]$ - Modal Mass Matrix

Matrix Trailer

Number of columns = number of eigenvectors found in READ
 Number of rows = number of eigenvectors found in READ
 Form = general
 Type = real single precision

2.3.30.4 OEIGS (TABLE)

Description

Real Eigenvalue Summary Table

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	21
	2	I	9
	3	I	2 If Inverse Power Method or FEER Method
			1 If Determinant Method
			4 If Givens Method
	4	I	0
	5	I	0
	6	I	0
	7	I	0
	8	I	0
	9	I	0
	10	I	0
	11	I	Number of eigenvalues extracted

Words 12-17 depend on the method used.

Determinant Method:

12	I	Number of passes through starting points
13	I	Number of criteria changes
14	I	Number of starting point moves
15	I	Number of triangular decompositions
16	I	Number of failures to iterate to a root
17	I	Reason for termination
		1 - All requested roots formed
		2 - Out of region predictions from every starting point
		3 - Insufficient time to extract another root
		4 - Everywhere singular matrix

DATA BLOCK DESCRIPTIONS

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
Inverse Power Method or FEER Method:			
	12	I	Number of starting points used
	13	I	Number of starting points moved
	14	I	Number of triangular decompositions
	15	I	Number of vector iterations
	16	I	0 if Inverse Power Method 1 if FEER Method
	17	I	Reasons for termination 1 - 2 Singularities encountered in a row 2 - 4 Shifts while tracking one root 3 - Regions completed 4 -3* Number of estimated roots were found 5 - All roots of problem found 6 - Number desired roots found 7 - λ outside maximum range 8 - Insufficient time to extract another root 9 - 200 iterations and 1 shift point move before locating a root
Givens Method:			
	12	I	Number of eigenvectors computed
	13	I	Number of failures to converge to an eigenvalue
	14	I	Number of failures to converge to an eigenvector
	15	I	Dummy
	16	I	Dummy
	17	I	Reason for termination 1 - Normal termination 3 - Insufficient time to evaluate eigenvectors
	18	R	Value of off-diagonal element of modal mass matrix having largest magnitude (zero where not applicable)
	19	I	Column of 18 in MI
	20	I	Row of 18 in MI
	21	I	Number of off-diagonal elements of modal mass matrix that fail to meet error criterion
	22-50		Not used
	51-146	B	Title, subtitle, label
Records 2 and 3 exist only when the Determinant Method is used.			
2	1	I	21
	2	I	9
	3	I	3
	4	I	0
	5	I	0
	6	I	0
	7	I	0
	8	I	0
	9	I	0
	10	I	6
	11-50		Not used
	51-146	B	Title, subtitle, label
3	1	I	Starting point ID
	2	R	λ - Starting point
	3	R	$\omega = \sqrt{\lambda}$ - Starting point
	4	R	$f = \omega/2\pi$ - Starting point
	5	R	Determinant at λ
			} Words 1-6 are repeated for each starting point

DATA BLOCK AND TABLE DESCRIPTIONS

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
	6	R	Scale factor (power of 10)) of determinant End-of-file
4			

Table Trailer

Nonzero

DATA BLOCK DESCRIPTIONS

2.3.31 Data Blocks Output From Module DSG1

2.3.31.1 KGG (MATRIX)

Description

$[K_{gg}^d]$ - Partition of differential stiffness matrix - g set.

Matrix Trailer

Number of columns = n
Number of rows = g
Form = symmetric
Type = real double precision

2.3.32 Data Blocks Output From Module SMP2

2.3.32.1 KDAA (MATRIX)

Description

$[K_{aa}^d]$ - Partition of differential stiffness matrix - a set.

Matrix Trailer

Number of columns = a
 Number of rows = a
 Form = symmetric
 Type = real double precision

2.3.32.2 HRAA (MATRIX)

Description

$[HK_{aa}]$ - Partition of radiation matrix - a set.

Matrix Trailer

Number of columns = a
 Number of rows = a
 Form = symmetric
 Type = real

2.3.32.3 HBAA (MATRIX)

Description

$[HB_{aa}]$ - Partition of capacity matrix - a set.

Matrix Trailer

Number of columns = a
 Number of rows = a
 Form = symmetric
 Type = real

DATA BLOCK DESCRIPTIONS

2.3.33 Data Blocks Output From Module DSMG2

2.3.33.1 KBLL (MATRIX)

Description

$[K_{\ell\ell}^b]$ - Partition of the stiffness matrix of the first order approximation to large displacements - ℓ set.

Matrix Trailer

Number of columns = ℓ
Number of rows = ℓ
Form = symmetric
Type = real double precision

2.3.33.2 KBFS (MATRIX)

Description

$[K_{fs}^b]$ - Partition of the stiffness matrix of the first order approximation to large displacements.

Matrix Trailer

Number of columns = s
Number of rows = f
Form = rectangular
Type = real double precision

2.3.33.3 KBSS (MATRIX)

Description

$[K_{ss}^b]$ - Partition of the stiffness matrix of the first order approximation to large displacements - s set.

Matrix Trailer

Number of columns = s
Number of rows = s
Form = symmetric
Type = real double precision

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.33.4 PBL (MATRIX)

Description

$\begin{Bmatrix} p \\ l \end{Bmatrix}^b$ - Partition of the load vector of the first order approximation to the large displacements - l set.

Matrix Trailer

Number of columns = 1
Number of rows = l
Form = rectangular
Type = real single precision

2.3.33.5 PBS (MATRIX)

Description

$\begin{Bmatrix} p \\ s \end{Bmatrix}^b$ - Partition of the load vector of the first order approximation to the large displacement problem - s set.

Matrix Trailer

Number of columns = 1
Number of rows = s
Form = rectangular
Type = real single precision

2.3.33.6 YBS (MATRIX)

Description

$\begin{Bmatrix} y \\ s \end{Bmatrix}^b$ - Partition of the constrained displacement vector of the first order approximation to the large displacement vector - s set.

Matrix Trailer

Number of columns = 1
Number of rows = s
Form = rectangular
Type = real single precision

2.3.33.7 UB00V (MATRIX)

Description

$\begin{Bmatrix} u \\ o \end{Bmatrix}^{ob}$ - Partition of the displacement vector of the first order approximation to the large displacement problem - o set.

Matrix Trailer

Number of columns = 1
Number of rows = o
Form = rectangular
Type = real single precision

DATA BLOCK DESCRIPTIONS

2.3.34 Data Blocks Output From Module PLA1.

2.3.34.1 KGGXL (MATRIX).

Description

$[K_{gg}^{xl}]$ - Stiffness matrix of linear elements exclusive of general elements - g set.

Matrix Trailer

Number of columns = g
Number of rows = g
Form = symmetric
Type = real double precision

2.3.34.2 ESTL (TABLE).

Description

Element Summary Table for Linear Elements.

The ESTL contains data copied from the EST data block. An element's EST data resides in the ESTL only if it is a linear element of the model.

Table Format

Same format as the EST data block output from module TA1.

Table Trailer

Word 1 = number of element entries in ESTL.

Words 2-6 = zero.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.34.3 ESTNL (TABLE).

Description

Element Summary Table for Non-Linear Elements.

The ESTNL, used only in the Piecewise Linear Analysis Rigid Format, is constructed from the Element Summary Table (EST). It contains one logical record for each element type for which at least one element of that type is non-linear (an element is defined to be non-linear if its modulus of elasticity is defined as the first derivative of a stress-strain tabular function input on a TABLES1 bulk data card) and for which a request for stress output is found. The construction of the ESTNL is as follows: the EST data block is read and each element is tested for possible non-linearity. If the element is non-linear and the user has requested element stress data to be output, its element data is copied onto the ESTNL data block and then initial stress data is appended. The only elements admissible to the ESTNL are: RØD, TUBE, CØNRØD, BAR, TRMEM, TRIA1, TRIA2, QDMEM, QUAD1, QUAD2.

Table Format

<u>Record</u>	<u>Word</u>	<u>Item</u>
0		Header record
1	1	Element type (integer)
	2 to N+1	Element EST data
	N+2 to N+M+1	Element stress data
		<div style="display: flex; align-items: center;"> <div style="margin-right: 10px;"> } repeated for } repeated for each } non-linear element </div> <div> } each admissible } element type </div> </div>

Notes

1. N is the number of words in the EST data section.
M is the number of stress words appended.
2. The number of records in the ESTNL corresponds to the number of separate admissible element types for which at least one element is non-linear.

Table Trailer

Word 1 = total number of elements in the ESTNL.
Words 2-6 = zero.

Detailed ESTNL Formats

RØD, CØNRØD:

<u>Word</u>	<u>Item</u>
1-17	EST data
18	ϵ_0^* , previous strain value once removed, initially zero
19	ϵ^* , previous strain value, initially zero
20	E^* , the previously calculated modulus of elasticity, initially the value of E given a MAT1 card.
21	T^* , the previously calculated torsional moment, initially zero

DATA BLOCK DESCRIPTIONS

TUBE:

<u>Word</u>	<u>Item</u>
1-16	EST data
17-20	Same as words 18-21 for the RØD.

BAR:

<u>Word</u>	<u>Item</u>
1-42	EST data
43	ϵ_0^* , previous strain value once removed, initially zero
44	ϵ^* , previous strain value, initially zero
45	E^* , the previously calculated modulus of elasticity, initially the value of E given on a MAT1 card
46	V_1^*
47	V_2^*
48	T^*
49	M_{1a}^*
50	M_{2a}^*
	} The previous element forces, initially zero

TRMEM:

<u>Word</u>	<u>Item</u>
1-21	EST data
22	ϵ_0^* , previous strain value once removed, initially zero
23	ϵ^* , previous strain value, initially zero
24	E^* , the previously calculated modulus of elasticity, initially the value of E given on a MAT1 card
25	σ_x^*
26	σ_y^*
27	σ_{xy}^*
	} The current membrane stresses, initially zero

TRIA1:

<u>Word</u>	<u>Item</u>
1-27	EST data
28-33	Same as words 22-27 for the TRMEM

DATA BLOCK AND TABLE DESCRIPTIONS

<u>Word</u>	<u>Item</u>
34	M_x^*
35	M_y^*
36	M_{xx}^*
37	V_x^*
38	V_y^*

The previous element forces, initially zero

TRIA2:

<u>Word</u>	<u>Item</u>
1-27	Same as words 1-27 for the TRMEM
28-32	Same as words 34-38 for the TRIA1

QDMEM:

<u>Word</u>	<u>Item</u>
1-26	EST data
27-32	Same as words 22-27 for the TRMEM

QUAD1:

<u>Word</u>	<u>Item</u>
1-32	EST data
33-38	Same as words 22-27 for the TRMEM
39-43	Same as words 34-38 for the TRIA1

QUAD2:

<u>Word</u>	<u>Item</u>
1-26	EST data
27-32	Same as words 22-27 for the TRMEM
33-37	Same as words 34-38 for the TRIA1

2.3.34.4 ECPTNL (TABLE).

Description

Element Connection and Properties Table for Non-Linear Elements.

The ECPTNL, used only in the Piecewise Linear Analysis Rigid Format, is constructed from the ECPT data block. The ECPTNL contains one logical record for each grid point or scalar point of the model. Each logical record contains Element Summary Table (EST) data plus initial element stress data appended to this data for each non-linear element connected to the pivot point. (An element is defined to be non-linear if its modulus of elasticity is defined as the first derivative of a stress-strain tabular function input on a TABLES1 card). The only elements admissible to the ECPTNL are: RØD, TUBE, CØNRØD, BAR, TRMEM, TRIA1, TRIA2, QDMEM, QUAD1, QUAD2, QUADTS, TRIATS.

Table Format

<u>Record</u>	<u>Word</u>	<u>Item</u>
0		Header record
1	1	SIL number for "pivot" grid or scalar point (integer)
	2	Element type (integer)
	3 to N+2	Element EST data
	N+3 to N+M+2	Element stress data
n+1		End-of-file

} repeated
for each grid
or scalar
point in the
model

Notes

1. N is the number of words in the EST data section.
M is the number of stress words appended. The number of stress words appended in generating the ECPTNL data block is not the same as in generating the ESTNL data block.
2. n is the total number of grid and scalar points in the model.
3. If all elements connected to a pivot point are linear, then the record contains only one word, the pivot point set negative.

Table Trailer

Word 1 = total number of element entries in the ECPTNL.

Words 2-6 = zero.

Detailed ECPTNL Formats

RØD, CØNRØD:

<u>Word</u>	<u>Item</u>
1-20	Same as ESTNL data. Note word 21 of the ESTNL is not present in the ECPTNL data for the RØD, CØNRØD.

DATA BLOCK AND TABLE DESCRIPTIONS

TUBE:

<u>Word</u>	<u>Item</u>
1-19	Same as ESTNL data. Note word 20 of the ESTNL is not present in the ECPTNL data for the TUBE.

BAR:

<u>Word</u>	<u>Item</u>
1-45	Same as ESTNL data. Note words 46-50 of the ESTNL are not present in the ECPTNL data for the BAR.

TRMEM:

<u>Word</u>	<u>Item</u>
1-27	Same as ESTNL data.

TRIA1:

<u>Word</u>	<u>Item</u>
1-33	Same as ESTNL data. Note words 34-38 of the ESTNL are not present in the ECPTNL data for the TRIA1.

TRIA2:

<u>Word</u>	<u>Item</u>
1-27	Same as ESTNL data. Note words 28-32 of the ESTNL are not present in the ECPTNL data for the TRIA2.

QDMEM:

<u>Word</u>	<u>Item</u>
1-32	Same as ESTNL data.

QUAD1:

<u>Word</u>	<u>Item</u>
1-38	Same as ESTNL data. Note words 39-43 of the ESTNL are not present in the ECPTNL data for the QUAD1.

DATA BLOCK DESCRIPTIONS

QUAD2:

Word

Item

1-32

Same as ESTNL data. Note words 33-37 of the ESTNL are not present in the ECPTNL data for the QUAD2.

2.3.35 Data Blocks Output From Module ADD

2.3.35.1 KGGSUM (MATRIX)

Description

Sum of $[K_{gg}^l]$ and $[K_{gg}^{nl}]$.

Used only in the Piecewise Linear Analysis Rigid Format and is equivalent to $[K_{gg}]$.

Matrix Trailer

Number of columns = g
 Number of rows = g
 Form = symmetric
 Type = real double precision

2.3.35.2 PG (MATRIX)

Description

$\{P_g\}$ - Incremental load vector used in Piecewise Linear Analysis.

Matrix Trailer

Number of columns = 1
 Number of rows = g
 Form = rectangular
 Type = real single precision

2.3.35.3 KDAAM (MATRIX)

Description

$[K_{aa}^{dm}]$ - The negative of $[K_{aa}^d]$ (see section 2.3.32).

Used only in the Buckling Analysis Rigid Format.

Matrix Trailer

Number of columns = a
 Number of rows = a
 Form = symmetric
 Type = real double precision

DATA BLOCK DESCRIPTIONS

2.3.36 Data Blocks Output From Module PLA2

2.3.36.1 UGV1 (MATRIX)

Description

$[u_g^1]$ - Matrix of successive sums of incremental displacement vectors - g set. Used only in the Piecewise Linear Analysis Rigid Format.

Matrix Trailer

Number of columns = number of factors on a PLFACT bulk data card
Number of rows = g
Form = rectangular
Type = real single precision

2.3.36.2 PGV1 (MATRIX)

Description

$[p_g^1]$ - Matrix of successive sums of incremental load vectors - q set. Used only in the Piecewise Linear Analysis Rigid Format.

Matrix Trailer

Number of columns = number of factors on a PLFACT bulk data card
Number of rows = g
Form = rectangular
Type = real single precision

2.3.36.3 QG1 (MATRIX)

Description

$[q_g^1]$ - Matrix of successive sums of incremental vectors of single point constraint forces - g set. Used in the Piecewise Linear Analysis Rigid Format only.

Matrix Trailer

Number of columns = number of factors on a PLFACT bulk data card
Number of rows = g
Form = symmetric
Type = real single precision

2.3.37 Data Blocks Output From Module PLA3.

2.3.37.1 ØNLES (TABLE).

Description

Output table for nonlinear element stresses.

Format

Same format as ØES1 table output from module SDR2.

Note

ØNLES is written in subroutine PLA32 of module PLA3.

Table Trailer

Word 1 = total number of element entries in ØNLES.

Word 2-6 = zero.

2.3.37.2 ESTNL1 (TABLE).

Description

Element summary table for nonlinear elements - updated.

Used only in the Piecewise Linear Analysis Rigid Format, the ESTNL1 data block is the same as the ESTNL data block except that the appended stress information is updated. See data block description for ESTNL for further details.

Table Format

Same format as the ESTNL data block.

Table Trailer

Word 1 = number of element entries in ESTNL1.

Word 2-6 = zero.

DATA BLOCK DESCRIPTIONS

2.3.38 Data Blocks Output From Module PLA4.

2.3.38.1 KGGNL (MATRIX).

Description

$[K_{gg}^{nl}]$ - Stiffness matrix of nonlinear elements - g set.

Used only in the Piecewise Linear Analysis Rigid Format.

Matrix Trailer

Number of columns = g
Number of rows = g
Form = symmetric
Type = real double precision

2.3.38.2 ECPTNL1 (TABLE).

Description

Element Connection and Properties Table for Non-Linear Elements - updated.

Used only in the Piecewise Linear Analysis Rigid Format, the ECPTNL1 data block is the same as the ECPTNL data block except that the appended stress information is updated. See description for ECPTNL for further details.

Table Format

Same format as the ECPTNL data block.

Table Trailer

Word 1 = total number of element entries in ECPTNL1.

Word 2-6 = zero.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.39 Data Blocks Output From Module CASE.

2.3.39.1 CASEXX (TABLE).

Description

Case Control data table for dynamics problems.

Table Format

The format of the records is exactly like CASECC, (see section 2.3.1.1) with dynamic looping records deleted.

Table Trailer

Word 1 = number of records in CASEXX.

Word 2-6 = zero.

DATA BLOCK DESCRIPTIONS

2.3.40 Data Blocks Output From Module MTRXIN

2.3.40.1 K2PP (MATRIX)

Description

$[K_{pp}^2]$ - Direct input stiffness matrix - p set.

Matrix Trailer

Number of columns = p
Number of rows = p
Form = square
Type = depends on input

2.3.40.2 M2PP (MATRIX)

Description

$[M_{pp}^2]$ - Direct input mass matrix - p set.

Matrix Trailer

Number of columns = p
Number of rows = p
Form = square
Type = depends on input

2.3.40.3 B2PP (MATRIX)

Description

$[B_{pp}^2]$ - Direct input damping matrix - p set.

Matrix Trailer

Number of columns = p
Number of rows = p
Form = square
Type = depends on input

2.3.40.4 HK2PP (MATRIX)

Description

See description of K2PP matrix - section 2.3.40.1.

2.3.40.5 HB2PP (MATRIX)

Description

See description of B2PP matrix - section 2.3.40.3.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.40.6 ABFL (MATRIX)

Description

$[A_{b,fl}]$ - Free surface matrix

Matrix Trailer

Number of columns = l
Number of rows = l
Form = square
Type = real

2.3.40.7 KBFL (MATRIX)

Description

$[K_{b,fl}]$ - Structure interface matrix

Matrix Trailer

Number of columns = l
Number of rows = l
Form = square
Type = real

2.3.40.8 M2DPP (MATRIX)

Description

See description of M2PP - section 2.3.40.2.

2.3.40.9 K2DPP (MATRIX)

Description

See description of K2PP - section 2.3.40.1.

2.3.40.10

The MTRXIN module may be used via DMAP to produce any desired p sized matrix from DMIG input data.

DATA BLOCK DESCRIPTIONS

2.3.41 Data Blocks Output From Module GKAD

2.3.41.1 KDD (MATRIX)

Description

$[K_{dd}]$ - Dynamic stiffness matrix - d set.

Matrix Trailer

Number of columns = d
Number of rows = d
Form = square
Type = complex double precision
 - frequency response/complex eigenvalue
 = real double precision
 - transient

2.3.41.2 BDD (MATRIX)

Description

$[B_{dd}]$ - Dynamic damping matrix - d set.

Matrix Trailer

Number of columns = d
Number of rows = d
Form = square
Type = complex double precision
 - frequency response/complex eigenvalue
 = real double precision
 - transient

2.3.41.3 MDD (MATRIX)

Description

$[M_{dd}]$ - Dynamic mass matrix - d set.

Matrix Trailer

Number of columns = d
Number of rows = d
Form = square
Type = complex double precision
 - frequency response/complex eigenvalue
 = real double precision
 - transient

2.3.41.4 GMD (MATRIX)

Description

$[G_m^d]$ - Multipoint constraint transformation matrix - dynamics.

Matrix Trailer

Number of columns = d
 Number of rows = m
 Form = rectangular
 Type = real double precision

2.3.41.5 G0D (MATRIX)

Description

$[G_0^d]$ - Omitted coordinate transformation matrix - dynamics.

Matrix Trailer

Number of columns = d
 Number of rows = 0
 Form = rectangular
 Type = real double precision

2.3.41.6 K2DD (MATRIX)

Description

$[K_{dd}^2]$ - Direct input stiffness matrix - d set.

Matrix Trailer

Number of columns = d
 Number of rows = d
 Form = square
 Type = complex double precision/real double precision

2.3.41.7 M2DD (MATRIX)

Description

$[M_{dd}^2]$ - Direct input mass matrix - d set.

Matrix Trailer

Number of columns = d
 Number of rows = d
 Form = square
 Type = complex double precision/real double precision

DATA BLOCK DESCRIPTIONS

2.3.41.8 B2DD (MATRIX)

Description

$[B_{dd}^2]$ - Direct input damping matrix - d set.

Matrix Trailer

Number of columns = d
Number of rows = d
Form = square
Type = complex double precision/real double precision

2.3.41.9 HKDD (MATRIX)

Description

$[HK_{dd}]$ - Dynamic conductivity matrix - d set.

Matrix Trailer

Number of columns = d
Number of rows = d
Form = square
Type = real

2.3.41.10 HBDD (MATRIX)

Description

$[HB_{dd}]$ - Dynamic capacity matrix - d set.

Matrix Trailer

Number of columns = d
Number of rows = d
Form = square
Type = real

2.3.41.11 HRDD (MATRIX)

Description

$[HR_{dd}]$ - Dynamic radiation matrix - d set.

Matrix Trailer

Number of columns = d
Number of rows = d
Form = square
Type = real

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.41.12 HG0D (MATRIX)

Description

See description of G0D - section 2.3.41.5.

2.3.41.13 HK2DD (MATRIX)

Description

$[HK_{dd}^2]$ - Direct input conductivity matrix - d set.

Matrix Trailer

Number of columns = d
Number of rows = d
Form = square
Type = real

2.3.41.14 HM2DD (MATRIX)

Description

$[HM_{dd}^2]$ - Direct input radiation matrix - d set.

Matrix Trailer

Number of columns = d
Number of rows = d
Form = square
Type = real

2.3.41.15 HB2DD (MATRIX)

Description

$[HB_{dd}^2]$ - Direct input capacity matrix - d set.

Matrix Trailer

Number of columns = d
Number of rows = d
Form = square
Type = real

DATA BLOCK DESCRIPTIONS

2.3.42 Data Blocks Output From Module CEAD

2.3.42.1 PHID (MATRIX)

Description

$[\phi_d]$ - Complex eigenvectors in the d set.

Matrix Trailer

Number of columns = number of eigenvalues found in CEAD
 Number of rows = d
 Form = rectangular
 Type = complex single precision

2.3.42.2 CLAMA (TABLE)

Description

λ - Complex eigenvalue table.

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1			ØFP ID record
	1	I	90
	2	I	1006
	3-9	I	0
	10	I	6
	11-50		Not defined
	51-146	B	Title, subtitle, and label from /ØUTPUT/
2			ØFP data record
	1	I	Mode number
	2	I	Extraction order
	3	R	Real part of eigenvalue
	4	R	Imaginary part of eigenvalue
	5	R	$ \text{Im}(\lambda) /2\pi$
	6	R	$-2*\text{Re}(\lambda)/ \text{Im}(\lambda) $

Note: The 6 data words are repeated in record 2 for each eigenvalue found in CEAD.

3 End-of-file

Table Trailer

Word 1 = 1006
 Word 2 = 0
 Word 3 = 0
 Word 4 = 0
 Word 5 = 6
 Word 6 = 0

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.42.3 ØCEIGS (TABLE).

Description

Complex eigenvalue summary table.

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0		B	Header record
1	1	I	0
	2	I	1009
	3	I	1 if determinant
			2 if inverse power or FEER
	4-10	I	0
	11	I	Number of eigenvalues extracted
	12-18		Depend on the method used
	Determinant		
	12	I	Number of passes through starting points
	13	I	Number of criteria changes
	14	I	Number of starting point moves
	15	I	Number of decompositions
	16	I	Number of failures to iterate to a root
	17	I	Number of predictions outside the region
	18	I	Reason for termination
			1 - all requested roots found
			2 - out of region prediction from every starting point
			3 - insufficient time to extract another root
			4 - everywhere singular matrix
	Inverse Power or FEER		
	12-18		Identical to words 12-18 for Inverse Power Method section of the ØEIGS data block output from the READ module (see Section 2.3.30.4) (except that the contents of words 16 and 17 are interchanged and word 18 is not used).
	19-50		Not defined
	51-146	B	Title, subtitle, label
Record 1 will be repeated for each region for Inverse Power.			
Records 2 + 3 exist only when METHØD = DETM.			
2	1	I	0
	2	I	1009
	3	I	3
	4-9	I	0
	10	I	6
	11-50		Not defined
	51-146	B	Title, subtitle, label

DATA BLOCK DESCRIPTIONS

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
3	1	I	Starting point number in region
	2	R	Real part of starting point
	3	R	Imaginary part of starting point
	4	R	Magnitude of starting point
	5	R	Phase of starting point
	6	I	Scale factor (power of 10) of magnitude

Words 1-6 are repeated for each starting point in each region.

4

End-of-file

Table Trailer

Non-zero.

2.3.42.4 PHIH (MATRIX)

Description

$[\phi_h]$ - Complex eigenvectors in the h set.

Matrix Trailer

Number of columns = number of eigenvalues found in CEAD
 Number of rows = h
 Form = rectangular
 Type = complex single precision

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.43 Data Blocks Output From Module VDR

2.3.43.1 ØPHID (TABLE)

Description

Output complex eigenvectors requests (solution set, SØRT1, complex).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0	1-2	B	Data block name
	3-5	I	Month, day, year
	6	I	Time
	7	I	1
1	1	I	Device code + 10 * approach code
	2	I	1014
	3	I	0
	4	I	Subcase number
	5	I	Mode number
	6-7	R	Complex eigenvalue
	8	I	0
	9	I	Format code
	10	I	Number of words per entry in record 2 = 14.
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10 * point ID + device code
	2	I	Point type
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3)
	9-14	R	I(T1), I(T2), I(T3), I(R1), I(R2), I(R3)

repeated
for
each
point

Notes

- Records 1 and 2 are repeated for each vector to be output.
- Device code = $\begin{cases} 0 & = \text{x y output only} \\ 1 & = \text{print} \\ 4 & = \text{punch} \\ 5 & = \text{print and punch} \end{cases}$
- Format code = $\begin{cases} 1 & = \text{real} \\ 2 & = \text{real/imaginary} \\ 3 & = \text{magnitude/phase} \end{cases}$
- Approach code = 9
- Point type = $\begin{cases} 1 & = \text{grid point} \\ 2 & = \text{scalar point} \\ 3 & = \text{extra point} \\ 4 & = \text{modal point} \end{cases}$
- Components (words 3-14 of even numbered records) which are not in the solution set are replaced by an integer 1.

DATA BLOCK DESCRIPTIONS

Table Trailer

Word 1 = (sum of all words in even numbered records)/65536
 Word 2 = remainder from division above
 Word 3-6 = zero.

2.3.43.2 ØUDVC1 (TABLE)

Description

Output displacement requests (solution set, SØRT1, complex)

Table Format

Record	Word	Type	Item
0	1-2	B	Data block name
	3-5	I	Month, day, year
	6	I	Time
	7	I	1
1	1	I	Device code + 10 * approach code
	2	I	{1015 = displacement
			{1016 = velocity
			{1017 = acceleration
	3	I	0
	4	I	Subcase number
	5	R	Frequency
	6	I	0
	7	I	0
	8	I	Dynamic load set ID
	9	I	Format code
	10	I	Number of words per entry in record 2 = 14
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10 * point ID + device code
	2	I	Point type
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3)
	9-14	R	I(T1), I(T2), I(T3), I(R1), I(R2), I(R3)

}repeated
for
each
point

Notes

- Records 1 and 2 are repeated for each vector to be output.
- Device code = $\begin{cases} 0 & \text{= x y output only} \\ 1 & \text{= print} \\ 4 & \text{= punch} \\ 5 & \text{= print and punch} \end{cases}$
- Format code = $\begin{cases} 1 & \text{= real} \\ 2 & \text{= real/imaginary} \\ 3 & \text{= magnitude/phase} \end{cases}$
- Approach code = 6

DATA BLOCK AND TABLE DESCRIPTIONS

Notes cont'd.

5. Point type = $\begin{cases} 1 = \text{grid point} \\ 2 = \text{scalar point} \\ 3 = \text{extra point} \\ 4 = \text{modal point} \end{cases}$

6. Components (words 3-14 of even numbered records) which are not in the solution set are replaced by an integer 1.

Table Trailer

Word 1 = (sum of all words in even numbered records)/65536.

Word 2 = remainder from division above.

Word 3-6 = zero.

2.3.43.3 ØUDV1 (TABLE)

Description

Output displacement requests (solution set, SØRT1, real).

Table Format

Record	Word	Type	Item
0	1-2	B	Data block name
	3-5	I	Month, day, year
	6	I	Time
	7	I	1
1	1	I	Device code + 10 * approach code
	2	I	{ 15 = displacement
			16 = velocity
			17 = acceleration
	3	I	0
	4	I	Subcase number
	5	R	Time
	6	I	0
	7	I	0
	8	I	Dynamic load set ID
	9	I	Format code = 1
	10	I	Number of words per entry in record 2 = 8
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10 * point ID + device code } repeated Point type. } for each T1, T2, T3, R1, R2, R3 } point
	2	I	
	3-8	R	

DATA BLOCK DESCRIPTIONS

Notes

1. Records 1 and 2 are repeated for each vector to be output.
2. Device code = $\begin{cases} 0 = x\ y\ \text{output only} \\ 1 = \text{print} \\ 4 = \text{punch} \\ 5 = \text{print and punch} \end{cases}$
3. Format code = $\begin{cases} 1 = \text{real} \\ 2 = \text{real/imaginary} \\ 3 = \text{magnitude/phase} \end{cases}$
4. Approach code = 7
5. Point type = $\begin{cases} 1 = \text{grid point} \\ 2 = \text{scalar point} \\ 3 = \text{extra point} \\ 4 = \text{modal point} \end{cases}$
6. Components (words 3-8 of even numbered records) which are not in the solution set are replaced by an integer 1.

Table Trailer

Word 1 = (sum of all words in even numbered records)/65536.
 Word 2 = remainder from division above.
 Word 3-6 = zero.

2.3.43.4 ØPNL1 (TABLE)

Description

Output nonlinear load requests (solution set, SØRT1, real)

Table Format

Record	Word	Type	Item
0	1-2	B	Data block name
	3-5	I	Month, day, year
	6	I	Time
	7	I	1
1	1	I	Device code + 10 * approach code
	2	I	12
	3	I	0
	4	I	Subcase number
	5	R	Time
	6	I	0
	7	I	0
	8	I	Dynamic load set ID
	9	I	Format code
	10	I	Number of words per entry in record 2 = 8
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label

DATA BLOCK AND TABLE DESCRIPTIONS

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
2	1	I	10 * point ID + device code
	2	I	Point type
	3-8	R	T1, T2, T3, R1, R2, R3

} repeated for each point

Notes

- Records 1 and 2 are repeated for each vector to be output.
- Device code = $\begin{cases} 0 = \text{x y output only} \\ 1 = \text{print} \\ 4 = \text{punch} \\ 5 = \text{print and punch} \end{cases}$
- Format code = $\begin{cases} 1 = \text{real} \\ 2 = \text{real/imaginary} \\ 3 = \text{magnitude/phase} \end{cases}$
- Approach code = 7
- Point type = $\begin{cases} 1 = \text{grid point} \\ 2 = \text{scalar point} \\ 3 = \text{extra point} \\ 4 = \text{modal point} \end{cases}$
- Components (words 3-8 in even numbered records) which are not in the solution set are replaced by an integer 1.

Table Trailer

Word 1 = (sum of all words in even numbered records)/65536.
 Word 2 = remainder from division above.
 Word 3-6 = zero.

2.3.43.5 ØPHIH (TABLE)

Description

Output complex eigenvector requests (solution set, SQRT1, complex).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0	1-2	B	Data block name
	3-5	I	Month, day, year
	6	I	Time
	7	I	1
1	1	I	Device code + 10 * approach code
	2	I	1014
	3	I	0
	4	I	Subcase number
	5	I	Mode number
	6-7	R	Complex eigenvalue
	8	I	0
	9	I	Format code
	10	I	Number of words per entry in record 2 = 14

DATA BLOCK DESCRIPTIONS

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10 * point ID + device code
	2	I	Point type
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3)
	9-14	R	I(T1), I(T2), I(T3), I(R1), I(R2), I(R3)

} repeated
for
each
point

Notes

- Records 1 and 2 are repeated for each vector to be output
- Device code = $\begin{cases} 0 = \text{x y output only} \\ 1 = \text{print} \\ 4 = \text{punch} \\ 5 = \text{print and punch} \end{cases}$
- Format code = $\begin{cases} 1 = \text{real} \\ 2 = \text{real/imaginary} \\ 3 = \text{magnitude/phase} \end{cases}$
- Approach code = 9
- Point type = $\begin{cases} 1 = \text{grid point} \\ 2 = \text{scalar point} \\ 3 = \text{extra point} \\ 4 = \text{modal point} \end{cases}$
- Components (words 3-14 of even numbered records) which are not in the solution set are replaced by an integer 1.

Table Trailer

- Word 1 = (sum of all words in even numbered records)/65536.
- Word 2 = remainder from division above.
- Word 3-6 = zero.

2.3.43.6 ØUHVC1 (TABLE)

Description

Output displacement requests (solution set, SØRT1, complex).

DATA BLOCK AND TABLE DESCRIPTIONS

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0	1-2	B	Data block name
	3-5	I	Month, day, year
	6	I	Time
	7	I	1
1	1	I	Device code + 10 * approach code
	2	I	{ 1015 = displacement
			{ 1016 = velocity
			{ 1017 = acceleration
	3	I	0
	4	I	Subcase number
	5	R	Frequency
	6	I	0
	7	I	0
	8	I	Dynamic load set ID
	9	I	Format code
	10	I	Number of words per entry in record 2 = 14
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10 * point ID + device code
	2	I	Point type
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3)
	9-14	R	I(T1), I(T2), I(T3), I(R1), I(R2), I(R3)

Notes

- Records 1 and 2 are repeated for each vector to be output.
- Device code = $\begin{cases} 0 = \text{x y output only} \\ 1 = \text{print} \\ 4 = \text{punch} \\ 5 = \text{print and punch} \end{cases}$
- Format code = $\begin{cases} 1 = \text{real} \\ 2 = \text{real/imaginary} \\ 3 = \text{magnitude/phase} \end{cases}$
- Approach code = 5
- Point type = $\begin{cases} 1 = \text{grid point} \\ 2 = \text{scalar point} \\ 3 = \text{extra point} \\ 4 = \text{modal point} \end{cases}$
- Components (words 3-14 of even numbered records) which are not in the solution set are replaced by an integer 1.

Table Trailer

- Word 1 = (sum of all words in even numbered records)/65536.
- Word 2 = remainder from division above.
- Word 3-6 = zero.

DATA BLOCK DESCRIPTIONS

2.3.43.7 ØUHV1 (TABLE)

Description

Output displacement requests (solution set, SØRT1, real).

Table Format

Record	Word	Type	Item
0	1-2	B	Data block name
	3-5	I	Month, day, year
	6	I	Time
	7	I	1
1	1	I	Device code + 10 * approach code
	2	I	{ 15 = displacement
	3	I	{ 16 = velocity
	4	I	{ 17 = acceleration
	5	R	0
	6	I	Subcase number
	7	I	Time
	8	I	0
	9	I	Dynamic load set ID
	10	I	Format code = 1
	11-50		Number of words per entry in record 2 = 8
	51-82	B	Not defined
	83-114	B	Title
	115-146	B	Subtitle
2	1	I	Label
	2	I	10 * point ID + device code { repeated
	3-8	R	point type { for each point T1, T2, T3, R1, R2, R3

Notes

- Records 1 and 2 are repeated for each vector to be output.
- Device code = $\begin{cases} 0 = x y \text{ output only} \\ 1 = \text{print} \\ 4 = \text{punch} \\ 5 = \text{print and punch} \end{cases}$
- Format code = $\begin{cases} 1 = \text{real} \\ 2 = \text{real/imaginary} \\ 3 = \text{magnitude/phase} \end{cases}$
- Approach code = 7
- Point type = $\begin{cases} 1 = \text{grid point} \\ 2 = \text{scalar point} \\ 3 = \text{extra point} \\ 4 = \text{modal point} \end{cases}$
- Components (words 3-8 of even numbered records) which are not in the solution set are replaced by an integer 1.

DATA BLOCK AND TABLE DESCRIPTIONS

Table Trailer

Word 1 = (sum of all words in even numbered records)/65536.

Word 2 = remainder from division above.

Word 3-6 = zero.

2.3.43.8 HØPNL1 (TABLE)

Description

See description and format for ØPNL1 table - section 2.3.43.4.

2.3.43.9 HØUDV1 (TABLE)

Description

Output temperature requests (solution set, SØRT1, real)

Table Format

See format of ØUDV1 table - section 2.3.43.3.

DATA BLOCK DESCRIPTIONS

2.3.44 Data Blocks Output From Module FRRD

2.3.44.1 UDVF (MATRIX)

Description

$[u_d^f]$ - Displacement vector matrix in a frequency response problem - d set.

Matrix Trailer

Number of columns = number of frequencies multiplied by the number of loads
Number of rows = d
Form = rectangular
Type = complex single precision

2.3.44.2 PSF (MATRIX)

Description

$[p_s^f]$ - Load vector for frequency response - s set.

Matrix Trailer

Number of columns = number of frequencies multiplied by the number of loads
Number of rows = s
Form = rectangular
Type = complex single precision

2.3.44.3 PDF (MATRIX)

Description

$\{p_d^f\}$ - Dynamic load matrix for frequency analysis - d set.

Matrix Trailer

Number of columns = number of frequencies multiplied by the number of loads
Number of rows = d
Form = rectangular
Type = complex single precision

2.3.44.4 PPF (MATRIX)

Description

$[p_p^f]$ - Dynamic loads for frequency response - p set.

Matrix Trailer

Number of columns = number of frequencies multiplied by the number of loads
Number of rows = p
Form = rectangular
Type = complex single precision

DATA BLOCK AND TABLE DESCRIPTIONS

Note

The header record contains the list of frequencies.

2.3.44.5 UHVF (MATRIX)

Description

$[u_h^f]$ - Modal frequency response solution vectors - h set.

Matrix Trailer

Number of columns	=	number of frequencies multiplied by the number of loads
Number of rows	=	h
Form	=	rectangular
Type	=	complex single precision

DATA BLOCK DESCRIPTIONS

2.3.45 Data Blocks Output From Module SDR3.

2.3.45.1 ØPP2 (TABLE)

Description

Output load vector requests (p set, SØRT2, real).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code + 10*approach code
	2	I	2002
	3	I	0
	4	I	Subcase number
	5	I	10*point ID + device code
	6	I	0
	7	I	0
	8	I	Dynamic load set ID
	9	I	Format code = 1
	10	I	Number of words per entry in next record = 8
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	R	Label
2	1	R	Time
	2	I	Point type
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3) } repeat for each time step

Notes

- Records 1 and 2 are repeated for each vector to be output.
- Device code = $\begin{cases} 0 & \text{= x y output only} \\ 1 & \text{= print} \\ 4 & \text{= punch} \\ 5 & \text{= print and punch} \end{cases}$
- Format code = $\begin{cases} 1 & \text{= real} \\ 2 & \text{= real/imaginary} \\ 3 & \text{= magnitude/phase} \end{cases}$
- Approach code = 6
- Point type = $\begin{cases} 1 & \text{= grid point} \\ 2 & \text{= scalar point} \\ 3 & \text{= extra point} \\ 4 & \text{= modal point} \end{cases}$

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.45.2 ØQP2 (TABLE).

Description

Output forces of single-point constraint (p set, SØRT2, real).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code + 10*approach code
	2	I	2003
	3	I	0
	4	I	Subcase number
	5	I	10*point ID + device code
	6	I	0
	7	I	0
	8	I	Dynamic load set ID
	9	I	Format code = 1
	10	I	Number of words per entry in next record = 8
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	R	Time
	2	I	Point type
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3)
			} repeat for each time step

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = $\begin{cases} 0 & \text{= x y output only} \\ 1 & \text{= print} \\ 4 & \text{= punch} \\ 5 & \text{= print and punch} \end{cases}$

3. Format code = $\begin{cases} 1 & \text{= real} \\ 2 & \text{= real/imaginary} \\ 3 & \text{= magnitude/phase} \end{cases}$

4. Approach code = 6

5. Point type = $\begin{cases} 1 & \text{= grid point} \\ 2 & \text{= scalar point} \\ 3 & \text{= extra point} \\ 4 & \text{= modal point} \end{cases}$

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK DESCRIPTIONS

2.3.45.3 ØUPV2 (TABLE).

Description

Output displacement vector requests (p set, SØRT2, real).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code + 10*approach code
	2	I	{ 2001 = Displacement
	3	I	{ 2010 = Velocity
	4	I	{ 2011 = Acceleration
	5	I	0
	6	I	Subcase number
	7	I	10*point ID + device code
	8	I	0
	9	I	0
	10	I	Dynamic load set ID
	11-50	I	Format code = 1
	51-82	B	Number of words per entry in next record = 8
	83-114	B	Not defined
	115-146	B	Title
			Subtitle
			Label
2	1	R	Time
	2	I	Point type
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3) } repeat for each time step

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = { 0 = x y output only
1 = print
4 = punch
5 = print and punch

3. Format code = { 1 = real
2 = real/imaginary
3 = magnitude/phase

4. Approach code = 6

5. Point type = { 1 = grid point
2 = scalar point
3 = extra point
4 = modal point

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.45.4 ØES2 (TABLE).

Description

Output element stress requests (SØRT2, real).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code + 10*approach code
	2	I	2005
	3	I	Element type
	4	I	Subcase number
	5	I	10*element ID + device code
	6	I	0
	7	I	0
	8	I	Dynamic load set ID
	9	I	Format code = 1
	10	I	Number of words per entry in next record = "IDS
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	R	Time
	2-NWDS	Mixed	Element stress data
			See section 2.3.51 for details
			} repeat for each time step

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = $\begin{cases} 0 = x y \text{ output only} \\ 1 = \text{print} \\ 4 = \text{punch} \\ 5 = \text{print and punch} \end{cases}$

3. Format code = $\begin{cases} 1 = \text{real} \\ 2 = \text{real/imaginary} \\ 3 = \text{magnitude/phase} \end{cases}$

4. Approach code = 6

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK DESCRIPTIONS

2.3.45.5 ØEF2 (TABLE).

Description

Output element force requests (SØRT2, real).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code + 10*approach code
	2	I	2004
	3	I	Element type
	4	I	Subcase number
	5	I	10*element ID + device code
	6	I	0
	7	I	0
	8	I	Dynamic load set ID
	9	I	Format code = 1
	10	I	Number of words per entry in next record = NWDS
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	R	Time
	2-NWDS	Mixed	Element force data
			See section 2.3.52 for details
			} repeat for each time step

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = $\begin{cases} 0 = x y \text{ output only} \\ 1 = \text{print} \\ 4 = \text{punch} \\ 5 = \text{print and punch} \end{cases}$

3. Format code = $\begin{cases} 1 = \text{real} \\ 2 = \text{real/imaginary} \\ 3 = \text{magnitude/phase} \end{cases}$

4. Approach code = 6

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.45.6 ØPNL2 (TABLE).

Description

Output nonlinear load requests (solution set, SØRT2, real).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code + 10*approach code
	2	I	2012
	3	I	0
	4	I	Subcase number
	5	I	10*point ID + device code
	6	I	0
	7	I	0
	8	I	Dynamic load set ID
	9	I	Format code = 1
	10	I	Number of words per entry in next record = 8
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	R	Time
	2	I	Point type
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3)
			}repeat for each time step

Notes

- Records 1 and 2 are repeated for each vector to be output.
- Device code = $\begin{cases} 0 & = \text{x y output only} \\ 1 & = \text{print} \\ 4 & = \text{punch} \\ 5 & = \text{print and punch} \end{cases}$
- Format code = $\begin{cases} 1 & = \text{real} \\ 2 & = \text{real/imaginary} \\ 3 & = \text{magnitude/phase} \end{cases}$
- Approach code = 6
- Point type = $\begin{cases} 1 & = \text{grid point} \\ 2 & = \text{scalar point} \\ 3 & = \text{extra point} \\ 4 & = \text{modal point} \end{cases}$
- Components (words 3-8 in even numbered records) which are not in the solution set are replaced by integer 1.

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK DESCRIPTIONS

2.3.45.7 ØUDV2 (TABLE).

Description

Output displacement vector requests (solution set, SØRT2, real).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code + 10*approach code
	2	I	{ 2015 = Displacement
	3	I	{ 2016 = Velocity
	4	I	{ 2017 = Acceleration
	5	I	0
	6	I	Subcase number
	7	I	10*point ID + device code
	8	I	0
	9	I	Dynamic load set ID
	10	I	Format code = 1
	11-50		Number of words per entry in next record = 8
	51-82	B	Not defined
	83-114	B	Title
	115-146	B	Subtitle
2	1	R	Label
	2	I	Time
	3-8	R	Point type
			R(T1), R(T2), R(T3), R(R1), R(R2), R(R3) } repeat for each time step

Notes

- Records 1 and 2 are repeated for each vector to be output.
- Device code = $\begin{cases} 0 = x y \text{ output only} \\ 1 = \text{print} \\ 4 = \text{punch} \\ 5 = \text{print and punch} \end{cases}$
- Format code = $\begin{cases} 1 = \text{real} \\ 2 = \text{real/imaginary} \\ 3 = \text{magnitude/phase} \end{cases}$
- Approach code = 6
- Point type = $\begin{cases} 1 = \text{grid point} \\ 2 = \text{scalar point} \\ 3 = \text{extra point} \\ 4 = \text{modal point} \end{cases}$
- Components (words 3-8 of even numbered records) which are not in the solution set are replaced by integer 1.

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.45.8 ØUHV2 (TABLE).

Description

Output displacement vector requests (solution set, SØRT2, real).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code + 10*approach code
	2	I	{ 2015 = Displacement
	3	I	{ 2016 = Velocity
	4	I	{ 2017 = Acceleration
	5	I	0
	6	I	Subcase number
	7	I	10*point ID + device code
	8	I	0
	9	I	0
	10	I	Dynamic load set ID
	11-50		Format code = 1
	51-82	B	Number of words per entry in next record = 8
	83-114	B	Not defined
	115-146	B	Title
			Subtitle
			Label
2	1	R	Time
	2	I	Point type
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3) } repeat for each time step

Notes

- Records 1 and 2 are repeated for each vector to be output.
- Device code = $\begin{cases} 0 = x y \text{ output only} \\ 1 = \text{print} \\ 4 = \text{punch} \\ 5 = \text{print and punch} \end{cases}$
- Format code = $\begin{cases} 1 = \text{real} \\ 2 = \text{real/imaginary} \\ 3 = \text{magnitude/phase} \end{cases}$
- Approach code = 6
- Point type = $\begin{cases} 1 = \text{grid point} \\ 2 = \text{scalar point} \\ 3 = \text{extra point} \\ 4 = \text{modal point} \end{cases}$
- Components (words 3-8 of even numbered records) which are not in the solution set are replaced by an integer 1.

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK DESCRIPTIONS

2.3.45.9 ØPPC2 (TABLE).

Description

Output load vector requests (p set, SØRT2, complex).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code + 10*approach code
	2	I	3002
	3	I	0
	4	I	Subcase number
	5	I	10*point ID + device code
	6	I	0
	7	I	0
	8	I	0
	9	I	Format code
	10	I	Number of words per entry in next record = 14
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	R	Frequency
	2	I	Point type
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3)
	9-14	R	I(T1), I(T2), I(T3), I(R1), I(R2), I(R3)

}repeat
for
each
frequency

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = $\begin{cases} 0 = x y \text{ output only} \\ 1 = \text{print} \\ 4 = \text{punch} \\ 5 = \text{print and punch} \end{cases}$

3. Format code = $\begin{cases} 1 = \text{real} \\ 2 = \text{real/imaginary} \\ 3 = \text{magnitude/phase} \end{cases}$

4. Approach code = 5

5. Point type = $\begin{cases} 1 = \text{grid point} \\ 2 = \text{scalar point} \\ 3 = \text{extra point} \\ 4 = \text{modal point} \end{cases}$

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.45.10 ØQPC2 (TABLE).

Description

Output forces of single-point constraint requests (p set, SØRT2, complex).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code + 10*approach code
	2	I	3003
	3	I	0
	4	I	Subcase number
	5	I	10*point ID + device code
	6	I	0
	7	I	0
	8	I	Load set ID
	9	I	Format code
	10	I	Number of words per entry in next record = 14
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	R	Frequency
	2	I	Point type
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3)
	9-14	R	I(T1), I(T2), I(T3), I(R1), I(R2), I(R3)

} repeat
for
each
frequency

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = $\begin{cases} 0 = x y \text{ output only} \\ 1 = \text{print} \\ 4 = \text{punch} \\ 5 = \text{print and punch} \end{cases}$

3. Format code = $\begin{cases} 1 = \text{real} \\ 2 = \text{real/imaginary} \\ 3 = \text{magnitude/phase} \end{cases}$

4. Approach code = 5

5. Point type = $\begin{cases} 1 = \text{gr point} \\ 2 = \text{scalar point} \\ 3 = \text{extra point} \\ 4 = \text{modal point} \end{cases}$

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK DESCRIPTIONS

2.3.45.11 ØUPVC2 (TABLE).

Description

Output displacement vector requests (p set, SQRT2, complex).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code + 10*approach code
	2	I	{ 3001 = Displacement
	3	I	{ 3010 = Velocity
	4	I	{ 3011 = Acceleration
	5	I	0
	6	I	Subcase number
	7	I	10*point ID + device code
	8	I	0
	9	I	0
	10	I	Load set ID
	11-50	I	Format code
	51-82	B	Number of words per entry in next record = 14
	83-114	B	Not defined
	115-146	B	Title
			Subtitle
			Label
2	1	R	Frequency
	2	I	Point type
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3)
	9-14	R	I(T1), I(T2), I(T3), I(R1), I(R2), I(R3)

repeat
for
each
frequency

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = { 0 = x y output only
1 = print
4 = punch
5 = print and punch

3. Format code = { 1 = real
2 = real/imaginary
3 = magnitude/phase

4. Approach code = 5

5. Point type = { 1 = grid point
2 = scalar point
3 = extra point
4 = modal point

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.45.12 ØESC2 (TABLE).

Description

Output element stress requests (SØRT2, complex).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code + 10*approach code
	2	I	3005
	3	I	Element type
	4	I	Subcase number
	5	I	10*Element ID + device code
	6	I	0
	7	I	0
	8	I	Load set ID
	9	I	Format code
	10	I	Number of words per entry in next record = NWDS
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	R	Frequency
	2-NWDS	Mixed	Element stress data
			See 2.3.51 for details
			} repeat for each frequency

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = $\begin{cases} 0 = x y \text{ output only} \\ 1 = \text{print} \\ 4 = \text{punch} \\ 5 = \text{print and punch} \end{cases}$

3. Format code = $\begin{cases} 1 = \text{real} \\ 2 = \text{real/imaginary} \\ 3 = \text{magnitude/phase} \end{cases}$

4. Approach code = 5

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK DESCRIPTIONS

2.3.45.13 ØEFC2 (TABLE).

Description

Output element force requests (SØRT2, complex).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code + 10*approach code
	2	I	3004
	3	I	Element type
	4	I	Subcase number
	5	I	10*element ID + device code
	6	I	0
	7	I	0
	8	I	Load set ID
	9	I	Format code
	10	I	Number of words per entry in next record = NWDS
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	R	Frequency
	2-NWDS	Mixed	Element force data
			See 2.3.52 for details
			} repeat for each frequency

Notes

Records 1 and 2 are repeated for each vector to be output.

2. Device code = $\begin{cases} 0 = x y \text{ output only} \\ 1 = \text{print} \\ 4 = \text{punch} \\ 5 = \text{print and punch} \end{cases}$

3. Format code = $\begin{cases} 1 = \text{real} \\ 2 = \text{real/imaginary} \\ 3 = \text{magnitude/phase} \end{cases}$

4. Approach code = 5

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.45.14 ØUDVC2 (TABLE).

Description

Output displacement vector requests (solution set, SØRT2, complex).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code + 10*approach code
	2	I	{ 3015 = Displacement
	3	I	{ 3016 = Velocity
	4	I	{ 3017 = Acceleration
	5	I	0
	6	I	Subcase Number
	7	I	10*point ID + device code
	8	I	0
	9	I	Dynamic load set ID
	10	I	Format code
	11-50		Number of words per entry in next record = 14
	51-82	B	Not defined
	83-114	B	Title
	115-146	B	Subtitle
			Label
2	1	R	Frequency
	2	I	Point type
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3)
	9-14	R	I(T1), I(T2), I(T3), I(R1), I(R2), I(R3)

} repeat
for
each
frequency

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = { 0 = x y output only
1 = print
4 = punch
5 = print and punch

3. Format code = { 1 = real
2 = real/imaginary
3 = magnitude/phase

4. Approach code = 5

5. Point type = { 1 = grid point
2 = scalar point
3 = extra point
4 = modal point

6. Components (words 3-14 of even numbered records) which are not in the solution set are replaced by integer 1.

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK DESCRIPTIONS

2.3.45.15 ØUHVC2 (TABLE).

Description

Output displacement vector requests (solution set, SØRT2, complex).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code + 10*approach code
	2	I	{ 3015 = Displacement
	3	I	{ 3016 = Velocity
	4	I	{ 3017 = Acceleration
	5	I	0
	6	I	Subcase number
	7	I	10*point ID + device code
	8	I	0
	9	I	0
	10	I	Dynamic load set ID
	11-50	I	Format code
	51-82	B	Number of words per entry in next record = 14
	83-114	B	Not defined
	115-146	B	Title
			Subtitle
			Label
2	1	R	Frequency
	2	I	Point type
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3)
	9-14	R	I(T1), I(T2), I(T3), I(R1), I(R2), I(R3)

repeat
for
each
frequency

Notes

- Records 1 and 2 are repeated for each vector to be output.
- Device code = $\begin{cases} 0 = \text{x y output only} \\ 1 = \text{print} \\ 4 = \text{punch} \\ 5 = \text{print and punch} \end{cases}$
- Format code = $\begin{cases} 1 = \text{real} \\ 2 = \text{real/imaginary} \\ 3 = \text{magnitude/phase} \end{cases}$
- Approach code = 5
- Point type = $\begin{cases} 1 = \text{grid point} \\ 2 = \text{scalar point} \\ 3 = \text{extra point} \\ 4 = \text{modal point} \end{cases}$
- Components (words 3-14 of even numbered records) which are not in the solution set are replaced by an integer 1.

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.45.16 ØUGV2 (TABLE)

Description

Output displacement vector requests (SØRT2, real)

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header Record
1	1	I	Device Code + approach code
	2	I	2001
	3	I	0
	4	I	Subcase number
	5	I	10*point ID + device code
	6	I	0
	7	I	0
	8	I	Dynamic load set ID
	9	I	Format code = 1
	10	I	Number of words per entry in next record = 8
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1		Subcase
	2	I	Point type
	3-8	R	R(T1), R(T2), R(T3), } repeat for each subcase R(R1), R(R2), R(R3)

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = $\begin{cases} 0 = x y \text{ output only} \\ 1 = \text{print} \\ 4 = \text{punch} \\ 5 = \text{print and punch} \end{cases}$

3. Format code = $\begin{cases} 1 = \text{real} \\ 2 = \text{real/imaginary} \\ 3 = \text{magnitude/phase} \end{cases}$

4. Approach code = 6

5. Point type = $\begin{cases} 1 = \text{grid point} \\ 2 = \text{scalar point} \\ 3 = \text{extra point} \\ 4 = \text{modal point} \end{cases}$

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK DESCRIPTIONS

2.3.45.17 ØPG2 (TABLE)

Description

Output load vector requests (g set, SØRT2, real)

Table Format

<u>Record</u>	<u>Word</u>	<u>Word</u>	<u>Item</u>
0			Header record
1	1	I	Device code + 10*approach code
	2	I	2002
	3	I	0
	4	I	Subcase number
	5	I	Load set ID
	6	I	0
	7	I	0
	8	I	0
	9	I	Format code
	10	I	Number of words per entry in next record = 8
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	Subcase number
	2	I	Print type
	3-8		R(T1), R(T2), R(T3), } repeated for each point R(R1), R(R2), R(R3)

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = $\begin{cases} 0 = x y \text{ output only} \\ 1 = \text{Point type} \\ 4 = \text{punch} \\ 5 = \text{print} \end{cases}$

3. Format code = $\begin{cases} 1 = \text{real} \\ 2 = \text{real/imaginary} \\ 3 = \text{magnitude/phase} \end{cases}$

4. Approach code = 6

5. Print type = $\begin{cases} 1 = \text{grid print} \\ 2 = \text{scalar point} \\ 3 = \text{extra point} \\ 4 = \text{modal point} \end{cases}$

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.45.18 ØQG2 (TABLE)

Description

Output forces of single-point constraint requests (g set, SØRT2, real)

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code + 10*approach code
	2	I	2003
	3	I	0
	4	I	Subcase number
	5	I	10*point ID + device code
	6	I	0
	7	I	0
	8	I	Load set ID
	9	I	Format code
	10	I	Number of words per entry in next record = 8
	11-50		not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1		Subcase number
	2	I	Point type
	3-8	R	R(T1), R(T2), R(T3), } repeated for each subcase R(R1), R(R2), R(R3) }

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = $\begin{cases} 0 = x y \text{ output only} \\ 1 = \text{print} \\ 4 = \text{punch} \\ 5 = \text{print and punch} \end{cases}$

3. Format code = $\begin{cases} 1 = \text{real} \\ 2 = \text{real/imaginary} \\ 3 = \text{magnitude/phase} \end{cases}$

4. Approach code = 6

5. Point type = $\begin{cases} 1 = \text{grid point} \\ 2 = \text{sc ar point} \\ 3 = \text{ext a point} \\ 4 = \text{modal point} \end{cases}$

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK DESCRIPTIONS

2.3.45.19 IQP2 (TABLE)

Description

Output forces of single-point constraint requests (p set, SØRT2)

Table Format

See format of ØQPC2 table - section 2.3.45.10.

2.3.45.20 IPHIP2 (TABLE)

Description

Output displacement vector requests (p set, SØRT2, complex).

Table Format

See format of ØUPVC2 table - section 2.3.45.11.

2.3.45.21 IES2 (TABLE)

Description

Output element stress requests (SØRT2, complex).

Table Format

See format of ØESC2 table - section 2.3.45.12.

2.3.45.22 IEF2 (TABLE)

Description

Output element force requests (SØRT2, complex).

Table Format

See format of ØEFC2 table - section 2.3.45.13.

2.3.45.23 ØPPCB (TABLE)

Description

Output load vector requests (P set, SØRT2, complex).

Table Format

See format of ØPPC2 table - section 2.3.45.9.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.45.24 ØPPB (TABLE)

Description

Output load vector requests (p set, SØRT2, real).

Table Format

See format of ØPP2 table - section 2.3.45.1.

2.3.45.25 IQP2 (TABLE)

Description

Output forces of single-point constraint (p set, SØRT2, real).

Table Format

See format of ØQP2 table - section 2.3.45.2.

2.3.45.26 HØPP2 (TABLE)

Description

See description and format for ØPP2 table - section 2.3.45.1.

2.3.45.27 HØQP2 (TABLE)

Description

See description and format for ØQP2 table - section 2.3.45.2.

2.3.45.28 HØUPV2 (TABLE)

Description

See description and format of ØUPV2 table - section 2.3.45.3.

2.3.45.29 HØEF2 (TABLE)

Description

See description and format of ØEF2 table - section 2.3.45.5.

2.3.45.30 HØUDV2 (TABLE)

Description

Output temperature vector requests (solution set, SØRT2, real).

Table Format

See format at ØUDV2 table - section 2.3.45.7.

DATA BLOCK DESCRIPTIONS

2.3.45.31 HØPNL2 (TABLE)

Description

See description and format of ØPNL2 table - section 2.3.45.6.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.46 Data Blocks Output from Module XYTRAN

2.3.46.1 XYPLTT (TABLE).

Description

Output plot request data in form for direct plotting of SØRT2 Transient Response output.

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I-R	Subcase ID or if a Random Response problem, the Mean Response
	2	I	Frame number
	3	I	Curve number
	4	I	Point ID or element ID
	5	I	Component number
	6	I	Vector number (1, 2, ... 11)
	7	I	1--Graph uses top half of frame 0--Graph uses full frame -1--Graph uses lower half of frame
	8	I	0--Axis, tics, labels, curves, etc. have been drawn and this curve is to be scaled and plotted identically as last except for curve symbols. 1--Axis, tics, labels, scaling, etc. are to be performed or computed and if word 7 of this record = 0 or 1, a skip to new frame is to be made.
	9	I	Number of blank frames between frames (frame-skip)
	10		Minimum X-increment
	11	R	XMIN
	12	R	XMAX
	13	R	YMIN
	14	R	YMAX
	15	R	graph
	16	R	Actual value of first tic
	17	R	Actual increment to successive tics
	18	R	Actual maximum frame limit
	19	I	Maximum number of digits in any print-value
	20	I	+ or - power for print values
	21	I	Total number of tics to print this edge
	22	I	Value print skip 0,1,2,3---
	23	I	Not used
	24	R	
	25	R	
	26	I	
	27	I	
	28	I	
	29	I	
	30	I	

x-direction
tics

} Same as 15 through 22
But for y-direction tics

DATA BLOCK DESCRIPTIONS

Record	Word	Type	Item
	31	I	Top edge tics
	32	I	Bottom edge tics
	33	I	Left edge tics
	34	I	Right edge tics
	35	I	0--x-direction is linear
			Greater than 0--number of cycles and x-direction is logarithmic
	36	I	0--y-direction is linear
			Greater than 0--number of cycles and y-direction is logarithmic
	37	I	0--no x-axis
			1--draw x-axis
	38	R	x-axis y-intercept
	39	I	0--no y-axis
			1--draw y-axis
	40	R	y-axis x-intercept
	41	I	Less than 0 -- Plot symbol for each curve point. Select symbol corresponding to curve number in word 3 of this record.
			Equal to 0 -- Connect points by lines where points are continuous i.e., (no integer 1 pairs).
			Greater than 0 -- do both of above.
	42		} Not used
	.		
	.		
	.		
	50		
	51	B	Title (32 words)
	.	B	Subtitle (32 words)
	.	B	Label (32 words)
	.	B	Curve title (32 words)
	.	B	x-axis Title (32 words)
	242	B	y-axis Title (32 words)
	243		} Not used
	.		
	.		
	281	I	
	282	I	
	283	I	Paper plot frame number
	284	I	Pen color (not implemented)
	285	I	Pensize
	286	R	Plotter and model (Plotter*63536+Model+100)
	287	R	Inches paper x-direction
	288	I	Inches paper y-direction
	289	I	Camera for SC4020
			Print flag 0 = no, 1 = yes
			Plot flag 0 = no, 1 = plotter, -1 = paper, 2 = plotter and paper
	290	I	Punch flag 0 = no, 1 = yes
	291	R	x-min of all data
	292	R	x-max of all data
	293	R	y-min within x-limits of graph
	294	R	x-value at this y-min
	295	R	y-max within x-limits of graph
	296	R	x-value at this y-max
	297	R	y-min for all data
	298	R	x-value at this y-min
	299	R	y-max for all data
	300	R	x-value at this y-max

DATA BLOCK AND TABLE DESCRIPTIONS

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
2	1	I	1 } Always is present
	2	I	
	3	R	x-value } coordinate pair
	4	R	y-value } repeats for all pairs plotted

Notes

1. Records 1 and 2 repeat for each curve plotted.
2. Even numbered records will contain integer 1 pairs to indicate where curve has moved outside of graph limits.

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK DESCRIPTIONS

2.3.46.2 XYPLTFA (TABLE).

Description

Identical to XYPLTT, for Frequency Response plots (solution set).

2.3.46.3 XYPLTF (TABLE).

Description

Identical to XYPLTT, for Frequency Response plots.

2.3.46.4 XYPLTR (TABLE).

Description

Identical to XYPLTT, for Random Response plots.

2.3.46.5 XYPLTTA (TABLE).

Description

Identical to XYPLTT, for Transient Response plots (solution set).

2.3.46.6 HXYPLTT (TABLE)

Description

Identical to XYPLTT, for Heat Transient Response plots.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.47 Data Blocks Output From Module RANDØM

2.3.47.1 PSDF (TABLE)

Description

Power Spectral Density Table.

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	50
	2	I	Code for data type DISP = 2001 VELØ = 2010 ACCE = 2011 LØAD = 2002 SPCF = 2003 ELFØ = 2004 STRE = 2005
	3	I	4001
	4	I	0
	5	I	Point or element ID times 10
	6	I	Component ID + 2
	7	I	0
	8	R	Mean response
	9	I	Number of zero crossings (NO)
	10	I	2
	11-50	I	0
	51-146	B	Title, subtitle, label
2	1	R	Frequency
	2	R	Power spectral density

Notes

1. Words 1 and 2 of record 2 are repeated for each frequency.
2. Records 1 and 2 are repeated for each power spectral density request.

Table Trailer

Words 1-5 = zero.

Word 6 = number of requests.

DATA BLOCK DESCRIPTIONS

2.3.47.2 AUTØ (TABLE).

Description

Autocorrelation function table.

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	50
	2	I	Code for data type DISP = 2001 VELØ = 2010 ACCE = 2011 LØAD = 2002 SPCF = 2003 ELFØ = 2004 STRE = 2005
	3	I	4002
	4	I	0
	5	I	Point or element ID times 10
	6	I	Component ID + 2
	7	I	0
	8	R	Mean response
	9	I	Number of zero crossings (NO)
	10	I	2
	11-50	I	0
	51-146	B	Title, subtitle, label
2	1	R	TAU
	2	R	Auto correlation function

Notes

1. Words 1 and 2 of record 2 are repeated for each TAU.
2. Records 1 and 2 are repeated for each autocorrelation request.

Table Trailer

Words 1-5 = zero.

Word 6 = number of requests.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.48 Data Blocks Output From Module TRD.

2.3.48.1 UDVT (MATRIX)

Description

$[u_d^t]$ - Displacement, velocity, and acceleration vector matrix in a transient analysis problem - d set.

Matrix Trailer

Number of columns = three times the number of output time steps
Number of rows = d
Form = rectangular
Type = real single precision

2.3.48.2 UHVT (MATRIX)

Description

$[u_h^t]$ - Modal transient solution vectors - h set.

Matrix Trailer

Number of columns = three times the number of output times
Number of rows = h
Form = rectangular
Type = real single precision

2.3.49 Data Blocks Output From Module GKAM

2.3.49.1 MHH (MATRIX)

Description

$[m_{hh}]$ - Modal mass matrix - h set.

Matrix Trailer

Number of columns = h
 Number of rows = h
 Form = symmetric
 Type = real double precision

2.3.49.2 BHH (MATRIX)

Description

$[b_{hh}]$ - Modal damping matrix - h set.

Matrix Trailer

Number of columns = h
 Number of rows = h
 Form = symmetric
 Type = real double precision

2.3.49.3 KHH (MATRIX)

Description

$[k_{hh}]$ - Modal stiffness matrix - h set.

Matrix Trailer

Number of columns = h
 Number of rows = h
 Form = symmetric
 Type = real double precision

2.3.49.4 PHIDH (MATRIX)

Description

$[\phi_{dh}]$ - Transformation matrix from d set to modal coordinates.

Matrix Trailer

Number of columns = h
 Number of rows = d
 Form = rectangular
 Type = real single precision

2.3.50 Data Blocks Output From Module DDR1

2.3.50.1 CPHID (MATRIX)

Description

$[\phi_d]$ - Complex eigenvector matrix transformed from modal to physical coordinates.

Matrix Trailer

Number of columns = number of eigenvectors
 Number of rows = d
 Form = rectangular
 Type = complex single precision

2.3.50.2 UDV1F (MATRIX)

Description

$[u_d^f]$ - Displacement vectors matrix in a frequency response problem - d set.

Matrix Trailer

Number of columns = number of frequencies times number of loads
 Number of rows = d
 Form = rectangular
 Type = complex single precision

2.3.50.3 UDV1T (MATRIX)

Description

$[u_d^t]$ - Displacement vectors matrix in a transient response problem - d set.

Matrix Trailer

Number of columns = number of output times multiplied by 3
 Number of rows = d
 Form = rectangular
 Type = real single precision

DATA BLOCK DESCRIPTIONS

2.3.51 Element Stress Output Data Description.

Note particular data block description (e.g., ØES1, ØESB1) for contents of word 1 for each element.

Element		Real Element Stresses		Complex Element Stresses		
Type	Name	Word or Component	Item	Word or Component	Item	Real Imag.
1	CRØD	2	Axial	2	Axial	R
		3	Axial Safety Margin *	3	Axial	I
		4	Torsional	4	Torsional	R
		5	Torsional Safety Margin*	5	Torsional	I
2	CBEAM	2	SA1	Undefined		
		3	SA2			
		4	SA3			
		5	Axial			
		6	SA-maximum			
		7	SA-minimum			
		8	Safety Margin in Tension*			
		9	SB1			
		10	SB2			
		11	SB3			
		12	SB-maximum			
		13	SB-minimum			
		14	Safety Margin in Comp*			
3	CTUBE	Same as CRØD		Same as CRØD		
4	CSHEAR	2	Maximum Shear	2	Maximum Shear	R
		3	Average Shear	3	Maximum Shear	I
		4	Safety Margin*	4	Average Shear	R
5	CTWIST			5	Average Shear	I
		2	Maximum	2	Maximum	R
		3	Average	3	Maximum	I
		4	Safety Margin*	4	Average	R
6	CTRIA1			5	Average	I
		2	Z1 = Fibre Distance 1	2	Z1 = Fibre Distance 1	
		3	Normal-x at Z1	3	Normal-x at 1	R
		4	Normal-y at Z1	4	Normal-x at 1	I
		5	Shear-xy at Z1	5	Normal-y at 1	R
		6	O-Shear Angle at Z1	6	Normal-y at 1	I
		7	Major-Principal at Z1	7	Shear-xy at 1	R
		8	Minor-Principal at Z1	8	Shear-xy at 1	I
		9	Max-Shear at Z1	9	Z2 = Fibre Distance 2	
		10	Z2 = Fibre Distance 2	10	Normal-x at 2	R
		11	Normal-x at Z2	11	Normal-x at 2	I
		12	Normal-y at Z2	12	Normal-y at 2	R
		13	Shear-xy at Z2	13	Normal-y at 2	I
		14	O-Shear Angle at Z2	14	Shear-xy at 2	R
		15	Major-Principal at Z2	15	Shear-xy at 2	I
		16	Minor-Principal at Z2			
		17	Maximum-Shear at Z2			

DATA BLOCK AND TABLE DESCRIPTIONS

Element		Real Element Stresses		Complex Element Stresses		Real Imag.
Type	Name	Word or Component	Item	Word or Component	Item	
7	CTRBSC		Note CTRIA1		Note CTRIA1	
8	CTRPLT		Note CTRIA1		Note CTRIA1	
9	CTRMEM	2	Normal-x	2	Normal-x	R
		3	Normal-y	3	Normal-x	I
		4	Shear-xy	4	Normal-y	R
		5	O-Shear Angle	5	Normal-y	I
		6	Major-Principal	6	Shear-xy	R
		7	Minor-Principal	7	Shear-xy	I
		8	Maximum Shear			
10	CØNRØD		Note CRØD		Note CRØD	
11	CELAS1	2	Stress	2	Stress	R
				3	Stress	I
12	CELAS2	2	Stress	2	Stress	R
				3	Stress	I
13	CELAS3	2	Stress	2	Stress	R
				3	Stress	I
14	CELAS4		Undefined		Undefined	
15	CQDPLT		Note CTRIA1		Note CTRIA1	
16	CQDMEM		Note CTRMEM		Note CTRMEM	
17	CTRIA2		Note CTRIA1		Note CTRIA1	
18	CQUAD2		Note CTRIA1		Note CTRIA1	
19	CQUAD1		Note CTRIA1		Note CTRIA1	
20	CDAMP1		Undefined		Undefined	
21	CDAMP2		Undefined		Undefined	
22	CDAMP3		Undefined		Undefined	
23	CDAMP4		Undefined		Undefined	
24	CVISC		Undefined		Undefined	
25	CMASS1		Undefined		Undefined	
26	CMASS2		Undefined		Undefined	
27	CMASS3		Undefined		Undefined	
28	CMASS4		Undefined		Undefined	
29	CØNM1		Undefined		Undefined	
30	CØNM2		Undefined		Undefined	
31	CPLØTEL		Undefined		Undefined	

DATA BLOCK DESCRIPTIONS

Element		Real Element Stresses		Complex Element Stresses		
Type	Name	Word or Component	Item	Word or Component	Item	Real Imag.
34	CBAR	2	SA1	2	SA1	R
		3	SA2	3	SA2	R
		4	SA3	4	SA3	R
		5	SA4	5	SA4	R
		6	Axial	6	Axial	R
		7	SA-maximum	7	SA1	I
		8	SA-minimum	8	SA2	I
		9	Safety Margin in Tension*	9	SA3	I
		10	SB1	10	SA4	I
		11	SB2	11	Axial	I
		12	SB3	12	SB1	R
		13	SB4	13	SB2	R
		14	SB-maximum	14	SB3	R
		15	SB-minimum	15	SB4	R
		16	Safety Margin in Comp*	16	SB1	I
				17	SB2	I
				18	SB3	I
				19	SB4	I
35	CCONEAX	2	Harmonic or point angle		Undefined	
		3	Z1 = Fibre Distance 1			
		4	Normal-u at 1			
		5	Normal-v at 1			
		6	Shear-uv at 1			
		7	O-Shear Angle at 1			
		8	Major-Principal at 1			
		9	Minor-Principal at 1			
		10	Maximum Shear at 1			
		11	Z2 = Fibre Distance 2			
		12	Normal-u at 2			
		13	Normal-v at 2			
		14	Shear-uv at 2			
		15	O-Shear Angle at 2			
		16	Major-Principal at 2			
		17	Minor-Principal at 2			
		18	Maximum-Shear at 2			
36	CTRIARG	2	Radial (x)		Undefined	
		3	Circum (Theta)			
		4	Axial (z)			
		5	Shear (zx)			
37	CTRAPRG	2	Radial (x) at 1		Undefined	
		3	Circum (Theta) at 1			
		4	Axial (z) at 1			
		5	Shear (zx) at 1			
		6	Radial (x) at 2			
		7	Circum (Theta) at 2			
		8	Axial (z) at 2			
		9	Shear (zx) at 2			
		10	Radial (x) at 3			
		11	Circum (Theta) at 3			
		12	Axial (z) at 3			
		13	Shear (zx) at 3			
		14	Radial (x) at 4			
		15	Circum (Theta) at 4			
		16	Axial (z) at 4			
		17	Shear (zx) at 4			
		18	Radial (x) at 5			

DATA BLOCK AND TABLE DESCRIPTIONS

Element		Real Element Stresses			Complex Element Stresses		
Type	Name	Word or Component	Item		Word or Component	Item	Real Imag.
37 cont'd.		19	Circum(Theta)	at 5			
		20	Axial (z)	at 5			
		21	Shear (zx)	at 5			
38	CTØRDRG	2	Mem.-Tagen.	at 1		Undefined	
		3	Mem.-Circum.	at 1			
		4	Flex.-Tangen.	at 1			
		5	Flex.-Circum.	at 1			
		6	Shear-Force	at 1			
		7	Mem.-Tangen.	at 2			
		8	Mem.-Circum.	at 2			
		9	Flex.-Tangen.	at 2			
		10	Flex.-Circum.	at 2			
		11	Shear-Force	at 2			
		12	Mem.-Tangen.	at 3			
		13	Mem.-Circum.	at 3			
		14	Flex.-Tangen.	at 3			
		15	Flex.-Circum.	at 3			
		16	Shear-Force	at 3			
53 - 61	CDUM1 thru CDUM9	2	S1		2	S1	R
		3	S2		3	S2	R
		4	S3		4	S3	R
		5	S4		5	S4	R
		6	S5		6	S5	R
		7	S6		7	S6	R
		8	S7		8	S7	R
		9	S8		9	S8	R
		10	S9		10	S9	R
					11	S1	I
					12	S2	I
					13	S3	I
					14	S4	I
					15	S5	I
					16	S6	I
					17	S7	I
					18	S8	I
					19	S9	I
62	CQDMEM1		Note CTRMEM			Note CTRMEM	
63	CQDMEM2		Note CTRMEM			Undefined	
65	CIHEX1*	2	External Grid point ID		2	External grid point ID	
		3	Normal - x		3	Normal - x	R
		4	Shear - xy		4	Normal - y	R
		5	First principal		5	Normal - z	R
		6	First principal x cosine		6	Snear - xy	R
		7	Second principal x cosine		7	Shear - yz	R
		8	Third principal x cosine		8	Shear - zx	R

DATA BLOCK DESCRIPTIONS

Element		Real Element Stresses		Complex Element Stresses	
Type	Name	Word or Component	Item	Word or Component	Real Imag.
65 cont'd.		9	Mean stress	9	Normal - x I
		10	Octahedral shear stress	10	Normal - y I
		11	Normal - y	11	Normal - z I
		12	Shear - yz	12	Shear - xy I
		13	Second principal	13	Shear - yz I
		14	First principal y cosine	14	Shear - zx I
		15	Second principal y cosine		
		16	Third principal y cosine		
		17	Normal - z		
		18	Shear - zx		
		19	Third principal		
		20	First principal z cosine		
		21	Second principal z cosine		
		22	Third principal z cosine		
66	CIHEX2*	Note CIHEX1		Note CIHEX1	
67	CIHEX3*	2	First external grid point ID	2	First external grid point ID
		3	Normal - x	3	Normal - x R
		4	Shear - xy	4	Normal - y R
		5	First principal	5	Normal - z R
		6	First principal x cosine	6	Shear - xy R
		7	Second principal x cosine	7	Shear - yz R
		8	Third principal x cosine	8	Shear - zx R
		9	Mean stress	9	Second external grid point ID
		10	Octahedral shear stress	10	Normal - x I
		11	Second external grid point ID	11	Normal - y I
		12	Normal - y	12	Normal - z I
		13	Shear - yz	13	Shear - xy I
		14	Second principal	14	Shear - yz I
		15	First principal y cosine	15	Shear - zx I
		16	Second principal y cosine		
		17	Third principal y cosine		
		18	Normal - z		
		19	Shear - zx		
		20	Third principal		
		21	First principal z cosine		
		22	Second principal z cosine		
		23	Third principal z cosine		

*The stresses are repeated for each stress point within each element.

DATA BLOCK AND TABLE DESCRIPTIONS

Element		Real Element Stresses		Complex Element Stresses		
Type	Name	Word or Component	Item	Word or Component	Item	Real Imag.
68	CTRIAAX	2	Harmonic or Point Angle	Undefined		
		3	Radial (R)			
		4	Axial (Z)			
		5	Circum (Theta-T)			
		6	Shear (ZR)			
		7	Shear (RT)			
		8	Shear (ZT)			
69	CTRAPAX	2	Harmonic or Point Angle	Undefined		
		3	Radial (R)			
		4	Axial (Z)			
		5	Circum (Theta-T)			
		6	Shear (ZR)			
		7	Shear (RT)			
		8	Shear (ZT)			

DATA BLOCK DESCRIPTIONS

2.3.52 Element Force Output Data Description.

Note particular data block description (e.g., ØEF1, ØEFB1) for contents word 1 for each element.

Element		Real Element Forces		Complex Element Forces		Real Imag.
Type	Name	Word or Component	Item	Word or Component	Item	
1	CRØD	2	Axial Force	2	Axial Force	R
		3	Torque	3	Axial Force	I
				4	Torque	R
				5	Torque	I
2	CBEAM	2	Bend-Mom A1	Undefined		
		3	Bend-Mom A2			
		4	Bend-Mom B1			
		5	Bend-Mom B2			
		6	Shear-1			
		7	Shear-2			
		8	Axial Force			
		9	Torque			
3	CTUBE	Same as CRØD		Same as CRØD		
4	CSHEAR	2	Force Pts 1,3	2	Force Pts 1,3	R
		3	Force Pts 2,4	3	Force Pts 1,3	I
				4	Force Pts 2,4	R
				5	Force Pts 2,4	I
5	CTWIST	2	Moment Pts 1,3	2	Moment Pts 1,3	R
		3	Moment Pts 2,4	3	Moment Pts 1,3	I
				4	Moment Pts 2,4	R
				5	Moment Pts 2,4	I
6	CTRIA1	2	Bend-Mom-x	2	Bend-Mom-x	R
		3	Bend-Mom-y	3	Bend-Mom-y	R
		4	Twist-Moment	4	Twist-Moment	R
		5	Shear-x	5	Shear-x	R
		6	Shear-y	6	Shear-y	R
				7	Bend-Mom-x	I
7	CTRBSC	Same as CTRIA1		8	Bend-Mom-y	I
				9	Twist-Moment	I
				10	Shear-x	I
				11	Shear-y	I
				Same as CTRIA1		
8	CTRPLT	Same as CTRIA1		Same as CTRIA1		
9	CTRMEM	Undefined		Undefined		
10	CØNRØD	Same as CRØD		Same as CRØD		
11	CELAS1	2	Force	2	Force	R
				3	Force	I
12	CELAS2	2	Force	2	Force	R
				3	Force	I
13	CELAS3	2	Force	2	Force	R
				3	Force	I

DATA BLOCK AND TABLE DESCRIPTIONS

Element		Real Element Forces		Complex Element Forces		
Type	Name	Word or Component	Item	Word or Component	Item	Real Imag.
14	CELAS4	2	Force	2 3	Force Force	R I
15	CQDPLT		Note CTRIA1		Note CTRIA1	
16	CQDMEM		Undefined		Undefined	
17	CTRIA2		Note CTRIA1		Note CTRIA1	
18	CQUAD2		Note CTRIA1		Note CTRIA1	
19	CQUAD1		Note CTRIA1		Note CTRIA1	
20	CDAMP1		Undefined		Undefined	
21	CDAMP2		Undefined		Undefined	
22	CDAMP3		Undefined		Undefined	
23	CDAMP4		Undefined		Undefined	
24	CVISC		Undefined		Undefined	
25	CMASS1		Undefined		Undefined	
26	CMASS2		Undefined		Undefined	
27	CMASS3		Undefined		Undefined	
28	CMASS4		Undefined		Undefined	
29	CØNM1		Undefined		Undefined	
30	CØNM2		Undefined		Undefined	
31	CPLØTEL		Undefined		Undefined	
34	CBAR	2 3 4 5 6 7 8 9	Bend-Mom A1 Bend-Mom A2 Bend-Mom B1 Bend-Mom B2 Shear-1 Shear-2 Axial Force Torque	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17	Bend-Mom A1 Bend-Mom A2 Bend-Mom B1 Bend-Mom B2 Shear-1 Shear-2 Axial Force Torque Bend-Mom A1 Bend-Mom A2 Bend-Mom B1 Bend-Mom B2 Shear-1 Shear-2 Axial Force Torque	R R R R R R R R I I I I I I I I

DATA BLOCK DESCRIPTIONS

Element		Real Element Forces		Complex Element Forces		
Type	Name	Word or Component	Item	Word or Component	Item	Real Imag.
35	CCØNEAX	2	Harmonic or point angle	Undefined		
		3	Bend-Mom u			
		4	Bend-Mom v			
		5	Twist-Moment			
		6	Shear u			
		7	Shear v			
36	CTRIARG	2	Radial (x) at 1	Undefined		
		3	Circum (Theta) at 1			
		4	Axial (z) at 1			
		5	Radial (x) at 2			
		6	Circum (Theta) at 2			
		7	Axial (z) at 2			
		8	Radial (x) at 3			
		9	Circum (Theta) at 3			
		10	Axial (z) at 3			
37	CTRAPRG	2	Radial (x) at 1	Undefined		
		3	Circum (Theta) at 1			
		4	Axial (z) at 1			
		5	Radial (x) at 2			
		6	Circum (Theta) at 2			
		7	Axial (z) at 2			
		8	Radial (x) at 3			
		9	Circum (Theta) at 3			
		10	Axial (z) at 3			
		11	Radial (x) at 4			
		12	Circum (Theta) at 4			
		13	Axial (z) at 4			
38	CTØRDRG	2	Radial (x) at 1	Undefined		
		3	Circum (Theta) at 1			
		4	Axial (z) at 1			
		5	Moment (zx) at 1			
		6	Direct Strain at 1			
		7	Curvature at 1			
		8	Radial (x) at 2			
		9	Circum (Theta) at 2			
		10	Axial (z) at 2			
		11	Moment (zx) at 2			
		12	Direct Strain at 2			
		13	Curvature at 2			
53-61	CDUM1 thru CDUM9	2	F1	2	F1	R
		3	F2	3	F2	R
		4	F3	4	F3	R
		5	F4	5	F4	R
		6	F5	6	F5	R
		7	F6	7	F6	R
		8	F7	8	F7	R
		9	F8	9	F8	R
		10	F9	10	F9	R
				11	F1	I
				12	F2	I
				13	F3	I
				14	F4	I

DATA BLOCK AND TABLE DESCRIPTIONS

Element		Real Element Forces		Complex Element Forces		Real
Type	Name	Word or Component	Item	Word or Component	Item	Imag.
53-61 cont'd.				15	F5	I
				16	F6	I
				17	F7	I
				18	F8	I
				19	F9	I
62	CQDMEM1	Undefined		Undefined		
63	CQDMEM2	2	Force 4 to 1	Undefined		
		3	Force 2 to 1			
		4	Force 1 to 2			
		5	Force 3 to 2			
		6	Force 2 to 3			
		7	Force 4 to 3			
		8	Force 3 to 4			
		9	Force 1 to 4			
		10	Kick Force on 1			
		11	Shear-12			
		12	Kick Force on 2			
		13	Shear-23			
		14	Kick Force on 3			
		15	Shear-34			
		16	Kick Force on 4			
		17	Shear-41			
65	CIHEX1	Undefined		Undefined		
66	CIHEX2	Undefined		Undefined		
67	CIHEX3	Undefined		Undefined		
68	CTRIAAX	2	Harmonic or Point Angle	Undefined		
		3	Radial (R) at 1			
		4	Circum (Theta-T)at 1			
		5	Axial (Z) at 1			
		6	Radial (R) at 2			
		7	Circum (Theta-T)at 2			
		8	Axial (Z) at 2			
		9	Radial (R) at 3			
		10	Circum (Theat-T)at 3			
		11	Axial (Z) at 3			
69	CTRAPAX	2	Harmonic or Point Angle			
		3	Radial (R) at 1			
		4	Circum (Theta-T)at 1			
		5	Axial (Z) at 1			
		6	Radial (R) at 2			
		7	Circum (Theta-T)at 2			
		8	Axial (Z) at 2			
		9	Radial (R) at 3			
		10	Circum (Theta-T)at 3			
		11	Axial (Z) at 3			
		12	Radial (R) at 4			
		13	Circum (Theta-T)at 4			
		14	Axial (Z) at 4			

2.3.53 Data Blocks Output From Module DDR2

2.3.53.1 UEVF (MATRIX)

Description

$[u_e^f]$ - Displacements at the extra points for a frequency response problem.

Matrix Trailer

Number of columns = number of frequencies times number of loads
 Number of rows = e
 Form = rectangular
 Type = single precision

2.3.53.2 PAF (MATRIX)

Description

$[P_a^f]$ - Equivalent load vector for mode acceleration computations in a frequency response problem - a set.

Matrix Trailer

Number of columns = number of frequencies times number of loads
 Number of rows = d
 Form = rectangular
 Type = single precision

2.3.53.3 UDV2F (MATRIX)

Description

$[u_d^{fa}]$ - Mode accelerated displacement vectors for a frequency response problem.

Matrix Trailer

Number of columns = number of frequencies times number of loads
 Number of rows = d
 Form = rectangular
 Type = complex single precision

2.3.53.4 UEVT (MATRIX)

Description

$[u_e^t]$ - Displacement at the extra points for a transient analysis problem.

DATA BLOCK AND TABLE DESCRIPTIONS

Matrix Trailer

Number of columns = number of output times multiplied by 3
Number of rows = e
Form = rectangular
Type = real single precision

2.3.53.5 PAT (MATRIX)

Description

$[P_a^t]$ - Equivalent load vector for mode acceleration in a transient analysis problem.

Matrix Trailer

Number of columns - number of output times multiplied by 3
Number of rows - d
Form - rectangular
Type - real single precision

2.3.53.6 UDV2T (MATRIX)

Description

$[u_d^{ta}]$ - Mode accelerated displacement vectors for a transient analysis problem.

Matrix Trailer

Number of columns = number of output times multiplied by 3
Number of rows = d
Form = rectangular
Type = real single precision

DATA BLOCK DESCRIPTIONS

2.3.54 Data Blocks Output from Module BMG

2.3.54.1 BDPØØL (TABLE)

Description

Hydroelastic boundary matrix tables.

Table Format

Same format as the MATPØØL data block DMIG card images.

Notes: The names of the matrices are KBFL and ABFL

Table Trailer

IFP format, 6 words containing 96 pointer bits for use by subroutines PRELØC and LØCATE.

2.3.55 Data Blocks Output from Module PLTTRAN

2.3.55.1 SIP (TABLE)

Description

Same format as data block SIL. If fluid points are present each fluid point, i , will cause the next SIL value to have a value:

$$SIL(i+1) = SIL(i) + 1$$

The SIP data will be:

$$SIP(i+1) = SIP(i) + 6$$

where i is a fluid point.

2.3.55.2 BGPDP (TABLE)

Description

Same format as data block BGPDT except fluid points have the value -2 in the fields corresponding to coordinate system identification numbers.

2.3.55.3 HSIP (TABLE)

Description

See description and format of SIP table - section 2.3.55.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.56 Data Blocks Output from Module RMG

2.3.56.1 HRGG (MATRIX)

Radiation matrix for heat transfer.

2.3.56.2 HQGE (MATRIX)

The first record is a list of the HBDY elements. Subsequent records are in matrix format, one column per element, defining the temperature-flux relationship between points and elements.

2.3.56.3 HKGG (MATRIX)

In heat transfer, KGG defines the temperature-heat flow relationships between points.

2.3.57 Data Blocks Output from Module TRLG

2.3.57.1 PPT or HPPØ (MATRIX)

Description

[P_{po}] - Load versus time matrix, each column is a complete load vector at a specific output time.

Matrix Trailer

Number of columns	=	number of output times
Number of rows	=	p
Form	=	rectangular
Type	=	real single precision

2.3.57.2 PST or HPSØ (MATRIX)

Description

[P_{so}] - Load versus output time matrix - s set.

Matrix Trailer

Number of columns	=	number of output times
Number of rows	=	s
Form	=	rectangular
Type	=	real single precision

DATA BLOCK DESCRIPTIONS

2.3.57.3 PDT or HPDØ (MATRIX)

Description

[P_{do}] - Load versus output time matrix - d set.

Matrix Trailer

Number of columns = number of output times
Number of rows = d
Form = rectangular
Type = real single precision

2.3.57.4 PD or HPDT (MATRIX)

Description

[P_{dt}] - Load versus all times matrix - d set.

Matrix Trailer

Number of columns = number of time steps
Number of rows = d
Form = rectangular
Type = real single precision

2.3.57.5 PH (MATRIX)

Description

[P_{ht}] - Load versus all times matrix - h set.

Matrix Trailer

Number of columns = number of time steps
Number of rows = h
Form = rectangular
Type = real single precision

2.3.57.6 TØL (TABLE)

List of output time values

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.58 Data Blocks Output from Module TRHT

2.3.58.1 HUDVT (MATRIX)

Description

$[HU_d^I]$ - Temperature and velocity vector matrix in a transient analysis problem - d set.

Matrix Trailer

Number of columns = three times number of output time steps.
Number of rows = d
Form = rectangular
Type = real

2.3.58.2 HPNLD (MATRIX)

Description

$[HP_d^{n2}]$ - Nonlinear loads in transient problem - d set.

Matrix Trailer

Number of columns = number of output times
Number of rows = d
Form = rectangular
Type = real

2.3.59 Data Blocks Output from Module SSGHT

2.3.59.1 HUGV (MATRIX)

Description

$[HU_g]$ - Temperature vector matrix giving temperatures in the g set.

Matrix Trailer

Number of columns = number of subcases in CASECC
Number of rows = g
Form = rectangular
Type = real

DATA BLOCK DESCRIPTIONS

2.3.59.2 HQG (MATRIX)

Description

[Q_g] - Heat flux matrix due to single point forces of constraint - g set.

Matrix Trailer

Number of columns = number of subcases in CASECC
Number of rows = g
Form = rectangular
Type = real

2.3.59.3 HRULV (MATRIX)

Description

[SP_ℓ] - Residual heat fluxes for the ℓ set.

Matrix Trailer

Number of columns = number of subcases in CASECC
Number of rows = ℓ
Form = rectangular
Type = real

2.3.60 Data Blocks Output from Module SDRHT

2.3.60.1 HØEFIX (TABLE)

Description

See description and format of ØEF1 - section 2.3.28.19.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.61 Data Blocks Output from Module GPCYC

2.3.61.1 CYCD (TABLE)

Description

Identification of independent degrees of freedom in cyclic symmetry problems. There is one logical record.

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0	1-2	B	Data Block Name
1	1	I	Code* word for the elements in U_a set
	2	I	
	.		
	.		
	.		
	n	I	
2			End of file

Table Trailer

Word 1 = 1 for rotational; 2 for dihedral symmetry

Word 2 = number of degrees of freedom in a displacement set

Words 3-6 = zero

Code* for rotational 0 = interior point

+m = this point on side 1 is connected to m

-m = this point on side 2 is connected to m

Code for dihedral 0 = interior point

1 = side 1, even component

2 = side 1, odd component

3 = side 2, even component

4 = side 2, odd component

DATA BLOCK DESCRIPTIONS

2.3.62 Data Blocks Output From Module APD

2.3.62.1 GPLA (TABLE)

Description

Grid Point List - Aerodynamics

One logical record which contains a list of all grid points, scalar points, extra points and aerodynamic points in the model in internal sort.

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	ID for first point
	:		
	n	I	ID for n th point
2			End-of-file

Table Trailer

Word 1 = number of grid points + number of scalar points + number of extra points + aerodynamic points

Word 2-6 = zero

2.3.62.2 SILA (TABLE)

Description

Scalar Index List - Aerodynamics.

Two logical records. First logical record contains scalar index values in the pA-displacement set for each point in the model (internal order). These values are defined as follows:

$$SILA_1 = 1$$

$$SILA_{i+1} = \begin{cases} SILA_i + 6 & \text{if } i \text{ corresponds to a grid point} \\ SILA_i + 1 & \text{if } i \text{ corresponds to a scalar or an extra point} \end{cases}$$

The second logical record contains an equivalence between scalar index values in the g-displacement set and scalar index values in the pA-displacement set.

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Scalar index for first point
	:	:	
	n	I	Scalar index for n th point

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
2	1,2	I	SIL value, SILA value
	:	:	:
3	2m-1,2m	I	SIL value, SILA value End-of-file

Table Trailer

Word 1 = number of degrees of freedom in the pA-displacement set (LUSETA)

Word 2 = number of extra points

Word 3-6 = zero

2.3.62.3 USETA (TABLE)

Description

Displacement set definitions table - aerodynamics.

USETA contains one logical record. Each word corresponds to each degree of freedom in the pA-displacement set (in internal order) and contains ones in specified bit positions indicating the displacement set to which the point belongs.

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	L	Mask for first degree of freedom
	:	:	:
	n	L	Mask for n th degree of freedom
2			End-of-file

Notes:

Bit positions* for the various displacement sets are defined as follows:

pA	k	sA	pS	d	f _e	n _e	p	e	s _b	s _g	l	a	f	n	g	r	o	s	m
13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32

*Bit positions are numbered 1-32 from left to right for the right-most 32 bits of the computer word.

Table Trailer

Word 1 = number of degrees of freedom in the pA-displacement set (LUSETA).

Word 2 = number of extra points.

Word 3 = zero.

Word 4 = logical "or" of all USETA masks.

Word 5 = zero.

DATA BLOCK DESCRIPTIONS

Word 6 = zero.

2.3.62.4 EQAERØ (TABLE)

Description

Equivalence between external points and scalar index values - aerodynamics.

EQAERØ contains two logical records. The first record contains pairs of external point numbers and internal grid point numbers in the pA-displacement set for the points in external order. The second record is essentially the same as the first except that the type of point (grid, scalar, extra) is coded in the second word of the pair and the internal grid point number is converted to an SILPA value. Note that the internal grid point number is a pointer into the BGPA and hence extra points have an internal grid point number of zero.

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1,2	I	internal grid point number - first point
	⋮	⋮	⋮
	2n-1,2n	I	internal grid point number - n th point
2	1,2	L	ID for first point, 10*scalar index + type
	⋮	⋮	⋮
	2n-1,2n	I	ID for n th point, 10*scalar index + type
3			End-of-file

Note:

1 for grid point
 Type = 2 for scalar point
 3 for extra point

Table Trailer

Word 1 = number of grid points + number of scalar points + number of extra points + number of aerodynamic points.
 Word 2 = number of extra points.
 Word 3-6 = zero.

2.3.62.5 ECTA (TABLE)

Description

Element Connection Table - Aerodynamics.

The ECTA contains one logical record for each element connection card type that has been input, and one logical record for GENEL elements if they have been input. Additionally, the

DATA BLOCK AND TABLE DESCRIPTIONS

ECTA contains one logical record for all box elements generated for aerodynamics.

Table Formats

For other than box elements the ECTA is identical in format to data block GEOM2, output from module IFP. All external grid or scalar numbers are replaced by internal numbers. SPØINT data is not copied on the ECT.

The format for aerodynamic box elements consists of six words per box

<u>Word</u>	<u>Type</u>	<u>Item</u>
1	I	Element ID
2	I	Internal ID Grid Point 1
3	I	Internal ID Grid Point 2
4	I	Internal ID Grid Point 3
5	I	Internal ID Grid Point 4
6	I	Internal ID Grid Point 5

Box Element

Table Trailer

The trailer of the ECT table is adjusted to reflect the presence of box elements and used for the ECTA table.

2.3.62.6 CSTMA (TABLE)

Description

Coordinate System Transformation Matrices - Aerodynamics.

One logical record contains all coordinate system transformations. Transformation is from global to basic by the following formulation:

(1) rectangular

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_B = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}_g + \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}$$

DATA BLOCK DESCRIPTIONS

(2) cylindrical

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_B = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} R \cos \theta \\ R \sin \theta \\ z \end{bmatrix}_g + \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}$$

(3) spherical

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_B = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} \rho \sin \theta \cos \phi \\ \rho \sin \theta \sin \phi \\ \rho \cos \theta \end{bmatrix}_g + \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}$$

Table Format

<u>Record</u>	<u>Word</u>	<u>Item</u>	
0		Header record	
1	1	Coordinate system ID	
	2	Coordinate system type	$\left. \begin{array}{l} 1 = \text{rectangular} \\ 2 = \text{cylindrical} \\ 3 = \text{spherical} \end{array} \right\} \text{repeated for each coordinate system}$
	3-5	t_1, t_2, t_3	
	6-14	$r_{11}, r_{12}, r_{13}, r_{21}, r_{22}, r_{23}, r_{31}, r_{32}, r_{33}$	
2		End-of-file	

Notes:

1. Coordinate system ID and coordinate system type are integers.
2. t_i and r_{ij} are single precision floating point.

Table Trailer

Word 1 = number of grid points + number of scalar points + aerodynamic points.

Word 2 = number of coordinate systems.

Word 3-6 = zero.

2.3.62.7 BGPA (TABLE)

Description

Basic Grid Point Definition Table - Aerodynamics.

One logical record contains a list of all grid, scalar and aerodynamic points in internal sort, with (for grid points) their x, y, z coordinates in the basic system along with a coordinate system ID for displacement computations.

DATA BLOCK AND TABLE DESCRIPTIONS

Table Format

<u>Record</u>	<u>Word</u>	<u>Item</u>
0		Header record
1	1	Coordinate system ID
	2-4	x, y, z in basic system
2		End-of-file

Notes:

1. Coordinate system ID is integer; x, y, z are single precision, floating point.
2. Scalar points are identified by coordinate system ID = -1, and x, y, z = 0.

Table Trailer

Word 1 = number of grid points + number of scalar points + number of aerodynamic points.
 Word 2-6 = zero.

2.3.62.8 AERØ (TABLE)

Description

Aerodynamic Matrix Data.

Contains information for control of aerodynamic matrix generation and flutter analysis.

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0	1-2	B	Data Block Name
1	1	I	ND, Y-sym flag SYMXZ
	2	I	NE, Z-sym flag SYMXY
	3	R	REFC, reference chord BREF
2	1	R	First value of m
	2	R	First value of k
	3	R	Second value of m
	4	R	Second value of k
	:		
		I	-1, the end

Table Trailer

Word 1 = 1
 Word 2-6 = zero.

2.3.62.9 ACPT (TABLE)

Description

Aerodynamic Connection and Property Table

DATA BLOCK DESCRIPTIONS

Contains one record for each independent group of aerodynamic elements.

Table Format

Record	Word	Type	Item
0	1-2	B	Data Block Name
1	1	I	Key word, 1 for doublet lattice
	2	I	Number of panels, NP
	3	I	Number of strips, NSTRIP
	4	I	Number of boxes, NTP
	5	R	F, fraction of box chord from center of pressure to downwash center
	NP words	I	NCARAY, boxes per chord
	NP words	I	NBARAY, last box on panel
	NSTRIP words	R	YS aero coordinates of strip
	NSTRIP words	R	ZS center
	NSTRIP words	R	EE strip half width
	NSTRIP words	R	SG sine of dihedral angle
	NSTRIP words	R	CG cosine of dihedral angle
	NTP words	R	XIC coordinate of center of pressure
	NTP words	R	DELX box chord
	NTP words	R	XLAM tangent of sweepback angle
2	1	I	Key word, 2 for Doublet Lattice with Bodies
	2	I	NJ, Number of J points
	3	I	NK, Number of K points
	4	I	NP, Number of Panels
	5	I	NB, Number of Bodies
	6	I	NTP, Number of Boxes
	7	I	NBZ, Number of Z Bodies
	8	I	NBY, Number of Y Bodies
	9	I	NTZ, Number of Z Interference Body Elements
	10	I	NTY, Number of Y Interference Body Elements
	11	I	NT0, Sum of NTP + NTZ + NTY
	12	I	NTZS, Number of Z Slender Body Elements
	13	I	NTYS, Number of Y Slender Body Elements
	14	I	NSTRIP, Number of strips on panels
	NP words	I	NCARAY, Boxes per chord
	NP words	I	NBARAY, Last box on panel
	NP words	I	NAS, Associated bodies per panel
	NB words	I	* NBEA1, Number of interference elements
	NB "	I	* NBEA2, Z-Y flag
	NB "	I	* NSBEA, Number of slender elements
	NB "	R	ZB, Z Body center
	NB "	R	YB, Y Body center
	NB "	R	AVR, Half-width of body
	NB "	R	ARB, Cross-section aspect ratio
	NB "	I	NFL, θ -distribution per body
	NB "	R	XLE, X-leading edge
	NB "	R	XTE, X-trailing edge
	NB "	I	NT121, number θ_1 's for bodies
	NB "	I	NT122, number θ_2 's for bodies
	NB+NSTRIP	R	ZS, Z - of strip center
	NB+NSTRIP	R	YS, Y-of strip center
	NSTRIP words	R	EE, strip half width
	NSTRIP words	R	SG, size of dihedral angle
	NSTRIP words	R	CG, cosine of dihedral angle
	NTP+ Σ NBEA1	R	X, 3/4 chord
	NTP+ Σ NBEA1	R	DELX, box chord
	NTP words	R	XIC, coordinates of center of pressure
	NTP words	R	XLAM, tangent of sweepback angle
	Σ NSBEA words	R	A0, half-widths for bodies
	Σ NSBEA words	R	XIS1, X-of slender leading edge
	Σ NSBEA words	R	XIS2, X-of slender trailing edge

* Sum of entries is noted by Σ NBEAi, where i = 1, 2)

DATA BLOCK AND TABLE DESCRIPTIONS

Table Format (Cont'd.)

Record	Word	Type	Item
2	ΣNSBEA words	R	AØP, X-derivatives of body half-width
	ΣNBEA1 words	R	RIA, Radius of interference elements
	ΣNAS words	I	NASB, associated bodies
	ΣNFL words	I	IFLA1, body with θ_1 distribution
	ΣNFL words	I	IFLA2, Body with θ_2 distribution
	ΣNT121 words	R	TN1A, θ_1 's for bodies
	ΣNT122 words	R	TN2A, θ_2 's for bodies
3	1	I	Key word, 3 for MACH Box
	2	I	Number of J and K points - NTOT
	3	L	CRANK1
	4	L	CRANK2
	5	L	CNTRL1
	6	L	CNTRL2
	7	I	NBØX
	8	I	NPTS0 - wing points
	9	I	NPTS1 - control 1 points
	10	I	NPTS2 - control 2 points
	11-34	R	Twelve (x,y) pairs to define platform
4	2*NTØT	R	NTØT (x,y) pairs to define control points (wing, control 1, control 2)
	1	I	Key word, 4 for STRIP theory
	2	I	Number of J and K points
	3	I	CLA
	4	I	LCLA
	5	I	CIRC
	6	I	LCIRC
	7	I	NMACH
	8	I	NSTRIP
	9	R	CLAM
	If(CLA=-1 & LCLA>0)		
	NSTRIP+1	R	$M, CLA_1, CLA_2, \dots, CLA_{NSTRIP}$
	If(CLA=1 & LCLA>0)		
	NMACH*(NSTRIP+1) R		$M1, CLA_1, \dots, CLA_{NSTRIP}$
			$M2, CLA_1, \dots, CLA_{NSTRIP}$
			$M_{NMACH}, CLA_1, \dots, CLA_{NSTRIP}$
	If(CIRC>0 & LCIRC>0)		
	NMACH*(2*CIRC+2) R		$M1, b_0, b_1, B_1, \dots, b_{CIRC}, B_{CIRC}$
			$M_{NMACH}, b_0, b_1, B_1, \dots, b_{CIRC}, B_{CIRC}$
	5	I	Key word, 5 for Piston Theory
	2	I	Number of J and K points
	3	I	NMACH
	4	I	NTHRY
	5	I	NTHICK
	6	I	NALPHA
	7	I	NXIS
	8	I	NTAUS
	9	I	NSTRIP
	10	R	SECLAM
	NSTRIP	R	DELTAX
	NSTRIP	R	BLØC
	NSTRIP	R	CA
	If(NALPHA=1)		
	2*NMACH	R	$M_1, ALPHA_1, \dots, M_{NMACH}, ALPHA_{NMACH}$
	If(NALPHA=NSTRIP)		
	NMACH*(NSTRIP+1) R		$M_1, ALPHA_1, \dots, ALPHA_{NSTRIP}$

DATA BLOCK DESCRIPTIONS

Table Format (Cont'd.)

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
5(Cont'd.)			
If(NTHICK 0 & NXIS=0)	6 words	R	I_1, \dots, I_6
If(NTHICK 0 & NXIS=1)	13 words	R	$I_1, \dots, I_6, J_1, \dots, J_6, \epsilon_{h1}$
If(NTHICK 0 & NXIS=NSTRIP)	12+NSTRIP	R	$I_1, \dots, I_6, J_1, \dots, J_6, \epsilon_{h1}, \dots, \epsilon_{NSTRIP}$
If(NTHICK=0 & NTAUS=1)	5 words	R	$\tau_1, \tau_{h1}, \tau_{T1}, \epsilon_{m1}, \epsilon_{h1}$
If(NTHICK=0 & NTAUS=NSTRIP)	5*NSTRIP	R	$\tau_1, \tau_{h1}, \tau_{T1}, \dots, \tau_{NSTRIP}, \tau_{hNSTRIP}, \tau_{TNSTRIP}$ $\epsilon_{m1}, \epsilon_{h1}, \dots, \epsilon_{MSTRIP}, \epsilon_{hNSTRIP}$

Table Trailer

Word 1 = 1
Words 2-6= zero

2.3.62.10 SPLINE (TABLE)

Card Types and Header Information

<u>Card Type</u>	<u>Header Word 1</u> <u>Card Type</u>	<u>Header Word 2</u> <u>Trailer Bit Position</u>	<u>Header Word 3</u> <u>System Card Number</u>
SET1	3502	35	268
SET2	3602	36	269
SPLINE	200	2	0
SPLINE1	3302	33	266
SPLINE2	3402	34	267
SPLINE3	4901	49	173

Card Type Formats

SET1 (open-ended) (grid points with internal numbering)	SID ...	G1 -1	G2
SET2 (26 words)	SET2 card with two pack words plus the CAERØ card referenced.		
SPLINE (open-ended)	3 words per k point in problem External ID BGPA pointer K column number		
SPLINE1 (26 words)	SPLINE1 card plus 4 extra cards plus the CAERØ card referenced.		
SPLINE2 (26 words)	SPLINE2 card plus the CAERØ card referenced.		
SPLINE3 (open-ended)	SPLINE3 card with grid points in internal order plus the CAERØ card referenced.		

NOTE: For all cards with appended CAERØ, the PID has been replaced by the CAERØ coord system ID, field 5 is NSPAN, field 6 is NCHØRD, and field 9 is CAERØ type (1-5). For CAERØ type 2, field 18 has body orientation.

Table Trailer

Words 1-6 = Identical to EPT Table.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.62.11 FLIST (TABLE)

Description

Flutter Control Table

The FLIST table contains copies of the AERØ. FLFACT and FLUTTER cards in IFP (LOCATE) format.

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0	1-2	B	FLIST (Data Block Name)
1	1-3	I	Locate code AERØ card
	4-EOF	mixed	AERØ card
2	1-3	I	Locate code FLFACT card
	4-EOF	mixed	FLFACT card
3	1-3	I	Locate code FLUTTER card
	4-EOF	mixed	FLUTTER card

Table Trailer

Word 1-6 = Identical to EDT table.

2.3.62.12 SILGA (TABLE)

Description

Scalar Index List - Aerodynamic boxes only.

The SILGA table is a copy of the SIL table with the grid points added by the Aerodynamic boxes appended. It does not contain Extra Points and hence it, BGPA and EQAERØ go together.

Table Format

See SIL

Table Trailer

See SIL

DATA BLOCK DESCRIPTIONS

2.3.63 Data Blocks Output from Module GI

2.3.63.1 GTKA (MATRIX)

Description

$[G_{ka}]^T$ - Transformation matrix for displacements in the k-set in terms of deflection in the a-set.

Matrix Trailer

Number of columns = k

Number of rows = a

Form = rectangular

Type = real - single precision

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.64 Data Blocks Output From Module AMG

2.3.64.1 AJJL (MATRIX)

Description

$[A_{JJ_N} + A_{JJ_N} \dots A_{JJ_N}]$ - aerodynamic influence matrix list where N = number of pairs of Mach number and reduced frequency input.

Matrix Trailer

number of columns = $j*N$ (see definition of N above)
number of rows = j
form = rectangular
type = complex single precision

AJJL Header

<u>Word</u>	<u>Value</u>
-------------	--------------

1 - 2	- AJJL
-------	--------

3	- NJ = total j points
---	-----------------------

4	- NK = total k points
---	-----------------------

5	- NMK = size of record 2 of AERØ - number of m,k pairs
---	--

6	- NMK*2 = m,k lists
---	---------------------

6+2NMK - NGP = number of records in ACPT

7+2NMK - METHOD - theory ID = 1 = Doublet Lattice; 2 = Doublet Lattice with Slender Bodies;
3 = Mach Box; 4 = Strip Theory; 5 = Piston Theory.

8+2NMK - NJ = number of j points for this record.

9+2NMK - NK = number of k points for this record.

Words 6+2NMK, 7+2NMK, 8+2NMK and 9+2NMK are repeated NGP times.

2.3.64.2 SKJ (MATRIX)

Description

$[S]$ = integration matrix

DATA BLOCK DESCRIPTIONS

Matrix Trailer

number of columns = J
number of rows = K
form = rectangular
type = real singular precision

2.3.64.3 D1JK

Description

$[D_{1jk}]$ - Real part of downwash matrix

Matrix Trailer

number of columns = j
number of rows = k
form = rectangular
type = real singular precision

2.3.64.4 D2JK

Description

$[D_{2jk}]$ - imaginary part of downwash matrix

Matrix Trailer

number of columns = j
number of rows = k
form = rectangular
type = real singular precision

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.65 Data Blocks Output from Module AMP

2.3.65.1 QHHL (MATRIX LIST)

Description

$[Q_{hh}]$ - Aerodynamic Matrix List - h-set

QHHL is a matrix list. QHHL is used to store a series of matrices. Except for the header record, the format is that of a matrix. If there are NMK matrices, each with NRØW rows and NCØL columns, then QHHL will be stored as a matrix with NRØW rows and NMK times NCØL columns.

A special header record is written which contains the following information:

<u>Record</u>	<u>Location</u>	<u>Value</u>
0	1-2	Data Block Name
	3	NCØL
	4	NMK
	5-(4+2*NMK)	(m,k) pairs

Matrix Trailer

Number of columns/submatrix = h
 Number of columns/list = h*NMK
 Number of rows = h
 Form/submatrix = square
 Form/list = rectangular
 Type = complex $\begin{cases} \text{S.P. on CDC} \\ \text{D.P. on IBM} \end{cases}$

2.3.65.2 QKHL (MATRIX LIST)

Description

$[Q_{kh}]$ list - Aerodynamic transformation matrix between h and k sets. See QHHL description (Section 2.3.65.1).

Matrix Trailer

Number of columns/submatrix = h
 Number of columns/list = h*NMK

DATA BLOCK DESCRIPTIONS

Number of rows	= k
Form/submatrix	= rectangular
Form/list	= rectangular
Type	= complex $\begin{cases} \text{S.P. on CDC} \\ \text{D.P. on IBM, UNIVAC} \end{cases}$

2.3.65.3 QKHL (MATRIX LIST)

Description

$[Q_{kh}]$ list - Aerodynamic transformation matrix between h and k sets.
See QHHL description (Section 2.3.65.1).

Matrix Trailer

Number of columns/submatrix	= h
Number of columns/list	= h*NMK
Number of rows	= j
Form/submatrix	= rectangular
Form/list	= rectangular
Type	= complex $\begin{cases} \text{S.P. on CDC} \\ \text{D.P. on IBM, UNIVAC} \end{cases}$

DATA BLOCK DESCRIPTIONS

2.3.66 Data Blocks Output from Module FA1

2.3.66.1 FSAVE (TABLE)

Description

Flutter storage save table.

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0	1-2	BCD	FSAVE
	3	I	1 = k-method, 2 = KE-method, 3 = PK-method
	4	I	1 = surface spline interpolation
	5	R	2 = linear spline interpolation
	6	R	V_{sound}
	7	R	B_{ref}
	8	I	ρ_{ref}
			NVALUE
1	1	R	MACH
	2	R	KFREQ or Velocity
	3	R	RH0
			} Repeat for each loop
2	1	R	MACH
	2	R	k-freq
			Repeat for each matrix on Q_{hh} list
3	(1-LCC)		Case Control Record
4-end			Interpolated matrix save area for the K-method
4-end	1-2		One record for each m,k pair for the KE-method
			Eigenvalue - complex Repeat for each eigenvalue
			Word 6 of trailer has number of eigenvalues in each record.
4-end	1-2		One record for each m,k pair for the PK-method
	3		Eigenvalue - complex
	4		Reduced frequency of acceptance - Real
	5		Frequency - $.5 (\text{Im}(p)) - \text{Real}$
			Damping - $b/\pi v(R(p) \text{ if } \text{Im}(p)=0.0) - \text{Real}$
			- $2R(p)/\text{Im}(p) \text{ if } \text{Im}(p) \neq 0.0$
			Word 6 of trailer has number of eigenvalues in each record.

Table Trailer

- Word 1 Loop counter
- Word 2 Number of loops, number of triplets in record 1
- Word 3 Number of pairs in record 2
- Word 4 LCC, number of words on record 3
- Word 5 Type of matrix on FSAVE (1=SP, 2=DP, 3=CS, 4=DP)
- Word 6 Last column used by linear spline or number of RH0's for surface spline - K-method

DATA BLOCK DESCRIPTIONS

2.3.66.2 KXHH (MATRIX)

Description

$[K_{hh}^x]$ - total modal stiffness matrix - h-set.

Matrix Trailer

number of columns = h

number of rows = h

form = square

type = complex

Note: KXHH looks like PHID for the PK-method (see Section 2.3.42.1).

2.3.66.3 BXHH (MATRIX)

Description

$[B_{hh}^x]$ - total modal damping matrix - h-set.

Matrix Trailer

number of columns = h

number of rows = h

form = square

type = complex

Note: BXHH looks like CLAMA for the PK-method (see Section 2.3.42.2).

2.3.66.4 MXHH (MATRIX)

Description

$[M_{hh}^x]$ - total modal mass matrix - h-set.

Matrix Trailer

number of columns = h

number of rows = h

form = square

type = complex

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.67 Data Blocks Output From Module FA2

2.3.67.1 PHIHL (MATRIX)

Description

$[\phi_h]$ - appended complex mode shapes - h set

Matrix Trailer

number of columns = number of modes

number of rows = h

form = rectangular

type = complex single precision

2.3.67.2 CLAMAL (TABLE)

Description

λ - appended complex eigenvalue output table

Table Format

Same as CLAMA (2.3.42.2) except that the six data words in record 2 contain all eigenvalues found and currently selected.

Table Trailer

Word 1 contains the total number of eigenvalues

Words 2 - 6 = 0

2.3.67.3 CASEYY (TABLE)

Description

Appended Case Control Data Table

Table Format

Same as CASECC except that there is one record for each eigenvalue. Also words 103 through 134 contain FLØØP, k, m, and RHØ to provide some automatic output identification.

Table Trailer

Word 1 = total number of eigenvalues

Word 2 = 0

Word 3 = maximum length of CASEYY Record

Word 4 = PLØT Flag

Words 5-6 = 0

DATA BLOCK DESCRIPTIONS

2.3.67.4 ØVG (TABLE)

Description

Output aeroelastic curve requests (V-g or V-f).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code +10*approach code
	2	I	2002
	3	I	0
	4	I	1
	5	I	10*point ID + device code
	6	I	0
	7	I	0
	8	I	0
	9	I	Format code = 1
	10	I	Number of words per entry in next record = 4
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	R	Velocity
	2	I	Not used
	3-4	R	g, f
			Repeat for each eigenvalue

Notes:

1. Records 1 and 2 are repeated for each (m,k,p) point.
2. Device code = 0 = x y output only
3. Format code = 1 = real
4. Approach code = 6
5. Point ID = FLØØP
6. The label contains FLØØP, m, k and p.

Table Trailer

Words 1 - 6 contain no significant values.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.68 Data Blocks Output from Module ØPTR1

2.3.68.1 ØPTP1 (TABLE)

Description

Property optimization input table.

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>	
0	1,2	B	File name	
	3	I	Maximum number of iterations	
	4	R	Convergence criteria	
	5	R	Iteration factor	
	6	I	Print control	
	7	B	Punch control, YES or NØ	
1	1-NTYPES	I	Element type pointers where NTYPES is the number of types possible	
	NTYPES+1 to NTYPES+1+2* (NPØW+1)	I	Element type optimization pointers where NPØW is the number of types current available	
2	1	I	Element ID	} Repeated for each element meeting criteria
	2-4	R	Temperature dependent element stress limits	
	5	I	Property card pointer	
3	1	I	Property ID	} Repeated for each unique property referenced by Record 2.
	2	I	EST design variable *100 + material stress limit parameter	
	3	R	Original property value	
	4	R	Last property value (initially word 3 = word 4. Module ØPTR2 updates with every iteration)	
	5	R	Property change ratio limit (-1.0)	
	6	I	Property change limit	

DATA BLOCK DESCRIPTIONS

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>	
4	1	R	Minimum property change ratio (New/Original).	} Repeated for each referenced limit in Record 3.
	2	R	Maximum property change ratio (New/Original).	
5			End of file.	

Table Trailer

Word 1 = 0
 Word 2 = Number of words in record 2.
 Word 3 = Number of words in record 3.
 Word 4 = Number of words in record 4.
 Word 5 = 0
 Word 6 = Number of element types available to NASTRAN.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.69 Data Blocks Output from Module EMA

2.3.69.1 XGG (Matrix)

Description

XGG = KGG, KGGX, HKGG, HKGGX, BGG, HBGG, MGG, or K4GG (i.e., any structural matrix is optional).
For instance,

$$[K_{gg}] = \text{Stiffness matrix in matrix notation}$$

Matrix Trailer

Number of columns = number of rows = total degrees of freedom in the problem

form = symmetric

type = real

2.3.69.2 GPST (TABLE)

Description

Grid Point Singularity Table

Table Format

<u>Record</u>	<u>Word</u>	<u>Item</u>	
0		Header record	
1	1	Order of singularity (1, 2, or 3)	} Repeated for each singularity
	2	N = number of SIL numbers that follow	
	3	SIL ₁	
	4	SIL ₂	
	.		
	.		
	2+N	SIL _N	
2		End-of-file	

Note

All entries are integers.

Table Trailer

Word 1 = undefined
Word 2 = 0
Word 3 = 1
Word 4 = 2
Word 5 = 1
Word 6 = 0

DATA BLOCK DESCRIPTIONS

2.3.70 Data Blocks Output from Module EMG

2.3.70.1 KELM (TABLE)

Description

Element stiffness matrices table.

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0	1-2	B	Data block name.
1	1- n_1	R	First element stiffness partition of first element in the EST. (Length = n_1 is variable, depending on the element type.)
2	1- n_1	R	Second element stiffness partition of first element in the EST.
	.		
	.		
	.		
1+i-1	1- n_1	R	Last partition as described above.
	.		
	.		
	.		
j			End-of-file.

Notes

- Records 1 through "i" repeat for each element present within a given element type. "i" may vary, and in general, will equal the number of grid points the element connects. The total number of records j is determined by considering the number of element types, the number of elements within each type, and the number of connected grid points for each element. This table is reduced in size if valid congruences exist in the problem.
- This file is designed to be a reference file in direct access when used for data block KDICT.

Table Trailer

Word 1 = precision of data, 1 or 2

Words 2-6 = zero

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.70.2 KDICT (TABLE)

Description

Element dictionary table for the element stiffness-matrix partitions found in data block KMAT.

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0	1-2	B	Data block name.
1	1	I	Element type number.
	2	I	Length of entries beginning in Word 4 of this record.
	3	I	Maximum number of partitions to be written in the element matrix data block for each element of this element type = NLØCS.
Repeating entry for each element in this element type.	4	I	Serial EST element number.
	5	I	Matrix type 1 = square; matrix type 2 = diagonal.
	6	I	Column size of the element matrix.
	7	I	Component code word.
	8	R	Zero or damping constant.
	9 thru 8+NLØCS	I	GINØ-LØC identifications for corresponding partitions on file KMAT.

Notes

1. This data block will have a comparable number of records to that of the EST.
2. Record 1 will repeat for each element type present.

Table Trailer

Word 1 = precision of data, 1 or 2 in matrix data block
 Words 2-6 = zero

DATA BLOCK DESCRIPTIONS

2.3.70.3 MELM (TABLE)

Description

Element mass matrices table. Identical in format to that of data block KMAT. See Section 2.3.70.1.

2.3.70.4 MDICT (TABLE)

Description

Element dictionary table for the element mass matrix partitions found in data block MMAT. Identical in format to that of data block KDICT. See Section 2.3.70.2.

2.3.70.5 BELM (TABLE)

Description

Element damping matrices table. Identical in format to that of data block KMAT. See Section 2.3.70.1.

2.3.70.6 BDICT (TABLE)

Description

Element dictionary table for the element damping matrix partitions found in data block BMAT. Identical in format to that of data block KDICT. See Section 2.3.70.2.

2.3.70.7 HKELM (TABLE)

Description

See description and format of KELM table - section 2.3.70.1.

2.3.70.8 HKDICT (TABLE)

Description

See description and format of KDICT table - section 2.3.70.2.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.70.9 HBELM (TABLE)

Description

See description and format of BELM table - section 2.3.70.5.

2.3.70.10 HBDICT (TABLE)

Description

See description and format of BDICT table - section 2.3.70.6.

DATA BLOCK DESCRIPTIONS

2.3.71 Data Blocks Output from Module ASDMAP

2.3.71.1 CASESS (TABLE)

Description

In a Phase 2 substructure analysis, the command data CASESS is placed on the CASECC file preceding the normal case control data (See Sec. 2.3.1.1). The CASESS data is arranged as one record per substructure command. (No end-of-file is used between the CASESS and case control data.)

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0	1-2	B	Name = CASESS
1	1	B	Command name
	2	I	Number of words in record
	3	I	(CØMB only) Number of substructures to be combined
	4,5,6	B and I	Groups of three words
	⋮		KEY, V1, V2
	⋮		If V1 = -1, V2 = integer
	⋮		If V1 = -2, V2 = real
	⋮		
	3n+1,3n+2,3n+3		Otherwise both are BCD
2	etc.		

The following is a description of the data actually used by the Phase 2 modules. Some additional KEY values are used only during SDMAP processing and are not presented here. Note that all values are BCD unless otherwise indicated.

KEY words for command CØMB:

<u>Key</u>	<u>Corresponding V1, V2 Values</u>
NA1,NA2,...,NA7	Names of component substructures
NAMC	Name of combined structure
SØRT	'X', 'Y', or 'Z' sorting option
TØLE	Automatic connection tolerance (real)
CØNN	Connection set ID (integer)
CØMP	Component name

DATA BLOCKS AND TABLE DESCRIPTIONS

<u>Key</u>	<u>Corresponding V1, V2 Values</u>
TRAN	Transformation set ID (integer)
SYMT	Planes of symmetry (X, XY, Z, etc.)
SRCH	Name of structure to search
ØPTS	Connection option 'AUTO' or 'MAN'

Key words for command REDU:

NAMA	Name of substructure A
NAMB	Name of substructure B
BØUN	Boundary set ID (integer)

Key words for command RECØ:

PRIN	Name of substructure to be saved and printed
SAVE	Name of substructure to be saved

DATA BLOCK DESCRIPTION

2.3.72 Data Blocks Output from Module CQMB2

2.3.72.1 MATC (MATRIX)

Description

$[K_{gg}^{(i)}]$, $[M_{gg}^{(i)}]$, or $[P_g^{(i)}]$ - Phase 2 stiffness, mass, or load matrix for substructure i-g set.

Matrix Trailer

Stiffness and mass:

Number of columns = g

Number of rows = g

Form = symmetric

Type = real double precision

Loads:

Number of columns = total number of Phase 1 subcases in all component basic substructures

Number of rows = g

Form = rectangular

Type = real single precision

DATA BLOCKS AND TABLE DESCRIPTIONS

2.3.73 Data Blocks Output from Module RCØVR

2.3.73.1 ØPHIG (TABLE)

Description

Output eigenvector requests (g set, SØRT1, real).

Table Format

See Section 2.3.28.13, ØPHIG.

2.3.73.2 ØUGV1 (TABLE)

Description

Output displacement vector requests (g set, SØRT1, real).

Table Format

See Section 2.3.28.1, ØUGV1.

2.3.73.3 ØPG1 (TABLE)

Description

Output load vector requests (g set, SØRT1, real).

Table Format

See Section 2.3.28.5, ØPG1.

2.3.73.4 ØQG1 (TABLE)

Description

Output reaction forces requests (g set, SØRT1, real).

Table Format

See Section 2.3.28.8, ØQG1.

DATA BLOCK DESCRIPTIONS

2.3.73.5 U1 }
 U2 }
 U3 } (MATRIX)
 U4 }
 U5 }

Description

[u_g] - Displacement vector matrices - g set of the appropriate substructure.

Matrix Trailer

Number of columns = number of subcases in Phase 2

Number of rows = g (of the appropriate substructure)

Form = rectangular

Type = real single precision

DATA BLOCKS AND TABLE DESCRIPTIONS

2.3.74 Data Blocks Output from Module RC0VR3

2.3.74.1 UAS (MATRIX)

Description

[u_a] - Displacement matrix from Phase 2 substructure solution - a set.

Matrix Trailer

Number of columns = number of subcases in Phase 2 during solution

Number of rows = a

Form = rectangular

Type = real single precision

2.3.74.2 QAS (MATRIX)

Description

[q_a] - Matrix of reaction forces from Phase 2 substructure solution - a set.

Matrix Trailer

Number of columns = number of subcases in Phase 2 during solution

Number of rows = a

Form = rectangular

Type = real single precision

2.3.74.3 PGS (MATRIX)

Description

[P_g] - Static load vectors for Phase 2 substructure solution - g set.

Matrix Trailer

Number of columns = number of subcases in Phase 2 during solution

Number of rows = g

Form = rectangular

Type = real single precision

2.3.74.4 PSS (MATRIX)

Description

[P_s] - Static load vectors for Phase 2 substructure solution - s set.

DATA BLOCK DESCRIPTIONS

Matrix Trailer

Number of columns = number of subcases in Phase 2 during solution

Number of rows = s

Form = rectangular

Type = real single precision

2.3.74.5 P0S (MATRIX)

Description

[P₀] - Static load vectors for Phase 2 substructure solution - o set.

Matrix Trailer

Number of columns = number of subcases in Phase 2 during solution

Number of rows = o

Form = rectangular

Type = real single precision

2.3.74.6 YSS (MATRIX)

Description

[Y_s] - Constrained displacement vectors for Phase 2 substructure solution - s set.

Matrix Trailer

Number of columns = number of subcases in Phase 2 during solution

Number of rows = s

Form = rectangular

Type = real single precision

2.3.74.7 LAMA (TABLE)

Description

Real eigenvalue table from Phase 2 substructure solution.

Table Format

See Section 2.3.30.1, LAMA.

DATA BLOCKS AND TABLE DESCRIPTIONS

2.3.75 Data Blocks Output from Module REDUCE

2.3.75.1 PVX (MATRIX)

Description

{PVX} - The partitioning vector defining the degrees of freedom retained after a reduce operation. The value 1.0 is assigned if the degree of freedom is retained, and the value 0.0 if the degree of freedom is reduced out.

Matrix Trailer

Number of columns = 1

Number of rows = n, the number of degrees of freedom in the unreduced substructure

Form = rectangular

Type = real single precision

2.3.75.2 USX (TABLE)

Description

The displacement set definitions table corresponding to PVX with only the flags for u_0 and u_a set.

Table Format

<u>Record</u>	<u>Word</u>	<u>Item</u>
0	1-2	Data block name
	3-4	0.0
1	1	Mask for first degree of freedom
	:	
	:	
	n	Mask for n th degree of freedom
2		End-of-file

Table Trailer

Word 1 = 0

Word 2 = n

Word 3 = 0

Word 4 = logical "OR" of all USET masks

Words 5-6 = 0

DATA BLOCK DESCRIPTIONS

2.3.75.3 INX (MATRIX)

Description

[INX] - The identity matrix partition of the reduction transformation:

$$[G_{AB}] = \left[\begin{array}{c} \text{INX} \\ \hline G_0 \end{array} \right]$$

Matrix Trailer

Number of columns = n

Number of rows = n

Form = square

Type = real single precision

DATA BLOCKS AND TABLE DESCRIPTIONS

2.3.76 Data Blocks Output from Module SGEN

2.3.76.1 CASES (TABLE)

Description

Substructure case control.

Table Format

CASES is identical to data block CASESS, Section 2.3.71.1.

2.3.76.2 CASEC (TABLE)

Description

Case Control data table for Phase 2 substructure analysis.

Table Format

See Section 2.3.1.1, CASECC.

2.3.76.3 GPL (TABLE)

Description

Grid point list for Phase 2 substructure SOLVE operation.

Table Format

Identical to data block GPL, Section 2.3.3.1, where a set of N scalar points with sequential identification numbers has been defined. N is the number of degrees of freedom in the g-set of the substructure to be solved.

2.3.76.4 EQEXIN (TABLE)

Description

Equivalence between external grid or scalar numbers and internal numbers for Phase 2 substructure SOLVE operation.

Table Format

Identical to data block EQEXIN, Section 2.3.3.2, where a set of N scalar points with sequential identification numbers has been defined. N is the number of degrees of freedom in the g-set of the substructure to be solved.

DATA BLOCK DESCRIPTIONS

2.3.76.5 GPD^T (TABLE)

Description

Grid point definition table for Phase 2 substructure SØLVE operation.

Table Format

Identical to data block GPD^T, Section 2.3.3.3, where a set of N scalar points with sequential identification numbers has been defined. N is the number of degrees of freedom in the g-set of the substructure to be solved.

2.3.76.6 BGPDT (TABLE)

Description

Basic grid point definition table for Phase 2 substructure SØLVE operation.

Table Format

Identical to data block BGPDT, Section 2.3.3.5, where a set of N scalar points with sequential identification numbers has been defined. N is the number of degrees of freedom in the g-set of the substructure to be solved.

2.3.76.7 SIL (TABLE)

Description

Scalar index list for Phase 2 substructure SØLVE operation.

Table Format

Identical to data block SIL, Section 2.3.3.6, where a set of N scalar points with sequential identification numbers has been defined. N is the number of degrees of freedom in the g-set of the substructure to be solved.

2.3.76.8 GP3S (TABLE)

Description

GEØM3 data block for Phase 2 substructure SØLVE operation.

Table Format

The format is identical to the GEØM3 data block, Section 2.3.2.3. However, GP3S contains only LØAD card and SLØAD card data.

DATA BLOCKS AND TABLE DESCRIPTIONS

2.3.76.9 GP4S (TABLE)

Description

GEØM4 data block for Phase 2 substructure SØLVE operation.

Table Format

The format is identical to the GEØM4 data block, Section 2.3.2.4. However, MPCs cards have been converted to MPC cards, SPCS to SPC, SPCS1 to SPC1, and SPCSD to SPCD. Also, all grid numbers now refer to scalar numbers used in the SØLVE operation. See Section 2.3.76.3, GPL, for Phase 2 substructure SØLVE operation.

2.3.76.10 CSTM (TABLE)

Description

This data block is always purged. It exists solely to prevent DMAP compilation errors.

DATA BLOCK DESCRIPTIONS

2.3.77 Data Blocks Output from Module PLTMRG

2.3.77.1 PLTP (TABLE)

Description

Plot parameters and plot control table for Phase 2 substructure plots.

Table Format

<u>Record</u>	<u>Item</u>
0	Header record
1	Duplicate of the
2	plot control data block
3	(PCDB) created by IFP1.
etc.	
Last	End-of-file

Table Trailer

Words 1-5 = 0

Word 6 = 1

2.3.77.2 GPS (TABLE)

Description

Grid point set related to the element plot set for Phase 2 substructure plots.

Table Format

GPS is identical to data block GPSETS, Section 2.3.5.3, except there is only one plot set and its ID is 1.

Table Trailer

Words 1-5 = 0

Word 6 = 1

DATA BLOCKS AND TABLE DESCRIPTIONS

2.3.77.3 ELS (TABLE)

Description

Element plot set connection table for Phase 2 substructure plots.

Table Format

ELS is identical to data block ELSETS, Section 2.3.5.4, except there is only one plot set.

Table Trailer

Words 1-5 = 0

Word 6 = 1

2.3.77.4 BGP (TABLE)

Description

Basic grid point definition table for Phase 2 substructure plots. BGP contains the coordinates of all grid points which were defined in Phase 1 for each basic substructure comprising the substructure to be plotted.

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>	
0			Header record	
1	1	I	Coordinate system ID	} Repeated for each grid or scalar point
	2-4	R	X,Y,Z in basic system	
2			End-of-file	

Notes

1. X,Y,Z are in the basic system of the substructure to be plotted, which may differ from the basic system of the basic substructure.
2. Scalar points are identified by a coordinate system ID = -1 and X,Y,Z = 0.

Table Trailer

Word 1 = Number of grid points + number of scalar points

Words 2-6 = 0

DATA BLOCK DESCRIPTIONS

2.3.77.5 CASEP (TABLE)

Description

Case control table for Phase 2 substructure plots.

Table Format

See Section 2.3.1.1, CASECC.

2.3.77.6 EQEX (TABLE)

Description

Equivalence between external grid or scalar numbers and internal numbers for Phase 2 substructure plots, external sort.

EQEX contains an entry for all grid and scalar points which were defined in Phase 1 for each basic substructure comprising the substructure to be plotted.

Table Format

See Section 2.3.3.2, EQEXIN.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.78 Data Blocks Output From Module M0DACC

2.3.78.1 CLAMA1 (TABLE)

CLAMA1 is identical to CLAMAL except that the eigenvalue list contains only those selected by the 0FREQ card.

2.3.78.2 CPHIH1 (MATRIX)

CPHIH1 is identical to PHIHL except that it contains only those columns which are selected by the 0FREQ card.

2.3.78.3 CASEZZ (TABLE)

CASEZZ is identical to CASEYY except only those records pertaining to the selected eigenvalues are retained.

DATA BLOCK DESCRIPTIONS

2.3.79 Data Blocks Output from Module DDRMM

2.3.79.1 ZUPV2 (TABLE)

Description

See data block description for ØUPV2, Section 2.3.45.3.

2.3.79.2 ZUPVC1 (TABLE)

Description

See data block description for ØUPVC1, Section 2.3.28.4.

2.3.79.3 ZUPVC2 (TABLE)

Description

See data block description for ØUPVC2, Section 2.3.45.11.

2.3.79.4 ZQP2 (TABLE)

Description

See data block description for ØQP2, Section 2.3.45.2.

2.3.79.5 ZQPC1 (TABLE)

Description

See data block description for ØQPC1, Section 2.3.28.12.

2.3.79.6 ZQPC2 (TABLE)

Description

See data block description for ØQPC2, Section 2.3.45.10.

2.3.79.7 ZES2 (TABLE)

Description

See data block description for ØES2, Section 2.3.45.4.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.79.8 ZESC1 (TABLE)

Description

See data block description for ØESC1, Section 2.3.28.18.

2.3.79.9 ZESC2 (TABLE)

Description

See data block description for ØESC2, Section 2.3.45.12.

2.3.79.10 ZEF2 (TABLE)

Description

See data block description for ØEF2, Section 2.3.45.5.

2.3.79.11 ZEFC1 (TABLE)

Description

See data block description for ØEFC1, Section 2.3.28.22.

2.3.79.12 ZEFC2 (TABLE)

Description

See data block description for ØEFC2, Section 2.3.45.13.

DATA BLOCK DESCRIPTIONS

2.3.80 Data Blocks Output from Module ØPTPR2

2.3.80.1 ØTP2 (TABLE)

Description

Property optimization output table.

Table Format

The ØTP2 is identical to form to data block ØTP1, output from ØTPR1. Record 3 word 4 is updated to reflect the last property value.

Table Trailer

Identical to trailer on ØTP1 data block.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.81 Data Blocks Output From Module INPUTT2

2.3.81.1 D1JE (MATRIX)

Description

$[D_{je}^1]$ - Downwash factors due to extra points - real

Matrix Trailer

Number of columns = e

Number of rows = j

Form = rectangular

Type = real

2.3.81.2 D2JE (MATRIX)

Description

$[D_{je}^2]$ - Downwash factors due to extra points - complex

Matrix Trailer

Number of columns = e

Number of rows = j

Form = rectangular

Type = real

DATA BLOCK DESCRIPTIONS

2.3.82 Data Blocks Output from Module CURV

2.3.82.1 ØES1M (TABLE).

Description

Output element stress requests (SØRT1, real).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code + 10* approach code
	2	I	5
	3	I	1000 + element type
	4	I	Subcase number
	5	I/R	Time, Load set ID, or mode number
	6	R/I	Eigenvalue or 0
	7	I	0
	8	I	Load set ID or 0
	9	I	Format code
	10	I	Number of words per entry in next record = NWDS
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*element ID + device code
	2-NWDS	Mixed	Element stress data
			See 2.3.51 for details

} repeat
for each
element.

Notes

- Records 1 and 2 are repeated for each vector to be output.
- Device code = $\begin{cases} 0 = x\ y\ \text{output only} \\ 1 = \text{print} \\ 4 = \text{punch} \\ 5 = \text{print and punch} \end{cases}$
- Format code = $\begin{cases} 1 = \text{real} \\ 2 = \text{real/imaginary} \\ 3 = \text{magnitude/phase} \end{cases}$
- Approach code = 1, 2, 3, 6, 7, or 10

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.82.2 DESIG (TABLE).

Description

Output requests for element stresses at grid points (SØRT1, real).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code = 10*approach code
	2	I	5
	3	I	2000 = Element type
	4	I	Subcase number
	5	I/R	Time, Load set ID, or mode number
	6	R/I	Eigenvalue or 0
	7	I	0
	8	I	Load set ID or 0
	9	I	Format code
	10	I	Number of words per entry in next record = NWDS
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*element ID + device code
	2-NWDS	Mixed	Element stress data
			See 2.3.51 for details

} repeat
for each
element.

Notes

- Records 1 and 2 are repeated for each vector to be output.
- Device code = $\begin{cases} 0 = x y \text{ output only} \\ 1 = \text{print} \\ 4 = \text{punch} \\ 5 = \text{print and punch} \end{cases}$
- Format code = $\begin{cases} 1 = \text{real} \\ 2 = \text{real/imaginary} \\ 3 = \text{magnitude/phase} \end{cases}$
- Approach code = 1, 2, 3, 6, 7, or 10

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK DESCRIPTIONS

2.3.82.3 ØESIAM (TABLE).

Description

Output element strain/curvature requests (SØRT1, real).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code + 10*approach code
	2	I	21
	3	I	1000 + Element type
	4	I	Subcase number
	5	I/R	Time, Load set ID, or mode number
	6	R/I	Eigenvalue or 0
	7	I	0
	8	I	Load set ID or 0
	9	I	Format code
	10	I	Number of words per entry in next record = NWDS
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*element ID = device code
	2-NWDS	Mixed	Element strain/curvature data
			See 2.3.51 for details

} repeat
for each
element.

Notes

- Records 1 and 2 are repeated for each vector to be output.
- Device code = $\begin{cases} 0 = x y \text{ output only} \\ 1 = \text{print} \\ 4 = \text{punch} \\ 5 = \text{print and punch} \end{cases}$
- Format code = $\begin{cases} 1 = \text{real} \\ 2 = \text{real/imaginary} \\ 3 = \text{magnitude/phase} \end{cases}$
- Approach code = 1, 2, 3, 6, 7, or 10

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.82.4 ØESIAG (TABLE).

Description

Output requests for element strains/curvatures at grid points (SØRT1, real).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code + 10*approach code
	2	I	21
	3	I	2000 + Element type
	4	I	Subcase number
	5	I/R	Time, Load set ID, or mode number
	6	R/I	Eigenvalue or 0
	7	I	0
	8	I	Load set ID or 0
	9	I	Format code
	10	I	Number of words per entry in next record = NWDS
	11-50		Not defined
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	10*element ID + device code
	2-NWDS	Mixed	Element strain/curvature data See 2.3.51 for details
			} repeat for each element.

Notes

1. Records 1 and 2 are repeated for each vector to be output.

2. Device code = $\begin{cases} 0 = x\ y\ \text{output only} \\ 1 = \text{print} \\ 4 = \text{punch} \\ 5 = \text{print and punch} \end{cases}$

3. Format code = $\begin{cases} 1 = \text{real} \\ 2 = \text{real/imaginary} \\ 3 = \text{magnitude/phase} \end{cases}$

4. Approach code = 1, 2, 3, 6, 7, or 10

Table Trailer

Words 1-6 contain no significant values.

DATA BLOCK DESCRIPTIONS

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.83 Data Blocks Output from Module GPFDR

2.3.83.1 ØNRGY1 (TABLE)

Description

Output element energy requests (SØRT1, real).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0	1-2	B	Header record
1	1	I	Device code + 10* approach code
	2	I	18 = ØFP major ID
	3	R	Total energy of all elements
	4	I	Subcase number
	5	I	0
	6-7	B	Element type name
	8-9	I	0
	10	I	3 = even number record's entry length
	11-50	I	0
	51-82	B	Title
	83-114	B	Subtitle
	115-146	B	Label
2	1	I	Device code + 10* Element ID
	2	R	Element energy
	3	R	Per cent of total energy

repeats for
each element
of type
specified by
words 6-7 of
previous
record

Notes

- Records 1 and 2 are repeated for each element type having at least one element requested for output.
- Device code = 1 = print, 4 = punch, 5 = print and punch.
- Approach code = 1 for STATICS.
- Additional subcases also cause all records to repeat.

DATA BLOCK DESCRIPTIONS

Table Trailer

Word 1 = number of element types output

Words 2-6 = 0

2.3.83.2 ØGPF1 (TABLE)

Description

Output grid point force balance requests (SØRT1, real).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>	
0	1-2	B	Header record	
1	1	I	Device code + 10* approach code	
	2	I	19 = ØFP major ID	
	3	I	0	
	4	I	Subcase number	
	5-9	I	0	
	10	I	10 = length of entries in record 2	
	11-50	I	0	
	51-82	B	Title	
	83-114	B	Subtitle	
	115-146	B	Label	
2	1	I	Device code + 10* Point ID	} repeats for each force of each point listed
	2	I	Element ID if element force entry or 0	
	3-4	B	Element name, APP-LOAD, F-OF-SPC, or *TOTALS*	
	5-10	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3)	

Notes

1. Records 1 and 2 repeat for each subcase having any output requests.
2. Device code = 1 = print, 4 = punch, 5 = print and punch.
3. Approach code = 1 for STATICS.

DATA BLOCK AND TABLE DESCRIPTIONS

Table Trailer

Word 1 = number of output line entries

Words 2-6 = 0.

2.3.84 Data Blocks Output from Module CYCT1

2.3.84.1 PX (MATRIX)

Description

$[P_x]$ - Partition of the load vector matrix giving static loads on - 1 set.

Matrix Trailer

Number of columns = number of subcases
Number of rows = 1
Form = rectangular
Type = real single precision

2.3.84.2 ULV

Description (MATRIX)

$[U_2]$ - Partition of the displacement vector matrix giving displacements - 1 set.

Matrix Trailer

Number of columns = number of subcases
Number of rows = 1
Form = rectangular
Type = real single precision

2.3.84.3 GCYCF (MATRIX)

Description

$[G_{cf}]$ - Transformation matrix for loads on symmetric components.

Matrix Trailer

Number of columns = number of subcases
Number of rows = 1
Form = rectangular
Type = real single precision

2.3.84.4 GCYCB (MATRIX)

Description

$[G_{cb}]$ - Transformation matrix for displacements on symmetric components.

Matrix Trailer

Number of columns = number of subcases
 Number of rows = 1
 Form = rectangular
 Type = real single precision

2.3.85 Data Blocks Output from Module CYCT2

2.3.85.1 KKK (MATRIX)

Description

$[K_{kk}]$ - Partition of transformed stiffness matrix - a set.

Matrix Trailer

Number of columns = k
 Number of rows = k
 Form = symmetric
 Type = real single/double precision

2.3.85.2 PK (MATRIX)

Description

$[P_k]$ - Partition of the transformed load vector matrix for the solution set.

Matrix Trailer

Number of columns = number of subcases
 Number of rows = 1 set
 Form = rectangular
 Type = real single precision

2.3.85.3 UXV (MATRIX)

Description

$[U_v]$ - Partition of the displacement vector matrix for the symmetric components.

Matrix Trailer

Number of columns = number of subcases
 Number of rows = 1 set
 Form = rectangular
 Type = real single precision

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.85.4 RUXV (MATRIX)

Description

$[\delta P_\ell]$ - Residual vector matrix for the ℓ set.

Matrix Trailer

Number of columns = number of subcases
Number of rows = ℓ set
Form = rectangular
Type = real single precision

2.3.85.5 MKK (MATRIX)

Description

$[M_{kk}]$ - Partition of transformed mass matrix - a set.

Matrix Trailer

Number of columns = k
Number of rows = k
Form = symmetric
Type = real single/double precision

2.3.85.6 PHIA (MATRIX)

Description

$[\phi_a]$ - Eigenvector matrix giving the eigenvector (displacements) in the a set.

Matrix Trailer

Number of columns = number of eigenvalues
Number of rows = a
Form = rectangular
Type = real single precision

2.3.85.7 LAMA (TABLE)

Description

λ_a - Real eigenvalue table.

Table Format

See section 2.3.30.1.

DATA BLOCK DESCRIPTIONS

2.3.86 Data Blocks Output from Module FRLG

2.3.86.1 Data Blocks PPF1, PSF1 and PDF1 are identical to PPF, PSF and PDF output by FRRD.

2.3.86.2 PHF (MATRIX)

Description

$[P_h^f]$ - Partition of load matrix - h-set.

Matrix Trailer

Standard trailer.

2.3.86.3 FØL (TABLE)

Description

Frequency Response Output List.

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0	1-2	BCD	Table Name (FØL)
	3-NFREQ+2	R	Frequencies f ($\omega=2\pi f$)

End-of-file

Table Trailer

Word 1 - Number of frequencies

Word 2 - Frequency set record number

Word 3 - Number of loads (proposed)

Words 4-6 - Zero

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.87 Data Blocks Output from Module LAMX

2.3.87.1 LAMX (TABLE or MATRIX)

Description

λ_a - Real Eigenvalue Table

Note: Table LAMX is the same as LAMA - Section 2.3.30.1.

$[\lambda]$ - Real Eigenvalue Matirx

Note: Number of rows is the number of eigenvalues on LAMA until generalized mass is zero.
Columns are eigenvalue, omega, frequency, generalized mass, and generalized stiffness.

DATA BLOCK DESCRIPTIONS

2.3.88 Data Blocks Output from Module IFT

2.3.88.1 UHVT (MATRIX)

UHVT is an h-set matrix where the columns relate to the time step output as follows:

Column 1 - displacement ($t=0$)

Column 2 - velocity (null)

Column 3 - acceleration (null)

Column 4 - displacement ($t=t_1$)

etc.

Matrix Trailer

Standard trailer.

2.3.88.2 TØL (TABLE)

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0	1-2	BCD	TØL
	3---N+3	R	Times

Table Trailer

Word 1 - Number of time step changes

Word 2 - Total number of times (N)

Words 3-6 - Zero

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.89 Data Blocks Output from Module FRRD2

2.3.89.1 UHVF (MATRIX)

Description

$[U_h^f]$ - Displacement vector matrix - h-set.

Matrix Trailer

Standard trailer.

DATA BLOCK DESCRIPTIONS

2.3.90 Data Blocks Output from Module ADR

2.3.90.1 PKF (MATRIX)

Description

$[P_{kf}]$ - Aerodynamic Forces (pressures) relating k-set to frequency.

Matrix Trailer

Standard trailer.

DATA BLOCK AND TABLE DESCRIPTIONS

2.3.91 Data Blocks Output from Module GUST

2.3.91.1 PHF (MATRIX)

Description

$[P_h^f]$ - Dynamic load vector matrix on modal degrees of freedom combining GUST and Dynamic Frequency Response Loads - h-set.

Matrix Trailer

Standard trailer.

DATA BLOCK DESCRIPTIONS

2.3.92 Data Blocks Output from module EQMCK

2.3.92.1 ØQM1

Description

Output forces of multi-point constraint requests (g set, SØRT1, real).

Table Format

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0			Header record
1	1	I	Device code + 10*approach code
	2	I	20
	3	I	0
	4	I	Subcase number
	5	I	Load set ID or mode number
	6	I	0 or eigenvalue
	7	I	0
	8	I	0
	9	I	Format code
	10	I	Number of words per entry in next record = 8
	11-50		Not defined
	51-82	A	Title
	83-114	A	Subtitle
	115-146	A	Label
2	1	I	10*point ID + device code
	2	I	Point
	3-8	R	R(T1), R(T2), R(T3), R(R1), R(R2), R(R3)

} repeat
for each
point

Notes

- Records 1 and 2 are repeated for each vector to be output.
- Device code, $\left\{ \begin{array}{l} 0 = \text{x-y output only} \\ 1 = \text{print} \\ 4 = \text{punch} \\ 5 = \text{print and punch} \end{array} \right.$
- Format code, 1 = real
- Approach code = 1 or 2
- Point type = $\left\{ \begin{array}{l} 1 = \text{grid point} \\ 2 = \text{scalar point} \end{array} \right.$

Table Trailer

Word 1 = Number of vectors
Word 2 = m set size
Word 3 = 0
Word 4 = 0
Word 5 = 1
Word 6 = 0

EXECUTIVE TABLE DESCRIPTIONS

2.4 EXECUTIVE TABLE DESCRIPTIONS

The following is an alphabetical index of Executive table descriptions.

<u>Section Number</u>	<u>Executive Table Name</u>	<u>Where Stored</u>	<u>Page Number</u>
2.4.1.5	CEITBL	/XCEITB/	2.4-9
2.4.1.4	DPL	/XDPL/	2.4-7
2.4.1.2	FIAT	/XFIAT/	2.4-3
2.4.1.3	FIST	/XFIST/ and /XPFIST/	2.4-5
2.4.2.8	IFPX0	/IFPX0/	2.4-62
2.4.2.9	IFPX1	/IFPX1/	2.4-63
2.4.2.7	LNKSPC	/XLKSPC/	2.4-60
2.4.2.2	MPL	/XGPI2/	2.4-22
2.4.2.1	ØSCAR	Data Pool File	2.4-16
2.4.2.4	PVT	/XPVT/	2.4-57
2.4.1.8	SYSTEM	/SYSTEM/	2.4-13
2.4.1.7	TAPID	/STAPID/	2.4-12
2.4.1.6	VPS	/XVPS/	2.4-10
2.4.2.6	XALTER	Problem Tape	2.4-59
2.4.2.5	XCSA	Problem Tape	2.4-58
2.4.1.1	XFIAT	/XXFIAT/	2.4-2
2.4.1.9	XLINK	/XLINK/	2.4-15
2.4.2.3	XPTDIC	Problem Tape	2.4-55
2.4.2.10	ITEMDT	/ITEMDT/	2.4-72

2.4.1 Executive Tables Which are Permanently Core Resident

2.4.1.1 XFIAT (Permanent File Allocation Table)

Description

A NASTRAN resident memory table containing the physical file identification for the permanent files (P00L, 0PTP, etc.).

Created in Module

The physical file identifications are output by GNFIAT (generate FIAT).

Table Format

Word 1	NOT USED	T P	FILE
	17	16	15
2	NOT USED	T P	FILE
	17	16	15
3	NOT USED	T P	FILE
	17	16	15
:	NOT USED	T P	FILE
:	17	16	15
N			

<u>Word</u>	<u>Item</u>	<u>Description</u>
1-N	TP	Tape Flag (1 bit) - set if physical file (FILE) is a magnetic tape.
	FILE	File ID (15 bits) - unique integer identification for a physical file.

Notes

1. The number of entries (N) is dictated by the integer value in PFIST (see FIST Executive Table Description - 2.4.1.3)
2. The XFIAT table is located in the named common block /XXFIAT/.

EXECUTIVE TABLE DESCRIPTIONS

2.4.1.2 FIAT (File Allocation Table)

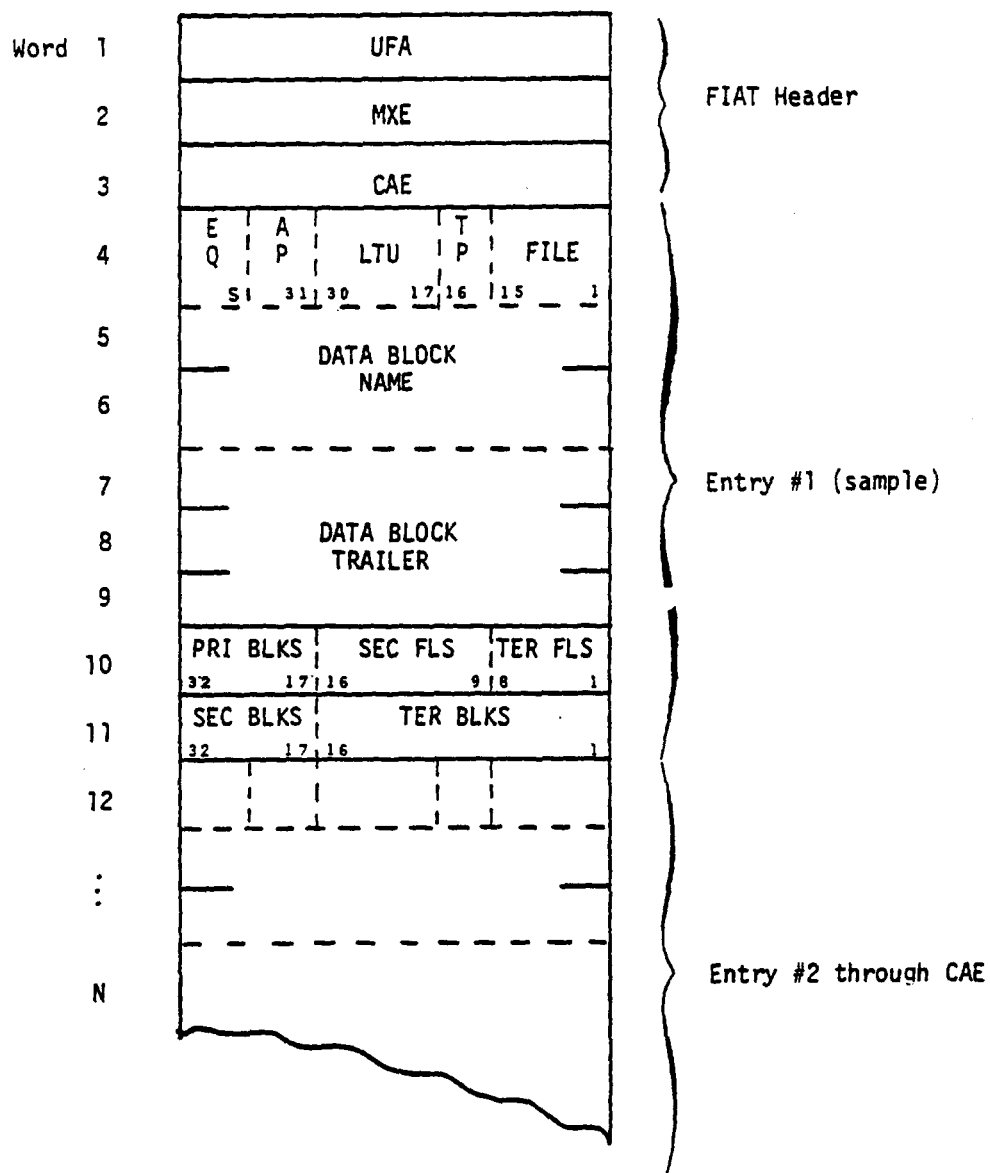
Description

A NASTRAN resident memory table containing the data block name vs. physical file ID. for a segment of DMAP modules.

Created in Module

The physical files available for the system/computer configuration are output by GNFIAT (generate FIAT). The data block names and other data block information are output by XSFA, Executive Segment File Allocator.

Table Format



DATA BLOCK AND TABLE DESCRIPTIONS

<u>Word</u>	<u>Item</u>	<u>Description</u>
1	UFA	Unique files available - this integer indicates the number of unique file entries in the FIAT.
2	MXE	Maximum entries - this integer shows the total entry size of the dimensioned FIAT table; the amount of memory reserved (N) = $6 \times \text{MXE} + 3$.
3	CAE	Current active entries - this integer designates the portion of FIAT currently containing valid data; $\text{UFA} \leq \text{CAE} \leq \text{MXE}$.

Words 4 through 9 describe a sample 6-word entry:

4	EQ	Equivalence flag (1 bit) - 0 bit, file not equivalenced. 1 bit, file equivalenced.
	AP	Append flag (1 bit) - set if append is specified for data block in DMAP sequence by a FILE DMAP instruction.
	LTU	Last time used (14 bit integer) - record number of ØSCAR entry for last use of data block.
	TP	Tape flag (1 bit) - set if physical file (FILE) is a magnetic tape.
	FILE	File ID (15 bits) - unique identification for a physical file.
5,6	NAME	Data block name - 8 characters (4 characters/word).
7,8,9	TRAILER	Data block trailer - storage for 6-16 bit data block trailer words.
10	PRI BLKS	Number of blocks written to the primary file (note: this includes all the data written on a NASTRAN file for the CDC and UNIVAC versions of NASTRAN. For IBM, this includes that portion of the NASTRAN file written on the "PRIxx" file but not that part of the NASTRAN file that was written on the "SECxx" and "TERxx" files that became logical extensions of the "PRIxx" file).
	SEC FLS	Number of "SECxx" files assigned as logical extensions of the "PRIxx" file. (Note: always "0" for CDC and UNIVAC).
	TER FLS	Number of "TERxx" files assigned as logical extensions of the "PRIxx" file. (Note: always "0" for CDC and UNIVAC).
11	SEC BLKS	Number of blocks written on all the "SECxx" files that are logical extensions of the "PRIxx" file (Note: always "0" for CDC and UNIVAC).
	TER BLKS	Number of blocks written on all the "TERxx" files that are logical extensions of the "PRIxx" file. (Note: always "0" for CDC and UNIVAC).

Words 12 through N contain repeated 6-word entries.

Trailer Information

Trailer information for each data block is sotred in and received from the FIAT by WRTTRL (write trailer) and RDTRL (read trailer).

EXECUTIVE TABLE DESCRIPTIONS

Notes

1. The FIAT table is located in the named common block /XFIAT/.
2. A printout of the FIAT may be obtained each time the file allocator (XSFA) is called. This diagnostic aid is activated by setting DIAG 2. The printout consists of a header and a 17-column table. The header contains UFA, MXE and CAE as defined above, and the ØSCAR record number of the last call (ØSCAR STR) made to XSFA and the ØSCAR record number of the current call (ØSCAR STP) to XSFA. The columns of the table are:

<u>Column</u>	<u>Symbol</u>	<u>Description</u>
1	EQ	See above
2	AP	See above
3	LTU	See above
4	TP	See above
5	UNIT	See FILE above
6	NTU	Next time used - ØSCAR record number for next use of data block.
7	ØF	Stack flag - turned off after stacking to prevent double stacking (see Section 4.9.5.2)
8	SG	Set to -1 if data block is allocated for module within current segment (see Section 4.9.5.2)
9	KN	Allocation type (see Section 4.9.5.2)
10	TR	Tape request flag (see Section 4.9.5.2)
11	DATA BLK	See NAME above
12 - 17	TRAILER	See above
18	PRI BLKS	See above
19	SEC FLS	See above
20	SEC BLKS	See above
21	TER FLS	See above
22	TER BLKS	See above

DATA BLOCK AND TABLE DESCRIPTIONS

THIS PAGE HAS BEEN LEFT BLANK INTENTIONALLY.

EXECUTIVE TABLE DESCRIPTIONS

2.4.1.3 FIST (File Status Table)

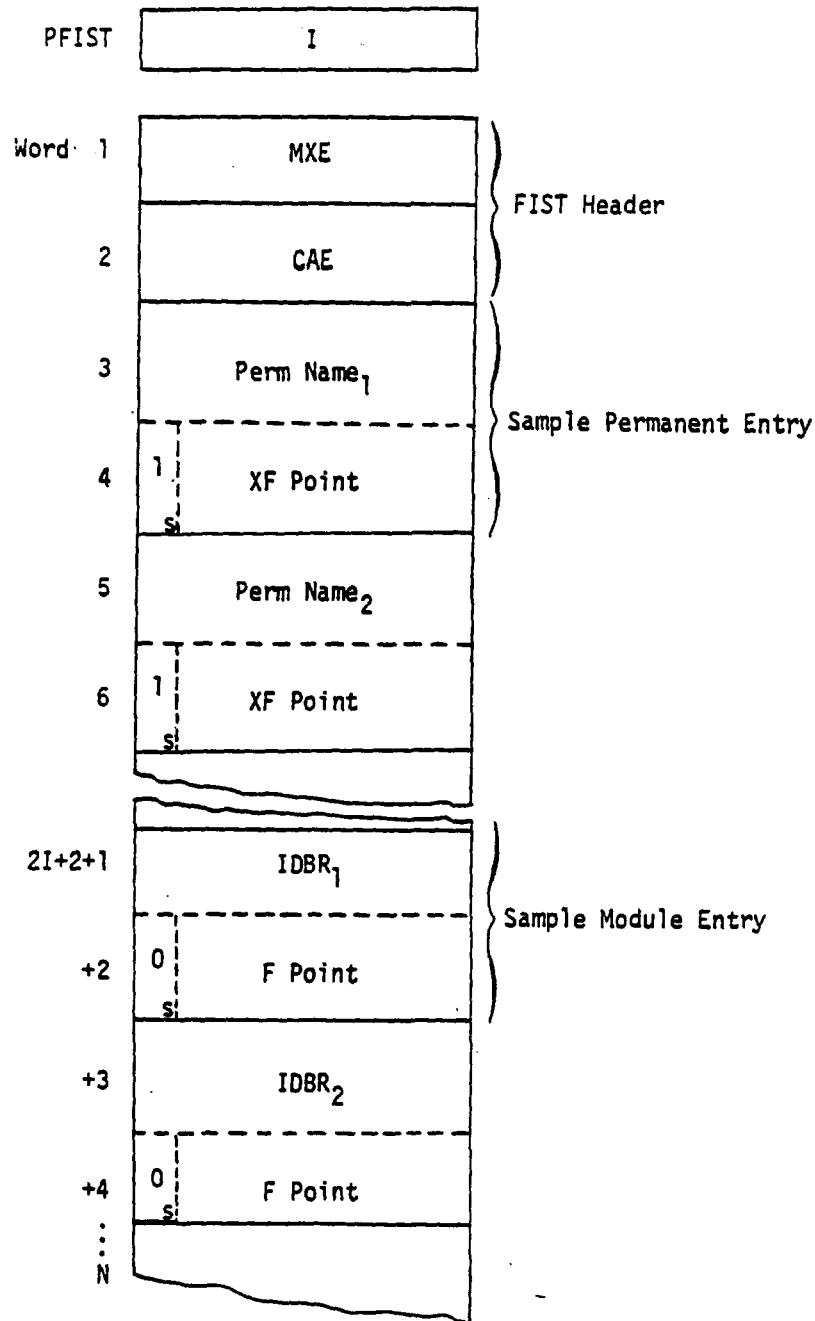
Description

A NASTRAN resident memory table containing the internal data block reference (IDBR) numbers vs. FIAT table pointers for a particular module; also, the permanent file reference names vs. XFIAT table pointers.

Created in Module

The module entries are generated prior to each module execution by subroutine GNFIST (Generate FIST). The permanent entries are initialized at system assembly.

Table Format



DATA BLOCK AND TABLE DESCRIPTIONS

<u>Word</u>	<u>Item</u>	<u>Description</u>
PFIST	I	Integer number of permanent FIST entries.
1	MXE	Maximum entries - this integer shows the total entry size of the dimensioned FIST table; the amount of memory reserved (N) = 2 * MXE + 2.
2	CAE	Current active entries - this integer designates the portion of FIST currently containing valid data; $I \leq CAE \leq MXE$.

Words 3 and 4 describe a sample 2-word permanent entry:

3	Perm Name	A permanent file reference name - 4 characters BCD (e.g., PØØL, ØPTP, PLT1, etc.).
4	XF Point	Points to the XFIAT position containing the file ID for this permanent file.

Words 2I+2+1 and +2 describe a sample 2-word module entry.

+1	IDBR	An internal data block reference number (GINØ file number) - integer (e.g., 104, 206, 301, etc.).
+2	F Point	Points to the FIAT position containing the file ID for this module entry.

Notes

1. XFIAT pointer values contain an S bit equal to 1, while FIAT pointer values contain an S bit equal to 0.
2. Permanent entries remain static throughout a run, while module entries are changed by GNFIST prior to each module call.
3. FIAT and XFIAT position pointers are indexes into the respective tables considering the first word of the table as position 0.
4. The FIST table is located in the named common block /XFIST/.
The PFIST entry is located in the named common block /XPFIST/.

EXECUTIVE TABLE DESCRIPTIONS

2.4.1.4 DPL (Data Pool Dictionary)

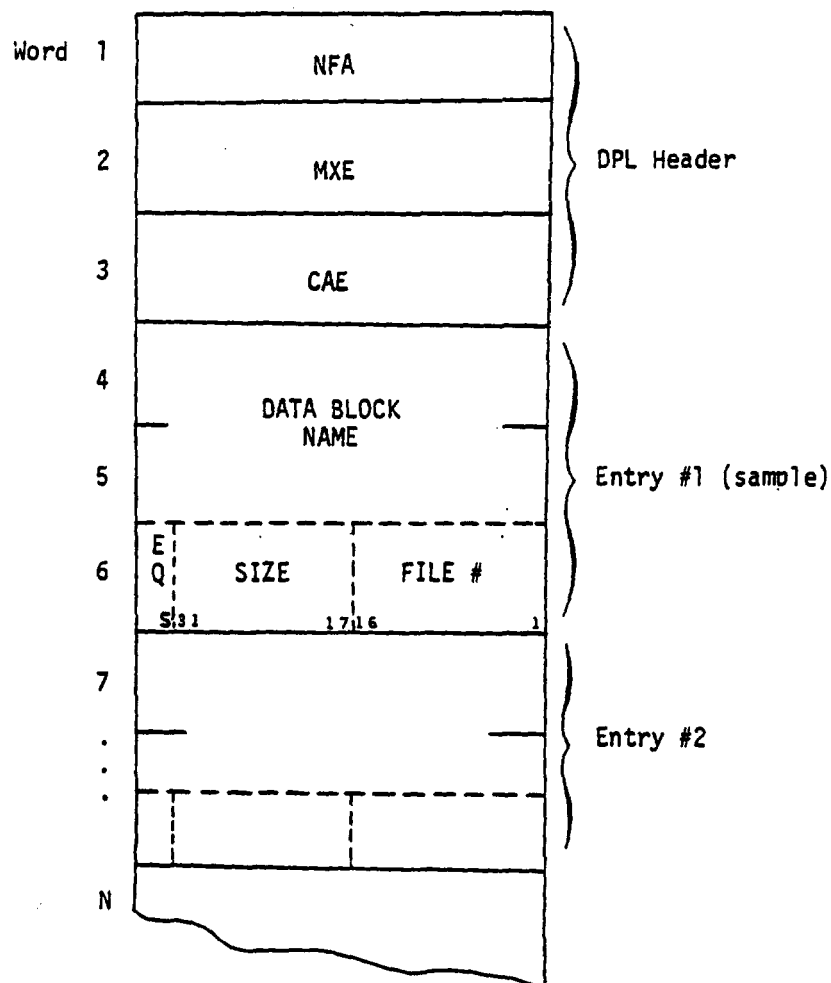
Description

A NASTRAN resident memory table describing the current contents and status of the Data Pool.

Created in Module

Data Pool, and therefore Dictionary entries, are created by pooling from SFA (Segment File Allocator), housekeeping operations by DPH (Data Pool Housekeeper) and restart initialization by GPI (General Problem Initialization), and IFP (Input File Processor) when writing DMI and DTI information (see section 2.3.2).

Table Format



DATA BLOCK AND TABLE DESCRIPTIONS

<u>Word</u>	<u>Item</u>	<u>Description</u>
1	NFA	Next file available - the next Data Pool File number (integer) available for output.
2	MXE	Maximum entries - this integer shows the total entry size of the dimensioned DPL table; the amount of memory reserved (N) = 3 * MXE + 3.
3	CAE	Current active entries - this integer designates the portion of the DPL currently containing valid data; $0 \leq CAE \leq MXE$.

Words 4 through 6 describe a sample 3-word entry.

4,5	NAME	Data block name - 8 characters (4 characters/word).
6	EQ	Equivalence flag (1 bit) - 0 bit, file not equivalenced. 1 bit, file equivalenced.
	SIZE	Size of the pooled data block - number of words/1000.
	FILE#	The file number (integer) showing the relative position of the data block file of the pool.

Note

The DPL table is located in the labeled common block /XDPL/.

EXECUTIVE TABLE DESCRIPTIONS

2.4.1.5 CEITBL (Control Entry Information Table)

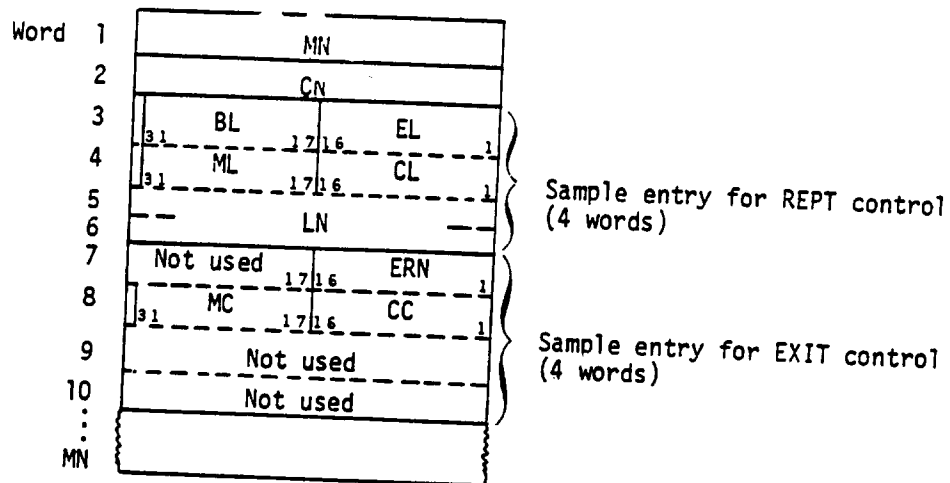
Description

CEITBL controls the REPT and EXIT DMAP module execution.

Created In Module

XGPI.

Table Format



Word	Item	Description
1	MN	Maximum number of words in table (integer).
2	CN	Current number of words being used (integer).
3	BL	ØSCAR record number where loop begins (integer).
	EL	ØSCAR record number where loop ends (integer).
4	ML	Maximum loop count as specified in REPT instruction (integer).
	CL	Current loop count, that is, the number of times loop has been repeated (integer).
5,6	LN	Location name specified in REPT instruction (BCD).
7	ERN	EXIT ØSCAR record number (integer).
8	MC	Maximum count specified in EXIT instruction.
	CC	Current count of number of times EXIT instruction not executed.
9,10		These two words are zeroed.

Notes

CEITBL is located in named common block /XCEITB/.

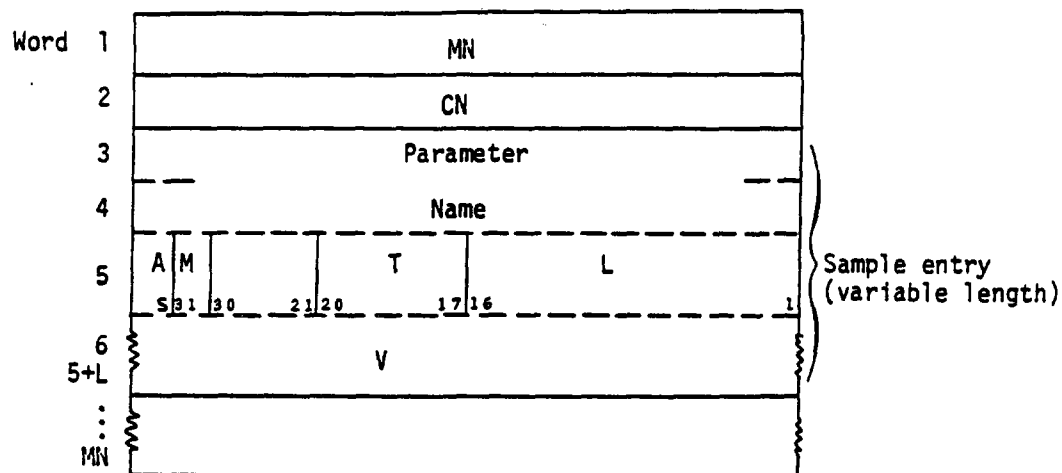
2.4.1.6 VPS (Variable Parameter Set Table)

Description

The VPS table contains the values of all variable parameters referenced by DMAP modules in a DMAP program. It is the means for transferring variable parameter values from one module to another.

Created in Module

XGPI.

Table Format

<u>Word</u>	<u>Item</u>	<u>Description</u>
1	MN	Maximum number of words in VPS (integer).
2	CN	Current number of words being used (integer).
3,4	Name	BCD name of variable parameter.
5	A	Assigned flag. A = 1 indicates value from DMAP instruction has been entered in VPS.
	M	Modified flag. M = 1 indicates parameter was modified by bulk data PARAM card on restart.
	T	Type code for parameter (integer).
	L	Length in words of item V (integer).
6 thru 5+L	V	Value of parameter.

Notes

1. Items A, M and T (word 5 of sample entry) are used only by the XGPI module and are cleared prior to exiting XGPI.
2. Type code and corresponding word length.

<u>I</u>	<u>L</u>
1 = integer	1
2 = real, S.P.	1
3 = BCD	2
4 = real, D.P.	2
5 = complex, S.P.	2
6 = complex, D.P.	4

3. The VPS table is located in the named common block /XVPS/.

2.4.1.7 TAPID (Problem Tape Identification Table)

Description

TAPID contains Problem Tape identification information.

Created in Module

XCSA.

Table Format

Word	1	ID ₁		
	2			
	3	ID ₂		
	4			
	5	M	D	Y
	6	R		

<u>Word</u>	<u>Item</u>	<u>Description</u>
1,2	ID ₁	First BCD field on ID Executive Control card.
3,4	ID ₂	Second BCD field on ID Executive Control card.
5	M,D,Y	The date - integers - month, day, year.
6	R	Reel number of Problem Tape (integer).

Notes

1. TAPID is written on Problem Tape as single record field. It is always the first file on the Problem Tape.
2. ØTAPID has same format as TAPID. ØTAPID is the ID information from an Old Problem Tape being used for restarting problem.
3. TAPID and ØTAPID are located in named common block /STAPID/.

EXECUTIVE TABLE DESCRIPTIONS

2.4.1.8 SYSTEM (NASTRAN System Parameters).

Description

A NASTRAN resident memory table containing various machine dependent, operating system and NASTRAN parameters. The length of the table is defined by one of the table items.

Created in Module

Most items are initialized by the NASTRAN block data program, SEMDBD. Several machine dependent items are set by subroutine BTSTRP.

Table Format

The sequential table description follows:

<u>Word</u>	<u>Symbol</u>	<u>Description</u>	<u>Initially Set by</u>
1	SYSBUF	Number of words in a GINØ buffer	BTSTRP
2	ØUTTAP	FØRTRAN logical unit for output	BTSTRP
3	NØGØ	Flag defining status during Preface	SEMDBD
4	INTP	FØRTRAN logical unit for input	BTSTRP
5	MPC	Multipoint constraint set ID	SEMDBD
6	SPC	Single-point constraint set ID	SEMDBD
7	LØGETR	Maximum number of entries to place in Log file "0" implies unlimited	SEMDBD
8	LØAD	First record pointer in Case Control data block	SEMDBD
9	NLPP	Number of lines per page of printed output	BTSTRP
10	MTEMP	Material temperature set ID	SEMDBD
11	NPAGES	Current page count	SEMDBD
12	NLINES	Number of lines on current page	SEMDBD
13	TLINES	Total number of lines printed for problem	SEMDBD
14	MXLINS	Maximum number of printed lines permitted	SEMDBD
15	DATE(1)	Today's date - integer month 1-12	SEMDBD
16	DATE(2)	Today's date - integer day 1-31	SEMDBD
17	DATE(3)	Today's date - integer year (XX)	SEMDBD
18	TIMEZ	CPU time of problem start - seconds (after midnite for UNIVAC)	SEMDBD
19	ECHØF	Bulk data output request type	SEMDBD
20	PLØTF	Structural plot request flag	SEMDBD
21	APPRCH	Approach type flag (2 = DISPL, 3 = DMAP)	SEMDBD
22	MACH	Computing machine code number (2 = 360, 3 = 1108, 4 = 6600)	BTSTRP
23	LSYSTM	Length of this table	SEMDBD
24	EDTUMF	User master file edit flag	SEMDBD
25		Not used	
26	CPPGCT	XCHK module page count	SEMDBD
27	MN	Used only in a conical shell problem. The lower order 16 bits contain N, the number of harmonics; the next higher order 16 bits contain M, the number of rings.	SEMDBD
28	ICØNFG	Machine configuration - subset of MACH code number	SEMDBD
29	MAXFIL	Maximum number of files to be added to FIAT	SEMDBD
30	MAXØPN	Maximum number of files to be opened simultaneously	SEMDBD
31	HICØRE	Length of core on UNIVAC	
32	TIMEW	Wall clock initial problem start time (integer seconds after midnite for UNIVAC)	SEMDBD
33	ØFPFLG	ØFP operate flag - set nonzerp when ØFP operates	SEMDBD

DATA BLOCK AND TABLE DESCRIPTIONS

<u>Word</u>	<u>Symbol</u>	<u>Description</u>	<u>Initially Set by</u>
34	NBRCBU	Length of FET + circular buffer (CDC 6600 only)	SEMDBD
35	NBRMST	Length of master index (CDC 6600 only)	SEMDBD
36	NBRSUB	Length of subindex (CDC 6600 only)	SEMDBD
37	KSEMTR	Input Data Transliteration Flag (IBM 360 only)	SEMDBD
38	QQ	Hydroelastic Problem Flag	SEMDBD
39	NBPC	Number of bits per character	BTSTRP
40	NBPW	Number of bits per word	BTSTRP
41	NCPW	Number of characters per word	BTSTRP
42	SYSDAT(1)	System Generation Date - Month	TTLPGE
43	SYSDAT(2)	System Generation Date - Day	TTLPGE
44	SYSDAT(3)	System Generation Date - Year	TTLPGE
45	TAPFLG	Permanent File Tape Flag	SEMDBD
46	ADUMEL(1)	Dummy Element Flag - DUM1	SEMDBD
47	ADUMEL(2)	Dummy Element Flag - DUM2	SEMDBD
48	ADUMEL(3)	Dummy Element Flag - DUM3	SEMDBD
49	ADUMEL(4)	Dummy Element Flag - DUM4	SEMDBD
50	ADUMEL(5)	Dummy Element Flag - DUM5	SEMDBD
51	ADUMEL(6)	Dummy Element Flag - DUM6	SEMDBD
52	ADUMEL(7)	Dummy Element Flag - DUM7	SEMDBD
53	ADUMEL(8)	Dummy Element Flag - DUM8	SEMDBD
54	ADUMEL(9)	Dummy Element Flag - DUM9	SEMDBD
55	IPREC	Precision Flag	SEMDBD
56	ITHRML	Heat Transfer Flag	SEMDBD
57	MØDCØM	Module Communication Region	SEMDBD
58	MØDCØM	Module Communication Region	SEMDBD
59	NTRCK	Set = 1 for 7-track plot tape Set = 2 for 9-track plot tape	SEMDBD
60	MØDCØM	Module Communication Region	SEMDBD
61	MØDCØM	Module Communication Region	SEMDBD
62	MØDCØM	Module Communication Region	SEMDBD
63	MØDCØM	Module Communication Region	SEMDBD
64	MØDCØM	Module Communication Region	SEMDBD
65	MØDCØM	Module Communication Region	SEMDBD
66	HDY		SEMDBD
67	HDY		SEMDBD
68	HDY		SEMDBD
69	SSCEL	Multi-level substructuring flag	SEMDBD
70	TØLEL	SMA1, EMG singularity tolerance (real)	SEMDBD
71	MESDAY	= 0 messages printed on Log file = 1 messages printed on Output = -1 messages are not printed Used by subroutine CØNMSG	SEMDBD
72	BITPASS	= .TRUE. indicates GNFIAT has been executed once	SEMDBD
73	PASS	= .TRUE. indicates LINK1 has been executed	CØNMSG
74	Unused		
75	Unused		
76	NØSLINK	= 0 use NØS TCS PP call for link switching = 1 use NØS-BE CCL file	
77	Unused		
78	Unused		
79	DIAG1	Bits 0-31 turned on for DIAGs 1 to 32	
80	DIAG2	Bits 32-63 turned on for DIAGs 33 to 64	

EXECUTIVE TABLE DESCRIPTIONS

2.4.1.9 XLINK (Link Specification Table - Non-resident Edit)

Description

This Link Specification Table (see also 2.4.2.7) contains an entry corresponding to each DMAP module within the MPL (2.4.2.2) table. These entries are indexed by MPL position and are thus ordered the same as the MPL entries. Each entry contains a key indicating the links in which the module resides.

Created in Module

XLINK data is created from the LNKSPC (2.4.2.7) and MPL (2.4.2.2) tables by the XGPBS subroutine within the XGPI (4.7) module.

Table Format

Word	1	LXLINK	
	2	MAXLNK	
	3	Key	Entry #1 (sample)
	.	Key	Entry #2
	.	Key	Entry #3
	N		

<u>Word</u>	<u>Item</u>	<u>Description</u>
1	LXLINK	Length of table (number of entries)
2	MAXLNK	Maximum permissible link number
3,N	Key	Link residence key for the corresponding MPL entry

The content of this Key word is identical to the Key word within LNKSPC (2.4.2.7) for the machine type currently operating. See section 2.4.2.7 for an explanation of the content.

Notes

1. The XLINK table must contain an entry in the same order for each module that is in the MPL (2.4.2.2) table.
2. XLINK table is located in /XLINK/.

DATA BLOCK AND TABLE DESCRIPTIONS

2.4.2 Executive Tables Not Permanently Core Resident

2.4.2.1 ØSCAR (Operation Sequence Control Array)

Description

The Operation Sequence Control Array (ØSCAR) controls the sequence of modules executed and aids in communicating data between modules.

The ØSCAR is generated from a DMAP instruction sequence supplied by the user or selected from the Rigid Format library. In general, an ØSCAR entry is a DMAP statement which has been translated to a more readily usable form for internal use.

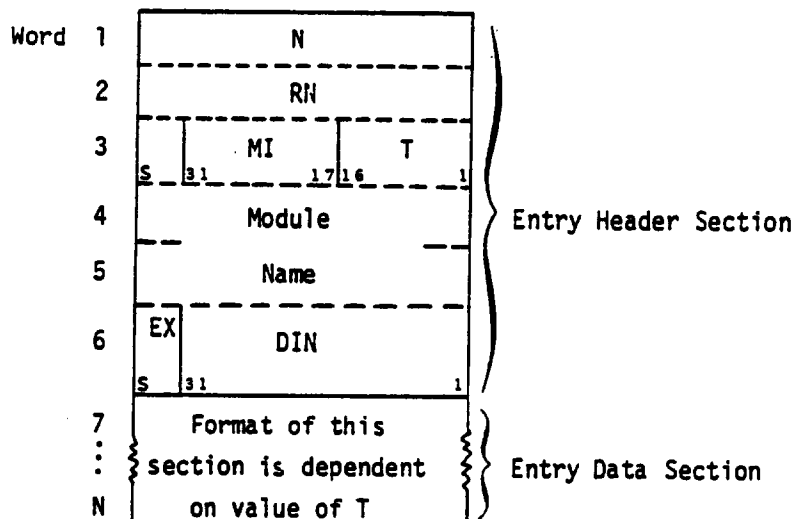
The four ØSCAR entry format types are:

1. Type 1 or F (functional) format is used for all functional modules except output processors.
2. Type 2 or Ø (output) format is used for output processors.
3. Type 3 or C (control) format is used for REPT, JUMP, CØND and END DMAP instructions.
4. Type 4 or E (executive) format is used for SAVE, CHPNT, PURGE and EQUIV DMAP instructions.

Created in Module

XGPI.

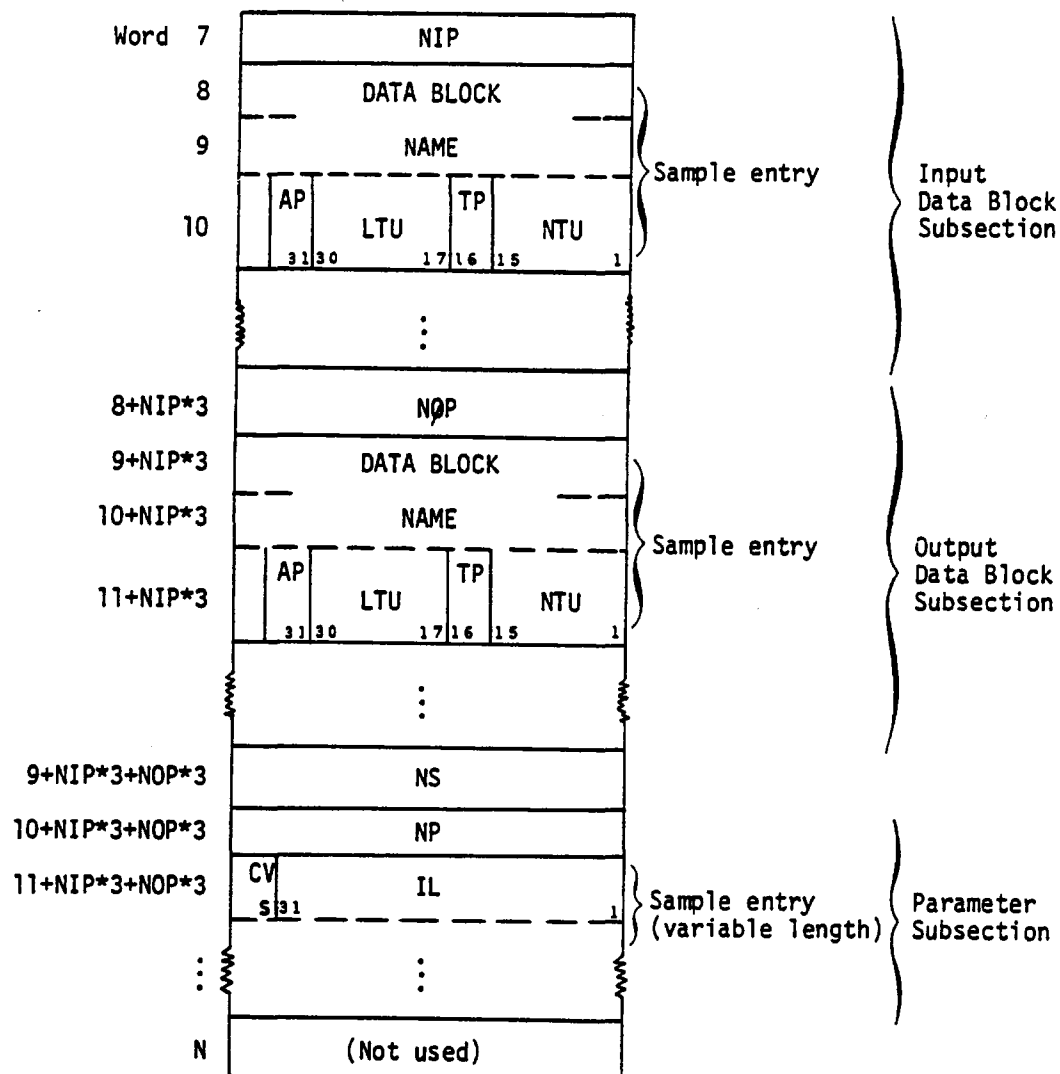
Table Format



EXECUTIVE TABLE DESCRIPTIONS

Word	Item	Description
1	N	Integer - number of words in entry.
2	RN	Integer - record number of entry in ØSCAR table.
3	MI	Integer - module index number assigned according to module's position in MPL and used to access the module's link specifications in /XLINK/.
	T	Integer - format type code (1, 2, 3, or 4) for data section of entry.
4,5	Name	BCD - module name is same as DMAP instruction name except when T = 4.
6	EX	Execute flag. EX = 1 indicates module is to be executed.
	DIN	Integer - DMAP instruction number which generated this entry.

Data Section Format for Type 1 or F Format:



DATA BLOCK AND TABLE DESCRIPTIONS

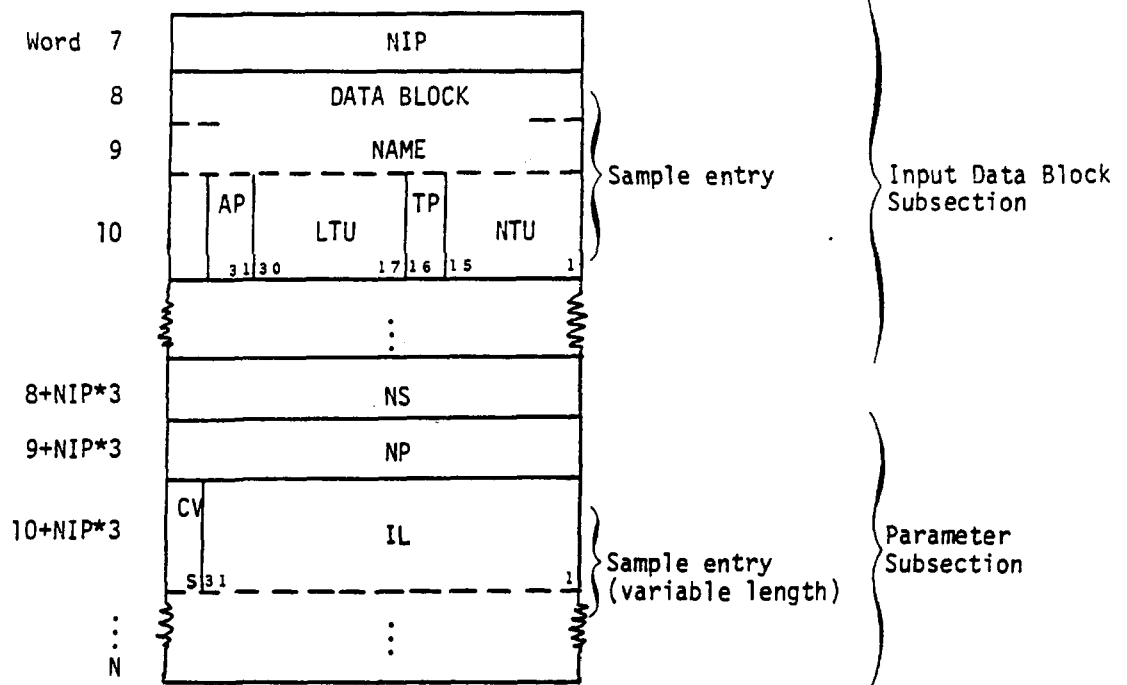
<u>Word</u>	<u>Item</u>	<u>Description</u>
7	NIP	Integer - number of data blocks input to module as specified in MPL.
8,9	NAME	BCD - name of first input data block specified in DMAP instruction or zero if none specified.
10	AP	Append flag used by subroutine XSFA and set by XØSGEN if APPEND is specified for data block in a FILE DMAP instruction.
	LTU	Integer - last time used. ØSCAR record number of entry after which data block will no longer be saved.
	TP	Tape flag used by subroutine XSFA and set by XØSGEN if tape is specified for data block in a FILE DMAP instruction.
	NTU	Integer - next time used. ØSCAR record number of entry where data block is next referenced.
8 + NIP*3	NØP	Integer - number of data blocks output from module as specified in MPL.
9 + NIP*3, 10 + NIP*3	NAME	BCD - name of first output data block specified in DMAP instruction or zero if none specified.
11 + NIP*3	AP,LTU, TP,NTU	Same descriptions as word 10.
9 + NIP*3 + NØP*3	NS	Integer - number of scratch data blocks used by module as specified in MPL.
10 + NIP*3 + NØP*3	NP	Integer - number of parameters used by module as specified in MPL.
11 + NIP*3 + NØP*3	CV	Constant/variable flag. Flag indicates meaning of IL.
	IL	Integer - VPS index/length of constant. If CV = 0 the parameter is a constant whose value is stored in the next IL words (i.e., words 12 + NIP*3 + NØP*3 through 11 + NIP*3 + NØP*3 + IL). If CV = 1 the parameter is a variable whose value is stored in the VPS table. IL points to the value in VPS.

Remarks:

1. A module with 0 input files in the MPL is given 1 null input file (i.e., NIP=1, NAME=(0,0), AP/LTU/TP/NTU=0) by XSØSGN.
2. NØP must be non-zero otherwise the format type (T) is changed to 2 by XSØSGN.
(see Type 2 format)

EXECUTIVE TABLE DESCRIPTIONS

Data Section Format for Type 2 or O Format:



Type 1 format description is applicable to type 2 format above.

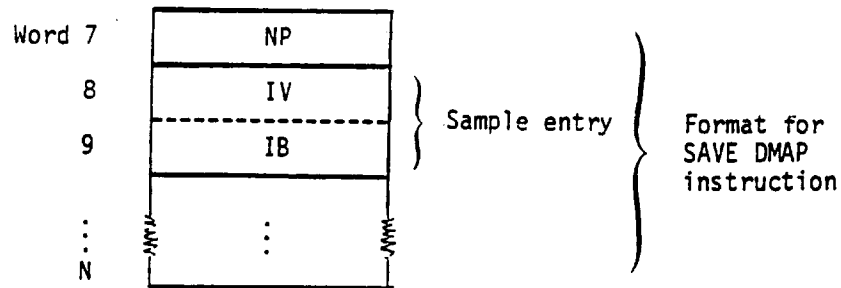
Data Section Format for Type 3 or C Format:



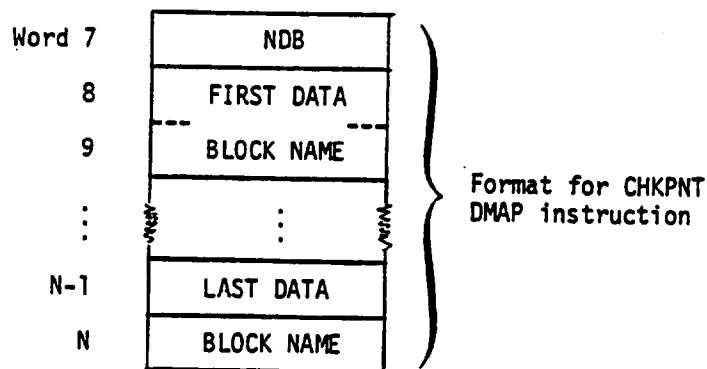
Word	Item	Description
N	RRN	Integer - re-entry record number. Indicates OSCAR record to jump to for JUMP, REPT and COND DMAP instructions. Not applicable for EXIT so RRN = 0.
	I	Integer - index into XCEITBL for REPT or EXIT DMAP instruction. Pointer to parameter value in XVPS table if COND DMAP instruction. Not applicable for JUMP so I = 0.

EXECUTIVE TABLE DESCRIPTIONS

Data Section Formats for Type 4 or E Formats:

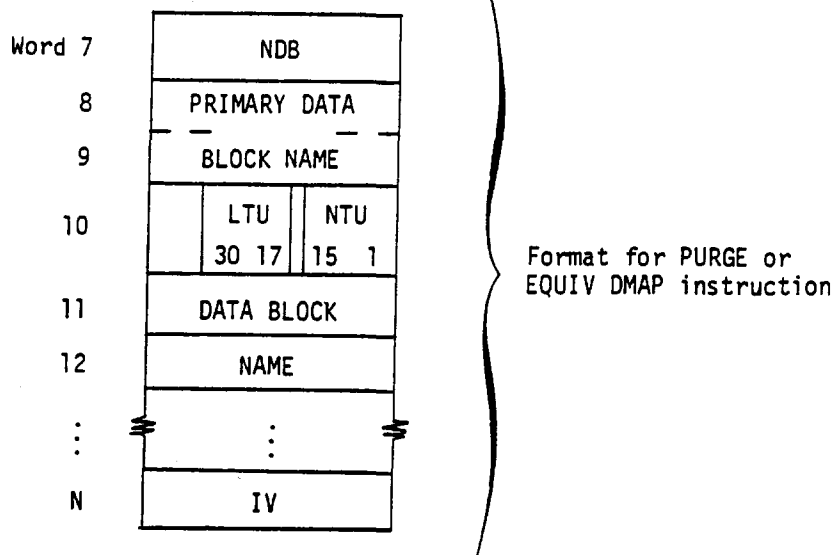


<u>Word</u>	<u>Item</u>	<u>Description</u>
7	NP	Integer - number of parameter values to be saved (i.e., number of entries).
8	IV	Integer - pointer to parameter value in VPS.
9	IB	Integer - pointer to parameter value in blank common.



<u>Word</u>	<u>Item</u>	<u>Description</u>
7	NDB	Integer - number of data blocks names in list.
8 thru N	NAMES	BCD - list of data blocks to be checkpointed.

DATA BLOCK AND TABLE DESCRIPTIONS



<u>Word</u>	<u>Item</u>	<u>Description</u>
7	NDB	Integer - number of data block names in first group. There may be one or more groups.
8,9	NAME	BCD - name of primary data block for first group.
10	LTU NTU	Same as Type 1 Format.
N	IV	Integer - pointer to parameter value in VPS table.

DATA BLOCK AND TABLE DESCRIPTIONS

THIS PAGE HAS BEEN LEFT BLANK INTENTIONALLY.

EXECUTIVE TABLE DESCRIPTIONS

Notes

1. ØSCAR is located in named common block /XGPI1/ while module XGPI is generating it.
2. After generating ØSCAR and prior to exiting XGPI the ØSCAR is written on the Data Pool File. The ØSCAR file header ID is XØSCAR.
3. By setting DIAG 4 or requesting the ØSCAR option on the XDMAP DMAP statement, a detailed ØSCAR listing may be obtained. Each of the four types of ØSCAR records is processed and printed according to its own format. Common header information is printed for all DMAP instructions. The format of the output is:

a) Header Data

ØSCAR record number, module type, module name and DMAP instruction number.

b) Type 1 and 2 Formats

Input and output data blocks - for each input data block the name and ØSCAR attributes are given in the format:

NAME AP/LTU/TP/NTU

where these symbols are as defined above. A null data block is indicated.

DATA BLOCK AND TABLE DESCRIPTIONS

2.4.2.2 MPL (Module Property List)

Description

The Module Properties List contains information needed by the module XGPI to correctly generate ØSCAR table entries for executable DMAP instructions and/or to determine whether or not the DMAP instructions adhere to the calling sequence described in section 4, Module Functional Descriptions.

Created in Module

XGPI (Block Data Program XMPLBD).

Table Format

There are two formats used in the MPL, one for Declarative (FILE, BEGIN, LABEL), Executive (CHKPNT, EQUIV, PURGE, SAVE) and Control (REPT, JUMP, COND, EXIT, END) DMAP modules and the other for functional modules. All entries in the MPL are integer except for module names which are BCD and BCD parameter values. Modules are in alphabetical order with the exception of substructuring modules which follow module ZMCE2.

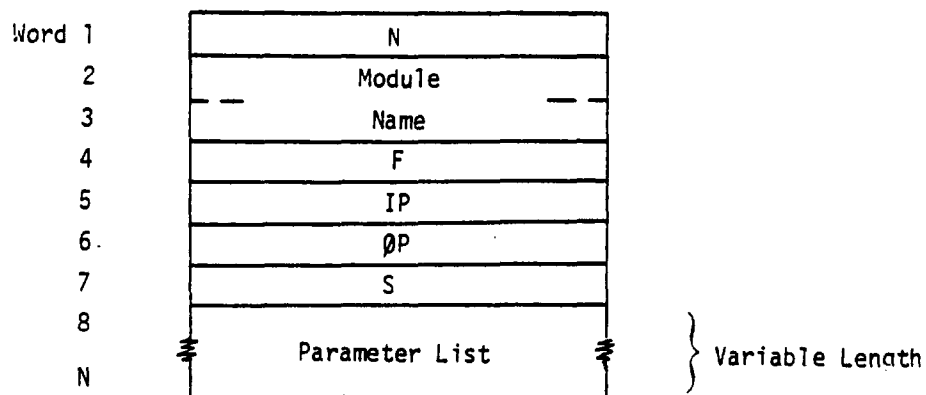
Format for Declarative, Executive and Control Modules:

Word 1	N = 4
2	Module
3	Name
4	F

<u>Word</u>	<u>Item</u>	<u>Description</u>
1	N	Number of words in entry.
2,3	Name	Name of DMAP module.
4	F	ØSCAR format type code 3 = Control module (C format) 4 = Executive module (E format) 5 = Declarative module (D format)

EXECUTIVE TABLE DESCRIPTIONS

Format for Functional Modules:



<u>Word</u>	<u>Item</u>	<u>Description</u>
1-3		Same as format for Declarative, Executive and Control modules.
4	F	F = 1 implies module has input and output data blocks F = 2 implies module has no output data blocks, e.g., ØFP, SETVAL etc.
5	IP	Number of input data blocks.
6	OP	Number of output data blocks.
7	S	Number of scratch data blocks.
8-N		Parameter List (omit if no parameters).

The parameter list for a module contains the types of all parameters residing in blank common that are referenced by the module. The order of the parameters in the MPL entry must coincide with the order of the parameters in blank common. Space must be allowed for a default value if the parameter type code is positive. The space following a positive type code will contain the default value if the type code is integer or BCD, otherwise the space will contain an index into another table which contains the default value.

<u>Type Code</u>	<u>Space Needed for Default Values</u>
1 = integer	1 word
2 = real, single precision	1 word
3 = BCD	2 words
4 = real, double precision	2 words
5 = complex, single precision	2 words
6 = complex, double precision	4 words

DATA BLOCK AND TABLE DESCRIPTIONS

An example of all possible entries in a parameter list follows. Note that for each parameter only the first index word will appear in the XMPLBD Block Data subprogram.

Word	
1	8 = Integer type code.
-3	9 = Integer default value.
-1	10 = Integer type code (no default value).
2	11 = Real, S.P. type code.
1	12 = Index into table containing a real S.P. default value.
-2	13 = Real, S.P. type code (no default value).
3	14 = BCD type code.
ABCD	15 = {BCD default value (2 words).
EFGH	16 = {BCD type code (no default value).
-3	17 = BCD type code (no default value).
4	18 = Real, D.P. type code.
1	19 = {Index into table containing a real D.P. default value.
1	20 = {Note index is in both words.
-4	21 = Real, D.P. type code (no default value).
5	22 = Complex, S.P. type code.
2	23 = {Index into table containing a complex S.P. default value.
2	24 = {Complex, S.P. type code (no default value).
-5	25 = Complex, S.P. type code (no default value).
6	26 = Complex, D.P. type code.
3	27 = {Index into table containing the real part of the complex D.P. default value.
3	28 = {Index into table containing the imaginary part of the complex D.P. default
4	29 = {value
4	30 = {Complex, D.P. type code (no default value).
-6	31 = Complex, D.P. type code (no default value).

Notes

1. MPL table is located in named common block /XGPI2/.
2. The default value table is located in named common block /XGPI2X/.
3. The following printout, generated automatically by NASTRAN subroutine MPLPRT when DIAG 31 is present, presents the contents of XMPLBD in readable format. The meaning of the various entries is given below.

<u>Item</u>	<u>Description</u>
MPLID	Module identification number
NWDS	Number of words for this entry
WD1	Pointer to the first word of the entry in /XGPI1/
M0D-NAME	Module Name
TYP	Module Type
IN	Number of Input Data Blocks for module
OUT	Number of Output Data Blocks for Module
SCR	Number of Scratch Data Blocks for Module

EXECUTIVE TABLE DESCRIPTIONS

<u>Item</u>	<u>Description</u>
TØT	Sum of IN, ØUT and SCR; this item will be flagged with the symbol *** if the allowable maximum number of total files is exceeded. Depending on the design and usage of the module, this may or may not result in execution time failure.

For functional modules having parameters, the following information is presented:

<u>Item</u>	<u>Description</u>
ID	Parameter number for module
TYP	BCD - Alpha numeric INT - Integer RSP - Real, Single-Precision RDP - Real, Double-Precision CSP - Complex, Single-Precision CDP - Complex, Double-Precision
P	Pointer to entry in /XGP11/
DEFAULT	MPL default value (if any)
W1-W2	Blank common indices for first and last words of parameter
FLG	The symbol *** will appear whenever a parameter of type CSP, RDP or CDP begins on an even numbered word of blank common. This can cause boundary alignment problems if the data is directly addressed by variables having the types indicated.

MODULE PROPERTIES LIST

MPLID	NWDS	WD1	MOD-NAME	TYP	IN	OUT	SCR	TOT	PARAMETERS				W1-W2	FLG
									ID	TYP	P	DEFAULT (IF ANY)		
1	4	1	FILE	5										
2	4	5	REGIN	5										
3	4	9	CHKPNT	4										
4	4	13	LABEL	5										
5	4	17	REPT	3										
6	4	21	JUMP	3										
7	4	25	COND	3										
8	4	29	SAVE	4										
9	4	33	PURGE	4										
10	4	37	EQUIV	4										
11	4	41	END	3										
12	4	45	EXIT	3										
13	20	49	(NONE)											
14	13	69	ADD	1	2	1	0	3	1. CSP	76	(1.0000E+00, 0.)	1- 2
									2. CSP	79	(1.0000E+00, 0.)	3- 4
15	22	82	ADD5	1	5	1	0	6	1. CSP	89	(1.0000E+00, 0.)	1- 2
									2. CSP	92	(1.0000E+00, 0.)	3- 4
									3. CSP	95	(1.0000E+00, 0.)	5- 6
									4. CSP	98	(1.0000E+00, 0.)	7- 8
									5. CSP	101	(1.0000E+00, 0.)	9-10

DATA BLOCK AND TABLE DESCRIPTIONS

M O D U L E P R O P E R T I E S L I S T									P A R A M E T E R S					
MPLID	NWDS	WD1	MOD-NAME	TYP	IN	OUT	SCR	TOT	ID	TYP	P	DEFAULT (IF ANY)	W1-W2	FLG
16	10	104	AMG	1	2	4	5	11						
									1.	INT	111	-- NO DEFAULT --	1	
									2.	INT	112	-- NO DEFAULT --	2	
									3.	INT	113	-- NO DEFAULT --	3	
17	11	114	AMP	1	10	3	14	27						
									1.	INT	121	-- NO DEFAULT --	1	
									2.	INT	122	-- NO DEFAULT --	2	
									3.	INT	123	-1	3	
18	12	125	APD	1	8	12	5	25						
									1.	INT	132	-- NO DEFAULT --	1	
									2.	INT	133	-- NO DEFAULT --	2	
									3.	INT	134	-- NO DEFAULT --	3	
									4.	RSP	135	0.	4	
19	10	137	RMG	1	4	1	1	6						
									1.	INT	144	-- NO DEFAULT --	1	
									2.	INT	145	-- NO DEFAULT --	2	
									3.	CSP	146	-- NO DEFAULT --	3- 4	
20	12	147	CASE	1	2	1	0	3						
									1.	RCD	154	-- NO DEFAULT --	1- 2	
									2.	INT	155	1	3	
									3.	INT	157	-1	4	
21	15	159	CYCT1	1	1	2	3	6						
									1.	BCD	166	-- NO DEFAULT --	1- 2	
									2.	RCD	167	-- NO DEFAULT --	3- 4	
									3.	INT	168	-- NO DEFAULT --	5	
									4.	INT	169	-- NO DEFAULT --	6	
									5.	INT	170	1	7	
									6.	INT	172	1	8	

2.4-28 (12/29/78)

MODULE PROPERTIES LIST									PARAMETERS					
MPLID	NWDS	WD1	MND-NAME	TYP	IN	OUT	SCR	TOT	ID	TYP	P	DEFAULT (IF ANY)	W1-W2	FLG
22	16	174	CYCT2	1	6	5	6	17	1.	BCD	181	-- NO DEFAULT --	1- 2	
									2.	INT	182	-- NO DEFAULT --	3	
									3.	INT	183	-- NO DEFAULT --	4	
									4.	INT	184	-1	5	
									5.	INT	186	1	6	
									6.	INT	188	1	7	
23	10	190	CFAD	1	5	4	12	21	1.	INT	197	-- NO DEFAULT --	1	
									2.	INT	198	1	2	
24	11	200	CURV	1	6	2	5	13	1.	INT	207	-1	1	
									2.	INT	209	0	2	
25	9	211	(NONE)											
26	10	220	DDR	1	1	1	0	2	1.	BCD	227	-- NO DEFAULT --	1- 2	
									2.	BCD	228	-- NO DEFAULT --	3- 4	
									3.	BCD	229	-- NO DEFAULT --	5- 6	
27	7	230	DDR1	1	2	1	1	4	----- NO PARAMETERS EXIST -----					
28	14	237	DDR2	1	9	3	6	18	1.	BCD	244	-- NO DEFAULT --	1- 2	
									2.	INT	245	-1	3	
									3.	INT	247	-1	4	
									4.	INT	249	-1	5	
29	7	251	DDRMM	1	11	5	7	23	----- NO PARAMETERS EXIST -----					

DATA BLOCK AND TABLE DESCRIPTIONS

MODULE PROPERTIES LIST									PARAMETERS				W1-W2 FLG	
MPLID	NWDS	WD1	MOD-NAME	TYP	IN	OUT	SCK	TOT	ID	TYP	P	DEFAULT (IF ANY)		
30	21	258	DECOMP	1	1	2	4	7	1.	INT	265	0	1	
									2.	INT	267	0	2	
									3.	RDP	269	0.	3- 4	
									4.	CSP	272	(0. , 0.)	5- 6	
									5.	INT	275	0	7	
									6.	INT	277	0	8	
31	12	279	DIAGONAL	1	1	1	0	2	1.	RCD	286	COLUMN	1- 2	
									2.	RSP	289	1.0000E+00	3	
32	19	291	OPD	1	4	11	4	19	1.	INT	298	-- NO DEFAULT --	1	
									2.	INT	299	-- NO DEFAULT --	2	
									3.	INT	300	-- NO DEFAULT --	3	
									4.	INT	301	-- NO DEFAULT --	4	
									5.	INT	302	-- NO DEFAULT --	5	
									6.	INT	303	-- NO DEFAULT --	6	
									7.	INT	304	-- NO DEFAULT --	7	
									8.	INT	305	-- NO DEFAULT --	8	
									9.	INT	306	-- NO DEFAULT --	9	
									10.	INT	307	1	10	
									11.	INT	309	-- NO DEFAULT --	11	
33	16	310	DSCHK	1	3	0	3	6	1.	RSP	317	-- NO DEFAULT --	1	
									2.	RSP	318	-- NO DEFAULT --	2	
									3.	INT	319	-- NO DEFAULT --	3	
									4.	INT	320	-- NO DEFAULT --	4	
									5.	INT	321	-- NO DEFAULT --	5	
									6.	INT	322	-- NO DEFAULT --	6	
									7.	INT	323	-- NO DEFAULT --	7	
									8.	INT	324	-- NO DEFAULT --	8	
									9.	INT	325	-- NO DEFAULT --	9	

2.4-30 (12/29/78)

MODULE PROPERTIES									TEST PARAMETERS					
MPLID	NWDS	WD1	MOD-NAME	TYP	IN	OUT	SCR	TOT	ID	TYP	P	DEFAULT (IF ANY)	W1-W2	FLG
34	8	326	DSMG1	1	10	1	1	12	1.	INT	333	-- NO DEFAULT --	1	
35	11	334	DSMG2	1	11	7	0	18	1.	INT	341	0	1	
									2.	INT	343	-- NO DEFAULT --	2	
									3.	INT	344	-- NO DEFAULT --	3	
36	20	345	(NONE)											
37	33	365	DUMMOD1	1	8	8	10	26	1.	INT	372	-1	1	
									2.	INT	374	-1	2	
									3.	INT	376	-1	3	
									4.	INT	378	-1	4	
									5.	PSP	380	-1.0000E+00	5	
									6.	RSP	382	-1.0000E+00	6	
									7.	BCD	384	ABCDEFGH	7- 8	
									8.	RDP	387	-1.0000D+00	9-10	
									9.	CSP	390	(-1.0000E+00,-1.0000E+00)	11-12	
									10.	CDP	393	(-1.0000D+00,-1.0000D+00)	13-16	
38	33	399	DUMMOD2	1	8	8	10	26	1.	INT	405	-1	1	
									2.	INT	407	-1	2	
									3.	INT	409	-1	3	
									4.	INT	411	-1	4	
									5.	PSP	413	-1.0000E+00	5	
									6.	RSP	415	-1.0000E+00	6	
									7.	BCD	417	ABCDEFGH	7- 8	
									8.	RDP	420	-1.0000D+00	9-10	
									9.	CSP	423	(-1.0000E+00,-1.0000E+00)	11-12	
									10.	CDP	426	(-1.0000D+00,-1.0000D+00)	13-16	

DATA BLOCK AND TABLE DESCRIPTIONS

MODULE PROPERTIES LIST									PARAMETERS					
MPLID	NWDS	WD1	MOD-NAME	TYP	IN	OUT	SCR	INT	ID	TYP	P	DEFAULT (IF ANY)	W1-W2	FLG
39	33	431	DUMMOD3	1	8	8	10	26						
									1.	INT	438	-1	1	
									2.	INT	440	-1	2	
									3.	INT	442	-1	3	
									4.	INT	444	-1	4	
									5.	RSP	446	-1.0000E+00	5	
									6.	RSP	448	-1.0000E+00	6	
									7.	RCD	450	ABCDEFGH	7- 8	
									8.	RDP	453	-1.0000D+00	9-10	
									9.	CSP	456	(-1.0000E+00,-1.0000E+00)	11-12	
									10.	CDP	459	(-1.0000D+00,-1.0000D+00)	13-16	
40	33	464	DUMMOD4	1	8	8	10	26						
									1.	INT	471	-1	1	
									2.	INT	473	-1	2	
									3.	INT	475	-1	3	
									4.	INT	477	-1	4	
									5.	RSP	479	-1.0000E+00	5	
									6.	RSP	481	-1.0000E+00	6	
									7.	RCD	483	ABCDEFGH	7- 8	
									8.	RDP	486	-1.0000D+00	9-10	
									9.	CSP	489	(-1.0000E+00,-1.0000E+00)	11-12	
									10.	CDP	492	(-1.0000D+00,-1.0000D+00)	13-16	
41	20	497	(NONE)											
42	11	517	FMA1	1	5	2	3	10						
									1.	INT	524	-1	1	
									2.	RSP	526	1.0000E+00	2	

2.4-32 (12/29/78)

MODULE PROPERTIES LIST									PARAMETERS										W1-W2 FLG		
MPLID	NWDS	WD1	MOD-NAME	TYP	IN	OUT	SCR	TOT	ID	TYP	P	DEFAULT (IF ANY)									
43	41	528	EMG	1	6	6	3	15	1. INT	535		-1		1							
									2. INT	537		-1		2							
									3. INT	539		-1		3							
									4. INT	541		-1		4							
									5. INT	543		-1		5							
									6. INT	545		-1		6							
									7. INT	547		-1		7							
									8. INT	549		-1		8							
									9. INT	551		-1		9							
									10. INT	553		-1		10							
									11. INT	555		-1		11							
									12. INT	557		-1		12							
									13. INT	559		-1		13							
									14. INT	561		-1		14							
									15. INT	563		-1		15							
									16. INT	565		-1		16							
									17. RSP	567		1.0000E-02		17							
44	11	569	FA1	1	6	4	6	16	1. INT	576		-- NO DEFAULT --		1							
									2. INT	577		-- NO DEFAULT --		2							
									3. INT	578		0		3							
45	12	580	FA2	1	3	4	0	7	1. INT	587		-- NO DEFAULT --		1							
									2. RSP	588		-- NO DEFAULT --		2							
									3. RCD	589		YES		3- 4							
46	15	592	FRS	1	3	1	0	4	1. INT	599		0		1							
									2. INT	601		1		2							
									3. INT	603		0		3							
									4. INT	605		0		4							

DATA BLOCK AND TABLE DESCRIPTIONS

MODULE PROPERTIES LIST									PARAMETERS			W1-W2 FLG		
MPLID	NWDS	WD1	MOD-NAME	TYP	IN	OUT	SCR	TOT	ID	TYP	P	DEFAULT (IF ANY)		
47	13	607	FPLG	1	8	5	4	17	1.	BCD	614	-- NO DEFAULT --	1- 2	
									2.	INT	615	-1	3	
									3.	BCD	617	CPFG	4- 5	
48	17	620	FRRD	1	11	4	8	23	1.	BCD	627	-- NO DEFAULT --	1- 2	
									2.	BCD	628	-- NO DEFAULT --	3- 4	
									3.	INT	629	-- NO DEFAULT --	5	
									4.	INT	630	-- NO DEFAULT --	6	
									5.	INT	631	-- NO DEFAULT --	7	
									6.	INT	632	-- NO DEFAULT --	8	
									7.	INT	633	-- NO DEFAULT --	9	
									8.	INT	634	-- NO DEFAULT --	10	
									9.	INT	635	1	11	
49	18	637	(NONE)											
50	9	655	GI	1	8	1	6	15	1.	INT	662	-- NO DEFAULT --	1	
									2.	INT	663	-- NO DEFAULT --	2	

MODULE PROPERTIES LIST

									PARAMETERS			W1-W2 FLG	
MPLID	NWDS	WD1	MOD-NAME	TYP	IN	OUT	SCR	TOT	ID TYP	P	DEFAULT (IF ANY)		
51	24	664	GKAD	1	10	8	6	24	1. RCD	671	-- NO DEFAULT --	1- 2	
									2. RCD	672	-- NO DEFAULT --	3- 4	
									3. RCD	673	-- NO DEFAULT --	5- 6	
									4. RSP	674	-- NO DEFAULT --	7	
									5. RSP	675	-- NO DEFAULT --	8	
									6. RSP	676	-- NO DEFAULT --	9	
									7. INT	677	-- NO DEFAULT --	10	
									8. INT	678	-- NO DEFAULT --	11	
									9. INT	679	-- NO DEFAULT --	12	
									10. INT	680	-- NO DEFAULT --	13	
									11. INT	681	-- NO DEFAULT --	14	
									12. INT	682	-- NO DEFAULT --	15	
									13. INT	683	-- NO DEFAULT --	16	
									14. INT	684	-- NO DEFAULT --	17	
									15. INT	685	-- NO DEFAULT --	18	
									16. INT	686	-- NO DEFAULT --	19	
									17. INT	687	-- NO DEFAULT --	20	
52	19	689	GKAM	1	9	4	4	17	1. INT	695	-- NO DEFAULT --	1	
									2. INT	696	-- NO DEFAULT --	2	
									3. RSP	697	-- NO DEFAULT --	3	
									4. RSP	698	-- NO DEFAULT --	4	
									5. INT	699	-- NO DEFAULT --	5	
									6. INT	700	-- NO DEFAULT --	6	
									7. INT	701	-- NO DEFAULT --	7	
									8. INT	702	-- NO DEFAULT --	8	
									9. INT	703	1	9	
									10. INT	705	-1	10	
53	11	707	GP1	1	3	6	2	11	1. INT	714	-- NO DEFAULT --	1	
									2. INT	715	-- NO DEFAULT --	2	
									3. INT	716	1	3	
54	7	719	GP2	1	2	1	4	7	----- NO PARAMETERS EXIST -----				

DATA BLOCK AND TABLE DESCRIPTIONS

2.4-34 (12/29/78)

4

2.4-35 (12/29/78)

MODULE PROPERTIES LIST									PARAMETERS					
MPLID	NWDS	WD1	MOD-NAME	TYP	IN	OUT	SCR	TOT	ID	TYP	P	DEFAULT (IF ANY)	W1-W2	FLG
55	12	725	GP3	1	3	2	2	7	1.	INT	732	-- NO DEFAULT --	1	
									2.	INT	733	1	2	
									3.	INT	735	1	3	
56	22	737	GP4	1	6	4	2	12	1.	INT	744	-- NO DEFAULT --	1	
									2.	INT	745	-- NO DEFAULT --	2	
									3.	INT	746	-- NO DEFAULT --	3	
									4.	INT	747	-- NO DEFAULT --	4	
									5.	INT	748	-- NO DEFAULT --	5	
									6.	INT	749	-- NO DEFAULT --	6	
									7.	INT	750	-- NO DEFAULT --	7	
									8.	INT	751	-- NO DEFAULT --	8	
									9.	INT	752	-- NO DEFAULT --	9	
									10.	INT	753	1	10	
									11.	INT	755	-1	11	
									12.	INT	757	0	12	
57	10	759	GPCYC	1	3	1	2	6	1.	BCD	766	-- NO DEFAULT --	1- 2	
									2.	INT	767	1	3	
58	8	769	GPFDR	1	9	2	4	15	1.	BCD	776	-- NO DEFAULT --	1- 2	
59	9	777	GPSP	1	4	1	0	5	1.	INT	784	1	1	
60	11	786	GPWG	1	4	1	4	9	1.	INT	793	-1	1	
									2.	RSP	795	1.0000E+00	2	
61	20	797	(NONE)											

EXECUTIVE TABLE DESCRIPTIONS

2.4-36 (12/29/78)

MODULE PROPERTIES LIST									PARAMETERS					
MPLID	NWDS	WD1	MOD-NAME	TYP	IN	OUT	SCR	TOT	ID	TYP	P	DEFAULT (IF ANY)	W1-W2	FLG
62	13	817	INPUT	1	5	5	0	10	1.	INT	824	-1	1	
									2.	INT	826	0	2	
									3.	INT	828	0	3	
63	14	830	INPUTT1	1	0	5	0	5	1.	INT	837	0	1	
									2.	INT	839	0	2	
									3.	BCD	841	XXXXXXXX	3- 4	
64	14	844	INPUTT2	1	0	5	0	5	1.	INT	851	0	1	
									2.	INT	853	11	2	
									3.	BCD	855	XXXXXXXX	3- 4	
65	13	858	INPUTT3	1	5	5	0	10	1.	INT	865	-1	1	
									2.	INT	867	0	2	
									3.	INT	869	0	3	
66	13	871	INPUTT4	1	0	5	0	5	1.	INT	878	1	1	
									2.	INT	880	10	2	
									3.	INT	882	-1	3	
67	13	884	MATGEN	2	1	1	0	2	1.	INT	891	0	1	
									2.	INT	893	0	2	
									3.	INT	895	0	3	
68	11	897	MATGPR	2	4	0	0	4	1.	BCD	904	-- NO DEFAULT --	1- 2	
									2.	BCD	905	Y	3- 4	
69	7	908	MATPRN	2	5	0	0	5	----- NO PARAMETERS EXIST -----					

DATA BLOCK AND TABLE DESCRIPTIONS

2.4-37 (12/29/78)

MODULE PROPERTIES LIST									PARAMETERS			WI-W2 FLG	
MPLID	NWDS	WD1	MOD-NAME	TYP	IN	OUT	SCR	INT	10 TYP	P	DEFAULT (IF ANY)		
70	11	915	MATPRT	2	1	0	0	1	1. INT	922	0	1	
									2. INT	924	0	2	
71	7	926	MCE1	1	2	1	7	10	----- NO PARAMETERS EXIST -----				
72	7	933	MCF2	1	6	4	6	16	----- NO PARAMETERS EXIST -----				
73	17	940	MERGE	1	6	1	0	7	1. INT	947	-1	1	
									2. INT	949	0	2	
									3. INT	951	0	3	
									4. INT	953	0	4	
									5. INT	955	0	5	
74	20	957	(NONE)										
75	20	977	MODA	1	0	4	0	4	1. RSP	984	-- NO DEFAULT --	1	
									2. RSP	985	-- NO DEFAULT --	2	
									3. RSP	986	-- NO DEFAULT --	3	
									4. RSP	987	-- NO DEFAULT --	4	
									5. RSP	988	-- NO DEFAULT --	5	
									6. INT	989	-- NO DEFAULT --	6	
									7. INT	990	-- NO DEFAULT --	7	
									8. INT	991	-- NO DEFAULT --	8	
									9. INT	992	-- NO DEFAULT --	9	
									10. INT	993	-- NO DEFAULT --	10	
									11. RSP	994	-- NO DEFAULT --	11	
									12. INT	995	-- NO DEFAULT --	12	
									13. INT	996	-- NO DEFAULT --	13	
76	10	997	MODACC	1	6	5	0	11	1. BCD	1004	TRAN	1- 2	

EXECUTIVE TABLE DESCRIPTIONS

DATA BLOCK AND TABLE DESCRIPTIONS

M O D U L E P R O P E R T I E S									L I S T						
									P A R A M E T E R S						
MPLID	NWDS	WDI	MOD-NAME	TYP	IN	OUT	SCR	TOT	ID	TYP	P	DEFAULT	(IF ANY)	W1-W2	FLG
77	18	1007	MQDB	1	3	4	0	7							
									1.	RSP	1014	--	NO DEFAULT	--	1
									2.	RSP	1015	--	NO DEFAULT	--	2
									3.	RSP	1016	--	NO DEFAULT	--	3
									4.	RSP	1017	--	NO DEFAULT	--	4
									5.	INT	1018	--	NO DEFAULT	--	5
									6.	INT	1019	--	NO DEFAULT	--	6
									7.	INT	1020	--	NO DEFAULT	--	7
									8.	RSP	1021	--	NO DEFAULT	--	8
									9.	INT	1022	--	NO DEFAULT	--	9
									10.	INT	1023	--	NO DEFAULT	--	10
									11.	INT	1024	--	NO DEFAULT	--	11
78	8	1025	MQDC	1	2	0	0	2							
									1.	INT	1032	--	NO DEFAULT	--	1
79	14	1033	MPYAD	1	3	1	1	5							
									1.	INT	1040	--	NO DEFAULT	--	1
									2.	INT	1041		1		2
									3.	INT	1043		1		3
									4.	INT	1045		0		4
80	14	1047	MTRXIN	1	5	3	7	15							
									1.	INT	1054	--	NO DEFAULT	--	1
									2.	INT	1055		-1		2
									3.	INT	1057		-1		3
									4.	INT	1059		-1		4
81	11	1061	QFP	2	6	0	0	6							
									1.	INT	1068		0		1
									2.	INT	1070		-1		2
82	10	1072	OPTPR1	1	5	1	1	7							
									1.	INT	1070	--	NO DEFAULT	--	1
									2.	INT	1080	--	NO DEFAULT	--	2
									3.	INT	1081	--	NO DEFAULT	--	3

2.4-39 (12/29/78)

11-9

M O D U L E P R O P E R T I E S									L I S T					
MPLID	NWDS	WD1	MOD-NAME	TYP	IN	OUT	SCR	TOT	ID	TYP	P	PARAMETERS	W1-W2	FLG
DEFAULT (IF ANY)														
83	12	1082	OPTPR2	1	3	2	0	5						
									1. INT 1089			-- NO DEFAULT --	1	
									2. INT 1090			-- NO DEFAULT --	2	
									3. INT 1091			-- NO DEFAULT --	3	
									4. INT 1092			0	4	
94	20	1094	(NONE)											
95	9	1114	OUTPUT	2	1	0	0	1						
									1. INT 1121			-1	1	
96	14	1123	OUTPUT1	2	5	0	0	5						
									1. INT 1130			0	1	
									2. INT 1132			0	2	
									3. BCD 1134			XXXXXXXX	3- 4	
97	14	1137	OUTPUT2	2	5	0	0	5						
									1. INT 1144			0	1	
									2. INT 1146			11	2	
									3. BCD 1148			XXXXXXXX	3- 4	
98	22	1151	OUTPUT3	2	5	0	0	5						
									1. INT 1158			0	1	
									2. BCD 1160			-- NO DEFAULT --	2- 3	
									3. BCD 1161			XXX	4- 5	
									4. BCD 1164			XXX	6- 7	
									5. BCD 1167			XXX	8- 9	
									6. BCD 1170			XXX	10-11	
99	11	1173	OUTPUT4	2	5	0	0	5						
									1. INT 1180			-1	1	
									2. INT 1182			10	2	

MODULE PROPERTIES									LIST		
									PARAMETERS		
MPLID	NWDS	WD1	MOD-NAME	TYP	IN	OUT	SCR	TOT	ID	TYP	P
									DEFAULT (IF ANY)		W1-W2 FLG
90	14	1184	PARAM	1	0	0	0	0	1. BCD 1191	-- NO DEFAULT --	1- 2
									2. INT 1192	1	3
									3. INT 1194	1	4
									4. INT 1196	1	5
91	21	1198	PARAML	2	1	0	0	1	1. BCD 1205	-- NO DEFAULT --	1- 2
									2. INT 1206	1	3
									3. INT 1208	1	4
									4. RSP 1210	0.	5
									5. INT 1212	0	6
									6. RSP 1214	0.	7
									7. BCD 1216		8- 9
92	25	1219	PARAMR	2	0	0	0	0	1. BCD 1226	-- NO DEFAULT --	1- 2
									2. RSP 1227	0.	3
									3. RSP 1229	0.	4
									4. RSP 1231	0.	5
									5. CSP 1233 (0.	, 0.) 6- 7 ***
									6. CSP 1236 (0.	, 0.) 8- 9 ***
									7. CSP 1239 (0.	, 0.) 10-11 ***
									8. INT 1242	0	12
93	23	1244	PARTN	1	3	4	0	7	1. INT 1251	-1	1
									2. INT 1253	0	2
									3. INT 1255	0	3
									4. INT 1257	0	4
									5. INT 1259	0	5
									6. INT 1261	0	6
									7. INT 1263	0	7
									8. INT 1265	0	8
94	20	1267	(NONE)								

MODULE PROPERTIES LIST									
MPLID	NWDS	WD1	MOD-NAME	TYP	IN	OUT	SCR	INT	PARAMETERS
									DEFAULT (IF ANY)
95	17	1287	MRFD1	1	4	4	1	9	1. BCD 1294 -- NO DEFAULT -- 2. INT 1295 -- NO DEFAULT -- 3. INT 1296 -- NO DEFAULT -- 4. INT 1297 -- NO DEFAULT -- 5. INT 1298 -- NO DEFAULT -- 6. BCD 1299 -- NO DEFAULT -- 7. INT 1300 0 8. RSP 1302 0.
96	14	1304	MRFD2	1	12	6	10	28	1. INT 1311 -- NO DEFAULT -- 2. INT 1312 -- NO DEFAULT -- 3. BCD 1313 0 4. INT 1316
97	12	1318	CMRED2	1	11	6	11	28	1. INT 1325 -- NO DEFAULT -- 2. INT 1326 -- NO DEFAULT -- 3. BCD 1327
98	13	1330	PLA1	1	7	4	0	11	1. INT 1337 -- NO DEFAULT -- 2. INT 1338 -- NO DEFAULT -- 3. INT 1339 -- NO DEFAULT -- 4. INT 1340 -- NO DEFAULT -- 5. INT 1341 -- NO DEFAULT -- 6. CSP 1342 -- NO DEFAULT --
99	8	1343	PLA2	1	3	3	0	6	1. INT 1350 -- NO DEFAULT --
100	9	1351	PLA3	1	6	2	1	9	1. INT 1358 -- NO DEFAULT -- 2. INT 1359 -- NO DEFAULT --

DATA BLOCK AND TABLE DESCRIPTIONS

2.4-42 (12/29/78)

2.4-43 (12/29/78)

MODULE PROPERTIES								LIST					
MPLID	NWDS	WD1	MOD-NAME	TYP	IN	OUT	SCR	TOT	ID TYP	P	PARAMETERS	W1-W2	FLG
									DEFAULT (IF ANY)				
101	10	1360	PLA4	1	6	2	1	9	1. INT 1367	-- NO DEFAULT --	1		
									2. INT 1368	-- NO DEFAULT --	2		
									3. CSP 1369	-- NO DEFAULT --	3-4		
102	80	1370	(NONE)										
103	14	1450	PL0T	1	11	1	4	16	1. INT 1457	-- NO DEFAULT --	1		
									2. INT 1458	-- NO DEFAULT --	2		
									3. INT 1459	-- NO DEFAULT --	3		
									4. INT 1460	1	4		
									5. INT 1462	0	5		
104	10	1464	PLTSET	1	3	4	2	9	1. INT 1471	-- NO DEFAULT --	1		
									2. INT 1472	-1	2		
105	11	1474	PLTTRAN	1	2	2	0	4	1. INT 1481	0	1		
									2. INT 1483	0	2		
106	7	1485	PRTMSG	2	1	0	0	1	----- NO PARAMETERS EXIST -----				
107	13	1492	PRTPARM	2	0	0	0	0	1. INT 1499	-- NO DEFAULT --	1		
									2. BCD 1500	XXXXXXXXXX	2-3		
									3. INT 1503	0	4		
108	9	1505	RANDOM	1	9	2	0	11	1. INT 1512	-1	1		
109	7	1514	RBMG1	1	3	6	1	10	----- NO PARAMETERS EXIST -----				
110	11	1521	RBMG2	1	1	1	4	6	1. INT 1528	1	1		
									2. RSP 1530	1.0000E+00	2		

EXECUTIVE TABLE DESCRIPTIONS

2.4-44 (12/29/78)

MODULE PROPERTIES LIST									PARAMETERS			
MPLID	NWDS	WD1	MOD-NAME	TYP	IN	OUT	SCR	TOT	ID TYP	P	DEFAULT (IF ANY)	W1-W2 FLG
111	7	1532	RBMG3	1	3	1	2	6		----	NO PARAMETERS EXIST	----
112	7	1539	RBMG4	1	4	1	3	8		----	NO PARAMETERS EXIST	----
113	20	1546	(NONE)									
114	11	1546	READ	1	7	4	9	20				
									1. RCD 1573	-- NO DEFAULT --		1- 2
									2. INT 1574	-- NO DEFAULT --		3
									3. INT 1575		1	4
115	14	1577	RMG	1	4	3	6	13				
									1. RSP 1584	0.		1
									2. RSP 1586	0.		2
									3. INT 1588	-1		3
									4. INT 1590	-- NO DEFAULT --		4
116	14	1591	SCALAR	2	1	0	0	1				
									1. INT 1598		1	1
									2. INT 1600		1	2
									3. CSP 1602 (0.	0.	1	3- 4
117	7	1605	SCE1	1	5	6	1	12		----	NO PARAMETERS EXIST	----
118	9	1612	SDR1	1	11	3	6	20				
									1. INT 1619	-- NO DEFAULT --		1
									2. RCD 1620	-- NO DEFAULT --		2- 3
119	12	1621	SDR2	1	15	6	3	24				
									1. RCD 1628	-- NO DEFAULT --		1- 2
									2. INT 1629		1	3
									3. INT 1631		-1	4
120	7	1633	SDR3	1	6	6	8	20		----	NO PARAMETERS EXIST	----

DATA BLOCK AND TABLE DESCRIPTIONS

2.4-45 (12/29/78)

MODULE PROPERTIES LIST									PARAMETERS					
MPLID	NWOS	WD1	MOD-NAME	TYP	IN	OUT	SCR	TOT	ID	TYP	P	DEFAULT (IF ANY)	W1-W2	FLG
121	11	1640	SDRHT	1	10	1	3	14	1.	RSP	1647	0.	1	
									2.	INT	1649	-1	2	
122	31	1651	SEEMAT	2	5	0	0	5	1.	PCD	1658	PRINT	1- 2	
									2.	INT	1661	0	3	
									3.	INT	1663	100	4	
									4.	BCD	1665	SC	5- 6	
									5.	INT	1668	0	7	
									6.	BCD	1670		8- 9	
									7.	INT	1673	0	10	
									8.	BCD	1675		11-12	
									9.	RSP	1678	3.0000E+01	13	
									10.	RSP	1680	3.0000E+01	14	
123	18	1682	(NONE)											
124	26	1700	SETVAL	2	0	0	0	0	1.	INT	1707	-- NO DEFAULT --	1	
									2.	INT	1708	-1	2	
									3.	INT	1710	-1	3	
									4.	INT	1712	-1	4	
									5.	INT	1714	-1	5	
									6.	INT	1716	-1	6	
									7.	INT	1718	-1	7	
									8.	INT	1720	-1	8	
									9.	INT	1722	-1	9	
									10.	INT	1724	-1	10	
125	11	1726	SMA1	1	5	3	2	10	1.	INT	1733	-- NO DEFAULT --	1	
									2.	INT	1734	-- NO DEFAULT --	2	
									3.	INT	1735	-1	3	

EXECUTIVE TABLE DESCRIPTIONS

MODULE PROPERTIES LIST									PARAMETERS			W1-W2 FLG	
MPLID	NWDS	WD1	MOD-NAME	TYP	IN	OUT	SCR	TOT	ID	TYP	P	DEFAULT (IF ANY)	
126	32	1737	SMA2	1	5	2	2	9	1.	RSP	1744	-- NO DEFAULT --	1
									2.	INT	1745	-- NO DEFAULT --	2
									3.	INT	1746	-- NO DEFAULT --	3
									4.	INT	1747	-1	4
									5.	INT	1749	-1	5
									6.	INT	1751	-1	6
									7.	INT	1753	-1	7
									8.	INT	1755	-1	8
									9.	INT	1757	-1	9
									10.	INT	1759	-1	10
									11.	INT	1761	-1	11
									12.	INT	1763	-1	12
									13.	INT	1765	-1	13
									14.	INT	1767	-1	14
127	10	1769	SMA3	1	2	1	7	10	1.	INT	1776	-- NO DEFAULT --	1
									2.	INT	1777	-- NO DEFAULT --	2
									3.	INT	1778	-- NO DEFAULT --	3
128	7	1779	SMP1	1	5	9	7	21	----- NO PARAMETERS EXIST -----				
129	7	1786	SMP2	1	3	1	6	10	----- NO PARAMETERS EXIST -----				
130	22	1793	SMPYAD	1	6	1	2	9	1.	INT	1800	-- NO DEFAULT --	1
									2.	INT	1801	1	2
									3.	INT	1803	1	3
									4.	INT	1805	0	4
									5.	INT	1807	0	5
									6.	INT	1809	0	6
									7.	INT	1811	0	7
									8.	INT	1813	0	8

M O D U L E P R O P E R T I E S									L I S T					
MPLID	NWDS	WD1	MOD-NAME	TYP	IN	OUT	SCR	INT	P A R A M E T E R S					
									ID	TYP	P	DEFAULT (IF ANY)	W1-W2	FLG
131	15	1815	SOLVE	1	2	1	5	8	1. INT 1822			0		1
									2. INT 1824			1		2
									3. INT 1826			0		3
									4. INT 1828			0		4
132	20	1830	(NONE)											
133	11	1850	SSG1	1	11	1	3	15	1. INT 1857		-- NO DEFAULT --			1
									2. INT 1858		-- NO DEFAULT --			2
									3. INT 1859		-1			3
134	7	1861	SSG2	1	7	4	4	15	----- NO P A R A M E T E R S E X I S T					
135	13	1868	SSG3	1	6	4	2	12	1. INT 1875		-- NO DEFAULT --			1
									2. INT 1876		-- NO DEFAULT --			2
									3. INT 1877		1			3
									4. INT 1879		1			4
136	8	1881	SSG4	1	11	2	5	18	1. INT 1888		-- NO DEFAULT --			1
137	23	1889	SSGHT	1	17	3	5	25	1. INT 1896		-1			1
									2. INT 1898		-1			2
									3. RSP 1900		1.0000E-03			3
									4. RSP 1902		0.			4
									5. INT 1904		4			5
									6. INT 1906		-1			6
									7. INT 1908		0			7
									8. INT 1910		0			8

EXECUTIVE TABLE DESCRIPTIONS

2.4-47a (12/29/78)

MODULE PROPERTIES LIST									PARAMETERS					
MPLID	NWDS	WD1	MOD-NAME	TYP	IN	OUT	SCR	TOT	ID	TYP	P	DEFAULT (IF ANY)	W1-W2	FLG
138	13	1912	TA1	1	6	5	4	15	1.	INT	1919	-- NO DEFAULT --	1	
									2.	INT	1920	-- NO DEFAULT --	2	
									3.	INT	1921	1	3	
									4.	INT	1923	-- NO DEFAULT --	4	
									5.	INT	1924	-- NO DEFAULT --	5	
139	22	1925	TABPCH	2	5	0	0	5	1.	BCD	1932	AA	1- 2	
									2.	BCD	1935	AB	3- 4	
									3.	BCD	1938	AC	5- 6	
									4.	BCD	1941	AD	7- 8	
									5.	BCD	1944	AE	9-10	
140	13	1947	(NONE)											
141	12	1960	TARPRT	2	1	0	0	1	1.	BCD	1967	-- NO DEFAULT --	1- 2	
									2.	INT	1968	0	3	
									3.	INT	1970	0	4	
142	7	1972	TABPT	2	5	0	0	5	----- NO PARAMETERS EXIST -----					
143	20	1979	(NONE)											
144	17	1999	TIMTFEST	1	0	2	2	4	1.	INT	2006	50	1	
									2.	INT	2008	200	2	
									3.	INT	2010	2	3	
									4.	INT	2012	1	4	
									5.	INT	2014	0	5	

DATA BLOCK AND TABLE DESCRIPTIONS

2.4-47b (12/29/78)

M O D U L E P R O P E R T I E S L I S T									P A R A M E T E R S							
MPLID	NWDS	WD1	MOD-NAME	TYP	IN	OUT	SCR	TOT	ID	TYP	P	DEFAULT	(IF ANY)	W1-W2	FLG	
145	13	2016	TRD	1	8	2	8	18	1.	BCD	2023	--	NO DEFAULT	--	1- 2	
									2.	INT	2024	--	NO DEFAULT	--	3	
									3.	INT	2025	--	NO DEFAULT	--	4	
									4.	INT	2026	--	NO DEFAULT	--	5	
									5.	INT	2027		-1		6	
146	15	2029	TRHT	1	10	2	7	19	1.	RSP	2036		5.5000E-01		1	
									2.	RSP	2038		0.		2	
									3.	INT	2040		-1		3	
									4.	INT	2042		-1		4	
147	13	2044	TRLG	1	14	6	9	29	1.	INT	2051		-1		1	
									2.	INT	2053		-1		2	
									3.	INT	2055		0		3	
148	7	2057	TRNSP	1	1	1	8	10	----- NO P A R A M E T E R S E X I S T -----							
149	10	2064	UMERGE	1	3	1	1	5	1.	BCD	2071	--	NO DEFAULT	--	1- 2	
									2.	BCD	2072	--	NO DEFAULT	--	3- 4	
									3.	BCD	2073	--	NO DEFAULT	--	5- 6	
150	10	2074	UPARTN	1	2	4	1	7	1.	BCD	2081	--	NO DEFAULT	--	1- 2	
									2.	BCD	2082	--	NO DEFAULT	--	3- 4	
									3.	BCD	2083	--	NO DEFAULT	--	5- 6	
151	14	2084	VDR	1	7	2	2	11	1.	BCD	2091	--	NO DEFAULT	--	1- 2	
									2.	BCD	2092	--	NO DEFAULT	--	3- 4	
									3.	INT	2093	--	NO DEFAULT	--	5	
									4.	INT	2094		0		6	
									5.	INT	2096	--	NO DEFAULT	--	7	
									6.	INT	2097	--	NO DEFAULT	--	8	

EXECUTIVE TABLE DESCRIPTIONS

2.4-47c (12/29/78)

MODULE PROPERTIES LIST									PARAMETERS				
MPLID	NWDS	W01	MOD-NAME	TYP	IN	OUT	SCR	TOT	ID	TYP	P	DEFAULT (IF ANY)	W1-W2 FLG
152	16	2098	VEC	1	1	1	0	2	1.	RCD	2105	-- NO DEFAULT --	1- 2
									2.	RCD	2106	COMP	3- 4
									3.	RCD	2109	COMP	5- 6
									4.	INT	2112	0	7
153	20	2114	(NONE)										
154	7	2134	XYPLOT	2	1	0	2	3	----- NO PARAMETERS EXIST -----				
155	7	2141	XYPRNPLT	2	1	0	0	1	----- NO PARAMETERS EXIST -----				
156	19	2148	XYTPAN	1	6	1	5	12	1.	RCD	2155	TRANS	1- 2
									2.	RCD	2158	SOL	3- 4
									3.	INT	2161	0	5
									4.	INT	2163	0	6
									5.	INT	2165	1	7
157	20	2167	(NONE)										
158	13	2187	COMB1	1	2	1	10	13	1.	INT	2194	0	1
									2.	INT	2196	-- NO DEFAULT --	2
									3.	RCD	2197		3- 4

DATA BLOCK AND TABLE DESCRIPTIONS

MODULE PROPERTIES LIST									PARAMETERS		W1-W2 FIG		
MPLID	NWDS	WD1	MOD-NAME	TYP	IN	OUT	SCR	TOT	ID TYP	P	DEFAULT (IF ANY)		
159	35	2200	COMB2	1	7	1	7	15	1. INT 2207		-- NO DEFAULT --		1
									2. BCD 2208		-- NO DEFAULT --		2- 3
									3. BCD 2209				4- 5
									4. BCD 2212				6- 7
									5. BCD 2215				8- 9
									6. BCD 2218				10-11
									7. BCD 2221				12-13
									8. BCD 2224				14-15
									9. BCD 2227				16-17
									10. BCD 2230				18-19
									11. INT 2233		0		20
160	36	2235	EXIO	2	0	0	2	2	1. INT 2242		-- NO DEFAULT --		1
									2. INT 2243		-- NO DEFAULT --		2
									3. BCD 2244		-- NO DEFAULT --		3- 4
									4. BCD 2245		-- NO DEFAULT --		5- 6
									5. BCD 2246		-- NO DEFAULT --		7- 8
									6. BCD 2247		-- NO DEFAULT --		9-10
									7. BCD 2248		-- NO DEFAULT --		11-12
									8. BCD 2249		ALL		13-14
									9. BCD 2252		WHOLESOFT		15-16
									10. BCD 2255		XXXXXXXXXX		17-18
									11. BCD 2258		XXXXXXXXXX		19-20
									12. BCD 2261		XXXXXXXXXX		21-22
									13. BCD 2264		XXXXXXXXXX		23-24
									14. INT 2267		0		25
									15. INT 2269		0		26

EXECUTIVE TABLE DESCRIPTIONS

MODULE PROPERTIES LIST									PARAMETERS				W1-W2 FLG	
MPLID	NWDS	WD1	MOD-NAME	TYP	IN	OUT	SCR	TGT	ID	TYP	P	DEFAULT (IF ANY)		
161	39	2271	RCOVR	1	11	8	9	28	1.	INT	2278	-- NO DEFAULT --		1
									2.	INT	2279	-- NO DEFAULT --		2
									3.	INT	2280	-- NO DEFAULT --		3
									4.	BCD	2281	-- NO DEFAULT --		4- 5
									5.	INT	2282	-- NO DEFAULT --		6
									6.	INT	2283	0		7
									7.	INT	2285	0		8
									8.	RCD	2287			9-10
									9.	RCD	2290			11-12
									10.	BCD	2293			13-14
									11.	BCD	2296			15-16
									12.	BCD	2299			17-18
									13.	INT	2302	-1		19
									14.	RSP	2304	0.		20
									15.	RSP	2306	0.		21
									16.	RSP	2308	0.		22
162	11	2310	(NONE)											
163	11	2321	RCOVR3	1	4	7	3	14	1.	INT	2328	-- NO DEFAULT --		1
									2.	RCD	2329	-- NO DEFAULT --		2- 3
									3.	INT	2330	-1		4
164	16	2332	REDUCE	1	2	3	2	7	1.	INT	2339	0		1
									2.	INT	2341	0		2
									3.	RCD	2343			3- 4
									4.	INT	2346	0		5
165	11	2348	SGEN	1	4	10	0	14	1.	INT	2355	-- NO DEFAULT --		1
									2.	RCD	2356	-- NO DEFAULT --		2- 3
									3.	INT	2357	-- NO DEFAULT --		4
									4.	INT	2358	-- NO DEFAULT --		5

2.4-47f (12/29/78)

MODULE PROPERTIES LIST									PARAMETERS					
MPLID	NWDS	WD1	MOD-NAME	TYP	IN	OUT	SCR	TOT	ID	TYP	P	DEFAULT (IF ANY)	W1-W2	FLG
166	25	2359	SOFI	1	0	5	0	5	1.	INT	2366	-1	1	
									2.	BCD	2368	-- NO DEFAULT --	2- 3	
									3.	BCD	2369		4- 5	
									4.	BCD	2372		6- 7	
									5.	BCD	2375		8- 9	
									6.	BCD	2378		10-11	
									7.	BCD	2381		12-13	
167	25	2384	SOFD	2	5	0	0	5	1.	INT	2391	-1	1	
									2.	BCD	2393	-- NO DEFAULT --	2- 3	
									3.	BCD	2394		4- 5	
									4.	BCD	2397		6- 7	
									5.	BCD	2400		8- 9	
									6.	BCD	2403		10-11	
									7.	BCD	2406		12-13	
168	37	2409	SOFUT	2	0	0	1	1	1.	INT	2416	-1	1	
									2.	BCD	2418	-- NO DEFAULT --	2- 3	
									3.	BCD	2419	-- NO DEFAULT --	4- 5	
									4.	INT	2420	0	6	
									5.	BCD	2422		7- 8	
									6.	BCD	2425		9-10	
									7.	BCD	2428		11-12	
									8.	BCD	2431		13-14	
									9.	BCD	2434		15-16	
									10.	BCD	2437		17-18	
									11.	BCD	2440		19-20	
									12.	BCD	2443		21-22	
169	15	2446	SURPH1	2	7	0	1	8	1.	INT	2453	0	1	
									2.	BCD	2455	-- NO DEFAULT --	2- 3	
									3.	INT	2456	0	4	
									4.	BCD	2458		5- 6	

EXECUTIVE TABLE DESCRIPTIONS

2.4-47g (12/29/78)

MODULE PROPERTIES LIST									PARAMETERS					
MPLID	NWDS	WD1	MOD-NAME	TYP	IN	OUT	SCR	TOT	ID	TYP	P	DEFAULT (IF ANY)	W1-W2	FLG
170	11	2461	PLTMRG	1	2	6	1	9	1.	BCD	2468	-- NO DEFAULT --	1- 2	
									2.	INT	2469	-- NO DEFAULT --	3	
									3.	INT	2470	-- NO DEFAULT --	4	
									4.	INT	2471	-- NO DEFAULT --	5	
171	18	2472	(NONE)											
172	9	2490	COPY	1	1	1	0	2	1.	INT	2497	-1	1	
173	9	2499	SWITCH	2	2	0	0	2	1.	INT	2506	-1	1	
174	11	2508	MPY3	1	3	1	3	7	1.	INT	2515	0	1	
									2.	INT	2517	0	2	
175	33	2519	SDCMPS	1	4	2	6	12	1.	INT	2526	0	1	
									2.	INT	2528	0	2	
									3.	INT	2530	20	3	
									4.	INT	2532	0	4	
									5.	INT	2534	0	5	
									6.	BCD	2536	L	6- 7	
									7.	INT	2539	0	8	
									8.	CSP	2541	(0. , 0.)	9-10	
									9.	RDP	2544	0.	11-12	
									10.	INT	2547	0	13	
									11.	BCD	2549	NONE	14-15	
176	9	2552	LODAPP	2	2	0	8	10	1.	BCD	2559	-- NO DEFAULT --	1- 2	
									2.	INT	2560	-- NO DEFAULT --	3	

DATA BLOCK AND TABLE DESCRIPTIONS

2.4-47h (12/29/78)

M O D U L E P R O P E R T I E S									L I S T					
									P A R A M E T E R S					
MPLID	NWDS	WDL	MOD-NAME	TYP	IN	OUT	SCR	TUT	ID	TYP	P	DEFAULT (IF ANY)	W1-W2	FLG
177	17	2561	GPSPC	1	4	2	0	6	1.	INT	2568	1	1	
									2.	INT	2570	-1	2	
									3.	INT	2572	-- NO DEFAULT --	3	
									4.	INT	2573	-- NO DEFAULT --	4	
									5.	INT	2574	-- NO DEFAULT --	5	
									6.	INT	2575	-- NO DEFAULT --	6	
									7.	INT	2576	-- NO DEFAULT --	7	
									8.	INT	2577	-- NO DEFAULT --	8	
178	15	2578	FOMCK	1	12	1	7	20	1.	INT	2585	0	1	
									2.	INT	2587	-1	2	
									3.	INT	2589	-- NO DEFAULT --	3	
									4.	RCD	2590	NONE	4- 5	
179	11	2593	ADR	1	7	1	5	13	1.	PSP	2600	-- NO DEFAULT --	1	
									2.	PSP	2601	0.	2	
									3.	RCD	2603	-- NO DEFAULT --	3- 4	
180	12	2604	FRRD2	1	6	1	9	16	1.	RSP	2611	-- NO DEFAULT --	1	
									2.	RSP	2612	0.	2	
									3.	RSP	2614	0.	3	
181	14	2616	GUST	1	10	1	7	18	1.	INT	2623	-- NO DEFAULT --	1	
									2.	RSP	2624	0.	2	
									3.	RSP	2626	0.	3	
									4.	RSP	2628	0.	4	
182	9	2630	IFT	1	4	2	0	6	1.	INT	2637	1	1	
193	9	2639	LAMX	1	2	1	0	3	1.	INT	2646	0	1	

EXECUTIVE TABLE DESCRIPTIONS

MODULE PROPERTIES LIST									PARAMETERS			W1-W2 FLG			
MPLID	NWDS	WD1	MOD-NAME	TYP	IN	OUT	SCR	TOT	ID	TYP	P	DEFAULT (IF ANY)			
144	15	2649	MTRXTEST	1	1	2	10	13	1.	INT	2655	31		1	
									2.	INT	2657	0		2	
									3.	INT	2659	-99025015		3	
									4.	INT	2661	-5099099		4	
185	11	2663	EMA	1	3	2	3	8	1.	INT	2670	-1		1	
									2.	RSP	2672	1.0000E+00		2	
136	9	2674	(NONE)												

*** END OF MPL PRINTOUT

*** THE MPL CONTAINS 186 ENTRIES. OF THESE, 20 ARE PAD ENTRIES.

EXECUTIVE TABLE DESCRIPTIONS

THIS PAGE HAS BEEN LEFT BLANK INTENTIONALLY.

DATA BLOCK AND TABLE DESCRIPTIONS

2.4.2.3 XPTDIC (Problem Tape Dictionary)

Description

XPTDIC is the Problem Tape Dictionary of data blocks checkpointed plus other information needed to restart a problem.

Created in Modules

XGPI, CHKPNT and XCEI.

Table Format

Record	Word	
0	1	ID
	2	
1	1	PR 31 17 16 NAF 1
	2	S
2	1	XVPS
	2	
	3	R 29 17 16 F 1
	4	BCD BLANKS
	5	BCD BLANKS
	6	DIN 31 17 16 ØRN 1
	7	DBN
	8	
	9	EQ ET ER S 31 30 29 R 17 16 F 1
	:	:
K	K	XVPS
	K+1	
	K+2	EQ ET ER S 31 30 29 R 17 16 F 1
3	:	:
	N	
		End-of-file

First entry in a group is a special entry

Repeat this entry for all data blocks referenced explicitly or implicitly in CHKPNT instruction

This group of entries is repeated for each CHKPNT module executed

EXECUTIVE TABLE DESCRIPTIONS

<u>Record</u>	<u>Word</u>	<u>Item</u>	<u>Description</u>
0	1,2	ID	Header record containing name XPTDIC (BCD).
1	1	PR	Present reel number of Problem Tape. Reels are numbered sequentially beginning with Reel 1.
		NAF	Next available file number on present reel. Files are numbered sequentially beginning with file 1.
	2	S	Sequence number of last restart dictionary card punched out.
2	1,2	XVPS	BCD name XVPS. The file corresponding to this entry contains named common blocks /XVPS/ and /XCEITB/.
	3	R,F	Reel number and file number where the file corresponding to this entry is located. For this entry the reel number must be one.
	4,5	(blanks)	BCD blanks indicate special entry.
	6	DIN	DMAP instruction number of DMAP instruction following CHKPNT module (i.e., re-entry point).
		ØRN	ØSCAR record number of CHKPNT module being executed.
	7,8	DBN	Data block name (BCD) of data block being checkpointed.
	9	EQ	Equivalence flag. EQ = 1 indicates data block is equivalenced to another data block.
		ET	End of tape flag. ET = 1 indicates that data block is split across two reels of problem tape.
		ER	End of logical record flag. ER = 1 indicates that the complete logical record was written out prior to changing reels when ET = 1.
		R,F	Reel number and file number where the file corresponding to this entry is located. For purged or not-generated data blocks, R = 0 and F = 0.
	K, K+1	XVPS	BCD name XVPS. The file corresponding to this entry contains named common blocks /XVPS/, /XCEITB/ and /SYSTEM/.
	K+2	EQ,ET, ER,R,F	See word 9 for descriptions.

Notes

1. All entries are integer unless otherwise noted.
2. The XPTDIC table is always the last file on the Problem Tape.
3. XGPI generates records 0, 1 and the first entry of record 2. CHKPNT modules add entries to record 3. XCEI drops entries from record 2 when a REPT DMAP instruction transfers control to the top of a DMAP loop.
4. XCSA also creates a XPTDIC table when problem is being restarted. This special XPTDIC table is created from the restart dictionary and its format is essentially the same as described above except that there are no special entries.

DATA BLOCK AND TABLE DESCRIPTIONS

2.4.2.4 PVT (Parameter Value Table)

Description

The Parameter Value Table contains the parameter names and values of all parameters input by means of the PARAM bulk data card.

Created in Module

IFP.

Table Format

<u>Record</u>	<u>Word</u>	<u>Item</u>
0	1,2	Header record contains name PVT (BCD).
1	1,2	Name of parameter (BCD)
	3	Type code for parameter value
	4	Value of parameter. Type codes
	⋮	and corresponding lengths, L, of
	3+L	values are given in table below.
	⋮	

} repeat
for all
parameters
on PARAM
cards.

Notes

Type Code	Meaning of Code	Corresponding Length in Words
1	Integer	1
2	Real, single precision	1
3	BCD	2
4	Real, double precision	2
5	Complex, single precision	2
6	Complex, double precision	4

1. IFP does not create PVT if no PARAM cards exist in the Bulk Data Deck.
2. PVT is written on a scratch file as 2 or more records (a header record and 1 record for each PARAM card).
3. The PVT table is located in named common block /XPVT/.

EXECUTIVE TABLE DESCRIPTIONS

2.4.2.5 XCSA (Executive Control Table)

Description

Executive control table derived from the Executive Control Deck.

Created in Module

XCSA.

Table Format

<u>Record</u>	<u>Word</u>	<u>Item</u>
0	1	BCD word XCSA - header ID.
	2,3	} Dictionary of contents of records to follow. Does not need to be in this order, nor is MED always present.
	4,5	
	6,7	
1	1	Approach code
	2	Start code
	3,4	Alter parameters
	5	Solution number
	6	Subset number
2	1	RD table (packed DMAP program)
	:	or user generated DMAP
	N	program (18 words per card image).
3	1	Number of DMAP instructions
	2	Number of words per IS1 table entry.
	3	} MED record included only if approach calls for a Rigid Format
	:	
	:	
	L	Number of entries in JNM table.
	L+1	JNM table (File Name Table)
	:	}
	M	Number of entries in INM table
	M+1	INM table (Card Name Table)
4	:	
		End-of-file

Notes

1. Data block XCSA is written on the Problem Tape.
2. A more detailed description of tables IS1, JNM and INM is given in the Module Functional Description for module XCSA, section 4.2.

DATA BLOCK AND TABLE DESCRIPTIONS

2.4.2.6 XALTER (Executive Alter Table)

Description

XALTER is generated from the ALTER data in the Executive Control Deck.

Created in Module

XCSA.

Table Format

<u>Record</u>	<u>Word</u>	<u>Item</u>	
0	1,2	BCD word XALTER - header record	
1	1,2	Numbers of DMAP instructions to be altered. (Integers).	
2	1	Card image (BCD)	{ Zero or more of these records. Repeat until next 2 word record encountered. } Repeat 1 or more times until EOF encountered.
	:		
	18		
:			
N		End-of-file	

Notes

XALTER data block is written on the Problem Tape.

EXECUTIVE TABLE DESCRIPTIONS

2.4.2.7 LNKSPC (Link Specification Table - Resident Base)

Description

This Link Specification Table (see also 2.4.1.9) contains an entry for each executable DMAP module available within the NASTRAN system. Each entry contains: a) the DMAP module name, b) the module's subroutine entry point name and c) a key indicating the links in which the module resides for each of four machine types.

Created in Module

LNKSPC data is stored by the XBSBD Block Data subprogram in module XGPI (4.7).

Table Format

Word	1	LLINK	
	2		
	3	DMAP Module	
		Name	
	4	Entry Point	
		Name	
	5		
	6	Key #1	Entry #1 (Sample)
	7	Key #2	
	8	Key #3	
	9	Key #4	
	:		

<u>Word</u>	<u>Item</u>	<u>Description</u>
1	LLINK	Length of table in words (excluding word 1)
2,3	DMAP Module Name	DMAP name-8 characters (4 characters/word)
4,5	Entry Point Name	Entry name-8 characters (4 characters/word)
6	Key #1	Link residence key for machine type #1
7	Key #2	Link residence key for machine type #2
8	Key #3	Link residence key for machine type #3
9	Key #4	Link residence key for machine type #4

DATA BLOCK AND TABLE DESCRIPTIONS

The machine type code number is the same as that defined in the MACH word of the SYSTEM (2.4.1.8) table. Each bit within the Key word specifies a link number that is to contain that module. Bits are numbered from right to left; the right most (least significant) bit specifies that the module is to reside in link 1, etc. For example, if a particular Key contained 26 (binary 011010), only links 2, 4 and 5 would contain the specified module.

Notes

1. The LNKSPC table must contain an entry for each executable module that is in the MPL (2.4.2.2) table.
2. The LNKSPC table is located in /XLKSPC/.

EXECUTIVE TABLE DESCRIPTIONS

2.4.2.8 IFPX0 (Modified Restart Table)

Description

IFPX0 records the types of changes to the input data which were made during a restart. In addition, it classifies each type of change as to substantive (solution affecting) or nonsubstantive (output only affecting). The basic data is stored in packed format, 31 bits to the word. The use of this array in restart and its companion /IFPX1/ is described in section 1.10.

Created in Module(s)

IFP1, XSORT and IFP and read by XGPI.

Table Format

<u>Word</u>	<u>Item</u>	
1	N	Number of pairs (I,L) to follow.
2	I1	Pointer to first word in /IFPX0/ that is used to flag modified bulk data.
3	L1	Number of words reserved for modified bulk data flags.
4	I2	Pointer to first word in /IFPX0/ that is used to flag modified Case Control data.
5	L2	Number of words reserved for modified Case Control data flags.
6	I3	Not used.
7	L3	
8 through 18	IB	Array containing flags which specify what input has been modified for restart (the meaning of each bit can be determined from /IFPX1/. (See section 2.4.2.9).
19 through 29		Array which specifies which bits in the IB array can initiate a modified restart.

DATA BLOCK AND TABLE DESCRIPTIONS

2.4.2.9 IFPX1 (Master Card Name Table)

Description

IFPX1 contains mnemonics for the various card types (and data types) which can be significant for restart. It is actually a key into common block IFPX0 (see section 2.4.2.8). The use of this array is described in section 1.10.

Created in Module

Modules IFP1, XSØRT, IFP and XGPI read this array.

Table Format

<u>Word No. In IFPX1</u>	<u>Bit No. In IFPX0</u>	<u>Contents</u>	<u>Output Only (PMR)</u>	<u>Supported</u>
1		310	Number of Card Types	
2	1	GRID		
4	2	GRDSET		
6	3	BEAMØR		No
8	4	SEQGP		
10	5	CØRD1R		
12	6	CØRD1C		
14	7	CØRD1S		
16	8	CØRD2R		
18	9	CØRD2C		
20	10	CØRD2S		
22	11	PLØTEL	Yes	
24	12	SPC1		
26	13	SPCADD		
28	14	SUPØRT		
30	15	ØMIT		
32	16	SPC		
34	17	MPC		
36	18	FØRCE		

6

EXECUTIVE TABLE DESCRIPTIONS

<u>Word No. In IFPX1</u>	<u>Bit No. In IFPX0</u>	<u>Contents</u>	<u>Output Only (PMR)</u>	<u>Supported</u>
38	19	MØMENT		
40	20	FØRCE1		
42	21	MØMENT1		
44	22	FØRCE2		
46	23	MØMENT2		
48	24	PLØAD		
50	25	SLØAD		
52	26	GRAV		
54	27	TEMP		
56	28	GENEL		
58	29	PRØD		
60	30	PTUBE		
62	31	PVISC		
64	32(word 2)	PBEAM		No.
66	33	PTRIA1		
68	34	PTRIA2		
70	35	PTRBSC		
72	36	PTRPLT		
74	37	PTRMEM		
76	38	PQUAD1		
78	39	PQUAD2		
80	40	PQDPLT		
82	41	PQDMEM		
84	42	PSHEAR		
86	43	PTWIST		
88	44	PMASS		
90	45	PDAMP		
92	46	PELAS		
94	47	CØNRØD		
96	48	CRØD		

DATA BLOCK AND TABLE DESCRIPTIONS

<u>Word No.</u> <u>In IFPX1</u>	<u>Bit No.</u> <u>In IFPX0</u>	<u>Contents</u>	<u>Output</u> <u>Only (PMR)</u>	<u>Supported</u>
98	49	CTUBE		
100	50	CVISC		
102	51	CBEAM		No
104	52	CTRIA1		
106	53	CTRIA2		
108	54	CTRBSC		
110	55	CTRPLT		
112	56	CTRMEM		
114	57	CQUAD1		
116	58	CQUAD2		
118	59	CQDPLT		
120	60	CQDMEM		
122	61	CSHEAR		
124	62	CTWIST		
126	63(word 3)	CØNM1		
128	64	CØNM2		
130	65	CMASS1		
132	66	CMASS2		
134	64	CMASS3		
136	68	CMASS4		
138	69	CDAMP1		
140	70	CDAMP2		
142	71	CDAMP3		
144	72	CDAMP4		
146	73	CELAS1		
148	74	CELAS2		
150	75	CELAS3		
152	76	CELAS4		
154	77	MAT1		

EXECUTIVE TABLE DESCRIPTIONS

<u>Word No.</u> <u>In IFPX1</u>	<u>Bit No.</u> <u>In IFPX0</u>	<u>Contents</u>	<u>Output</u> <u>Only (PMR)</u>	<u>Supported</u>
156	78	MAT2		
158	79	CTRIARG		
160	80	CTRAPRG		
162	81	DEFØRM		
164	82	PARAM	Yes	
166	83	MPCADD		
168	84	LØAD		
170	85	EIGR		
172	86	EIGB		
174	87	EIGC		
176	88	REACT		
178	89		Yes	
180	90	MATS1		
182	91	MATT1		
184	92	ØMIT1		
186	93	TABLEM1		
188	94(word 4)	TABLEM2		
190	95	TABLEM3		
192	96	TABLEM4		
194	97	TABLES1		
196	98	TEMPD		
198	99	TABLES2		No
200	100	TABLES3		No
202	101	TABLES4		
204	102	MATT2		
206	103	MATS2		No
208	104	CTØRDRG		
210	105	SPØINT		
212	106	SEQD		FØRCE
214	107	SEQDBFE		FØRCE

DATA BLOCK AND TABLE DESCRIPTIONS

<u>Word No. In IFPX1</u>	<u>Bit No. In IFPX0</u>	<u>Contents</u>	<u>Output Only (PMR)</u>	<u>Supported</u>
216	108	QDSEP		FØRCE
218	109	SPQUAD1		FØRCE
220	110	SPQUAD2		FØRCE
222	111	SPQDMEM		FØRCE
224	112	SPQDPLT		FØRCE
226	113	ZI		FØRCE
228	114	CTRIA3		FØRCE
230	115	PTRIA3		FØRCE
232	116	SETRBFE		FØRCE
234	117	VECDN		FØRCE
236	118	VECGP		FØRCE
238	119	DMI		
240	120	DMIG		
242	121	PTØRDRG		
244	122	MAT3		
246	123	DLØAD		
248	124	EPØINT		
250	125(word 5)	FREQ1		
252	126	FREQ		
254	127	NØLIN1		
256	128	NØLIN2		
258	129	NØLIN3		
260	130	NØLIN4		
262	131	RLØAD1		
264	132	RLØAD2		
266	133	TABLED1		
268	134	TABLED2		
270	135	SEQEP		
272	136	TF		
274	137	TIC		

EXECUTIVE TABLE DESCRIPTIONS

<u>Word No. In IFPX1</u>	<u>Bit No. In IFPX0</u>	<u>Contents</u>	<u>Output Only (PMR)</u>	<u>Supported</u>
276	138	TLØAD1		
278	139	TLØAD2		
280	140	TABLED3		
282	141	TABLED4		
284	142	TSTEP		
286	143	DSFACT		
288	144	AXIC		
290	145	RINGAX		
292	146	CCØNEAX		
294	147	PCØNEAX		
296	148	SPCAX		
298	149	MPCAX		
300	150	ØMITAX		
302	151	SUPAX		
304	152	PØINTAX		
306	153	SECTAX		
308	154	PRESAX		
310	155	TEMPAX		
312	156(word 6)	FØRCEAX		
314	157	MØMAX		
316	158	EIGP		
318	159	MASSC		
320	160	EDFIR		FØRCE
322	161	DFØRM		FØRCE
324	162	TABDMP1		
326	163	TABDMP2		
328	164	TABDMP3		
330	165	TABDMP4		
332	166	FREQ2		
334	167	CQUAD3		FØRCE

DATA BLOCK AND TABLE DESCRIPTIONS

<u>Word No.</u> <u>In IFPX1</u>	<u>Bit No.</u> <u>In IFPX0</u>	<u>Contents</u>	<u>Output</u> <u>Only (PMR)</u>	<u>Supported</u>
336	168	PQUAD3		FØRCE
338	169	SPQUAD3		FØRCE
340	170	SETR		FØRCE
342	171	SPTRIA1		FØRCE
344	172	SPTRIA2		FØRCE
346	173	SPTRMEM		FØRCE
348	174	SPTRBSC		FØRCE
350	175	SPTRPLT		FØRCE
352	176	SECL		FØRCE
354	177	SECP		FØRCE
356	178	SEPTRIA3		FØRCE
358	179	BARØR		
360	180	CBAR		
362	181	PBAR		
364	182	DAREA		
366	183	DELAY		
368	184	DPHASE		
370	185	PLFACT		
372	186	CGENEL		FØRCE
374	187	PGENEL		FØRCE
376	188	ELDELE		FØRCE
378	189	MATT3		
380	190	RFØRCE		
382	191	TABRND1		
384	192	TABRND2		
386	193	TABRND3		
388	194	TABRND4		
390	195	RANDPS		
392	196	RANDT1		
394	197	RANDT2		

EXECUTIVE TABLE DESCRIPTIONS

<u>Word No. In IFPX1</u>	<u>Bit No. In IFPX0</u>	<u>Contents</u>	<u>Output Only (PMR)</u>	<u>Supported</u>
396	198	PLØAD1		
398	199	PLØAD2		
400	200	DTI		
402-598	201-299	Not used		
600	300	CØUPMASS		
602	301	GRDPNT	Yes	
604	302	WTMASS	Yes	
606	303	IRES	Yes	
608	304	LFREQ		
610	305	HFREQ		
612	306	LMØDES		
614	307	G		
616	308	W3		
618	309	W4		
620	310	MØDACC		
622	311	MPC\$		
624	312	SPC\$		
626	313	LØAD\$		
628	314	METHØD\$		
630	315	DEFØRM\$		
632	316	TEMPLD\$		
634	317	TEMPMT\$		
636	318	IC\$		
638	319	AØUT\$	Yes	
640	320	LØØP\$		
642	321	LØØP1\$		
644	322	DLØAD\$		
646	323	FREQ\$		
648	324	TF\$		

DATA BLOCK AND TABLE DESCRIPTIONS

<u>Word No.</u> <u>In IFPX1</u>	<u>Bit No.</u> <u>In IFPX0</u>	<u>Contents</u>	<u>Output</u> <u>Only (PMR)</u>	<u>Supported</u>
650	325	PLØT\$	Yes	
652	326	TSTEP\$		
654	327	PØUT\$	Yes	
656	328	TEMPMX\$		
658	329	DSCØ\$		
660	330	K2PP\$		
662	331	M2PP\$		
664	332	B2PP\$		
666	333	CMETHØD\$		
668	334	SDAMP\$		
670	335	INERTIA\$		FØRCE
672	336	NLFØRCE\$		
674	337	XYØUT\$	Yes	
676	338	DELETES		FØRCE
678	339	RANDØM\$		
680	346	AXYØUT\$	Yes	
682	341	NØLØØP\$ Not Used		

EXECUTIVE TABLE DESCRIPTIONS

2.4.2.10 ITEM DT (Substructure item properties)

Description

ITEM DT contains most of the data required to describe a substructure item. The table is used by most substructure utilities when processing the SØF.

Created in Module

The ITEM DT values are initialized by block data ITEM BD which is included in any link which requires the SØF.

Table Format

Word	1	NITEM			}	Header
	2	ITEM NAME				
	3	ITEM TYPE				
	4	NUMB*10000	FIRST* 100	INC * 1	}	Entry #1
	5	IMAGE SUBSTRUCTURE DATA				
	6	SECONDARY SUBSTRUCTURE DATA				
	7	HIGHER LEVEL SUBSTRUCTURE DATA				
	8	EDIT DATA			}	Entry #2
	.					
	.					
	.					
	.					
	N					

DATA BLOCK AND TABLE DESCRIPTIONS

<u>Word</u>	<u>Item</u>	<u>Description</u>
1	NITEM	Number of items described in the table
Words 2-8 describe a sample 7 word entry		
2	NAME	Item name, 4 bcd characters
3	TYPE	Item type - 0 table item 1 matrix item
4	Describes modifications to be performed to table items during EQUIV	
	NUMB	Group 0 word with number of component names to be modified (*10000)
	FIRST	Group 0 word which contains first component name to be modified (*100)
	INC	Number of words in each name group (*1)
5	IMAGE	describes storage for image substructure data 0 - data is stored in primary 1 - data is stored in image
6	SECONDARY	describes storage for secondary substructure data (same as above)
7	HIGHER	describes relationship between item and higher level substructures 0 - item does not relate to higher level substructure 1 - item describes higher level substructure (example HØRG)
8	EDIT	Each bit is set if the item belongs to that edit group. (For example all substructures have bit 5, 2 ⁵ =32, set)

Notes

- ITEMDT is contained in named common block /ITEMDT/
- The following listing, of block data ITEM DT, gives the current item structure for the SØF.

EXECUTIVE TABLE DESCRIPTIONS

NAME	TYPE	EQUIV	IMAGE	SECONDARY	HIGHER	EDIT
EQSS	0	3005002	1	1	0	32
BGSS	0	0	0	0	0	32
CSTM	0	0	0	0	0	32
LØDS	0	4005002	1	1	0	36
PLTS	0	3004014	1	1	0	32
KMTX	1	0	0	0	0	33
KMTX	1	0	0	0	0	34
PVEC	1	0	0	0	0	36
PØVE	1	0	0	1	1	48
UPRT	1	0	0	1	1	48
HØRG	1	0	0	1	1	560
UVEC	1	0	1	1	0	40
QVEC	1	0	1	1	0	40
SØLN	0	0	1	1	0	40
PAPP	1	0	0	0	0	100
PØAP	1	0	0	1	1	112
LØAP	0	4005002	1	1	0	100
LMTX	1	0	0	1	1	48
GIMS	1	0	0	1	1	48
PHIS	1	0	0	1	1	288
LAMS	0	0	0	1	1	288
K4MX	1	0	0	0	0	160
BMTX	1	0	0	0	0	160
PHIL	1	0	0	1	1	288
HLFT	1	0	0	1	1	560

MISCELLANEOUS TABLE DESCRIPTIONS

2.5 MISCELLANEOUS TABLE DESCRIPTIONS.

The following is an alphabetical index of miscellaneous table descriptions.

<u>Section Number</u>	<u>Table Name</u>	<u>Where Stored</u>	<u>Page Number</u>
2.5.2.2	BITPØS	/BITPØS/	2.5-8
2.5.2.4	CHAR94	/CHAR94/	2.5-11
2.5.2.7	CHRDW	/CHRDW/	2.5-14
2.5.1.6	DESCRP	/DESCRP/	2.5-3
2.5.2.1	GPTA1	/GPTA1/	2.5-6
2.5.1.5	MSGX	/MSGX/	2.5-3
2.5.1.8	NAMES	/NAMES/	2.5-4
2.5.2.8	NTIME	/NTIME/	2.5-15
2.5.1.1	ØSCENT	/ØSCENT/	2.5-2
2.5.1.2	ØUTPUT	/ØUTPUT/	2.5-2
2.5.2.3	PLTDAT	/PLTDAT/	2.5-9
2.5.1.3	STIME	/STIME/	2.5-2
2.5.2.6	SYMBLS	/SYMBLS/	2.5-13
2.5.1.7	TWØ	/TWØ/	2.5-4
2.5.1.9	TYPE	/TYPE/	2.5-4
2.5.1.4	XMDMSK	/XMDMSK/	2.5-3
2.5.2.5	XXPARM	/XXPARM/	2.5-12

2.5.1 Miscellaneous Tables Which Are Permanently Core Resident.

2.5.1.1 ØSCENT (ØSCAR Entry)

Description

A 200 word storage array containing the ØSCAR entry (record) currently being processed.

Created in Module

The entry is read from the ØSCAR and stored in ØSCENT by the XSEMI (section 3.3.7) sub-routine. Other executive routines that require details of the current entry will search ØSCENT.

Table Format

The ØSCENT format is identical to the ØSCAR (section 2.4.2.1) entry it currently contains.

2.5.1.2 ØOUTPUT (Output headings)

Description

A storage array containing problem title, subtitle, label and various headings required by the PAGE (section 3.4.24) routine to properly annotate the NASTRAN output.

Created in Module

The title, subtitle and label are taken from Case Control Deck cards and stored in ØOUTPUT by IFPI (section 4.3). Other heading lines may be stored by output modules prior to calling PAGE.

Table Format

ØOUTPUT contains sufficient space for seven 128 character lines. The first three lines contain the title, subtitle, and label. The subsequent three lines contain local headings, and the final line contains the plotter ID. Since 4 characters occupy each computer word, the ØOUTPUT array requires 224 words of storage.

2.5.1.3 STIME (Solution Time)

Description:

A storage cell containing the user's estimated solution time.

Created in Module

The estimated solution time is taken from a Executive Control Deck card and stored into STIME by XCSA (section 4.2)

Table Format

STIME consists of a single cell containing the estimated time in integer seconds.

MISCELLANEOUS TABLE DESCRIPTIONS

2.5.1.4 XMDMSK (Executive Module Decision Mask)

Description

Contains the 155 bit master module execution mask (see section 1.10) and a cell indicating checkpoint status.

Created in Module

The 155 bit master module execution mask is generated and used by XGPI (section 4.7). The checkpoint status set on (YES) by XCSA (section 4.2) by the presence of a CHPNT = YES card in the Executive Control Deck.

Table Format

The 155 bit mask occupies the low order 31 bits of the first five words of XMDMSK. The sixth word is the checkpoint status (flag).

2.5.1.5 MSGX (Message Queue)

Description

A queue table to hold four word NASTRAN information and error messages between the time they are generated by a module and printed by the message writer, MSGWRT (section 3.4.26).

Created in Module

Messages may be generated by any NASTRAN module through a call to MESSAGE (section 3.4.25).

Table Format

Word 1	-	Number of messages queued.
Word 2	-	Maximum number of messages queue can hold
Word 3-6	-	Four word message entry (typical)
Word 6-end	-	Additional four word message entries

2.5.1.6 DESCRP (Matrix Description)

Description

A storage block used by subroutine INTPK (section 3.5.3) to buffer the matrix unpacking procedure. This buffering reduces the number of I/O accesses to the particular matrix data block.

Created in Module

DESCRP is filled and used exclusively by INTPK

Table Format

An array with the first word defining the length of the array.

2.5.1.7 TWØ (Powers of Two)

Description

A 32 word array with each word (starting with 1 in the 32nd word) containing the next power of two.

Created in Module

The 32 integer values are defined within the NASTRAN system block data program (SEMDBD).

Table Format

Word 32 - 1
 Word 31 - 2
 Word 30 - 4
 Word 29 - 8
 etc.

2.5.1.8 NAMES (Symbolic Names)

Description

A series of symbolic names identified with their NASTRAN numeric equivalents. Defines values for GINØ file options, arithmetic types and matrix forms.

Created in Module

The values are defined within the NASTRAN system block data program (SEMDBD).

Table Format

<u>Word</u>	<u>SYMBOL</u>	<u>VALUE</u>	<u>Word</u>	<u>SYMBOL</u>	<u>VALUE</u>	<u>Word</u>	<u>SYMBOL</u>	<u>VALUE</u>
1	RD	= 2	7	EØFN RW	= 3	13	RECT	= 2
2	RDREW	= 0	8	RSP	= 1	14	DIAG	= 3
3	WRT	= 3	9	RDP	= 2	15	UPPER	= 4
4	WRTREW	= 1	10	CSP	= 3	16	LOWER	= 5
5	REW	= 1	11	CDP	= 4	17	SYM	= 6
6	NØREW	= 2	12	SQUARE	= 1	18	RØW	= 7
						19	IDENT	= 8

2.5.1.9 TYPE (Number Types)

Description

A series of properties are defined as a function of a number type. The type may be Real Single Precision (RSP-1), Real Double Precision (RDP-2), Complex Single Precision (CSP-3), or Complex Double Precision (CDP-4). The properties that may be returned include precision (single, double), number of words, and real or complex.

Created in Module

The properties are defined within the NASTRAN system block data program (SEMDBD).

MISCELLANEOUS TABLE DESCRIPTIONS

Table Format

<u>Word</u>	<u>Property (Values)</u>	<u>Type</u>	
1	1	Precision	(RSP)
2	2	Precision	(RDP)
3	1	Precision	(CSP)
4	2	Precision	(CDP)
5	2	Words	(RSP)
6	4	Words	(RDP)
7	1	Words	(CSP)
8	1	Words	(CDP)
9	2	Real/Complex	(RSP)
10	2		(RDP)
			(CSP)
			(CDP)

Example

Assume the number of words required to contain a Complex Single Precision (CSP-3) is desired. The third item in the Words array is indexed and found to contain a 2 (words).

DATA BLOCK AND TABLE DESCRIPTIONS

2.5.2 Miscellaneous Tables Not Permanently Core Resident

2.5.2.1 /GPTA1/

Purpose

To describe connection and property characteristics of each element. /GPTA1/ is used in modules GP1, GP2, GP3, TA1, SMA1, SMA2, DSMG1, SDR2, PLTSET, and SSG1, and is initialized by the block data program GPTABD and subroutine DELSET.

Description

<u>Group</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
1	1	I	Number of entries (i.e., elements) in table (NENT)
	2	I	Pointer to first word of last entry in table
	3	I	Number of words per entry in table
2-(NENT+1)	1-2	B	Name of element (e.g., RØD)
	3	I	Internal element identification number
	4-5	I	ECT record ID and trailer bit for LØCATE
	6	I	Number of words per entry on ECT
	7-8	I	EPT record ID and trailer bit for LØCATE
	9	I	Number of words per entry on EPT
	10	I	Number of grid points per element
	11	I	+1 : Scalar element with grid point and component code
			0 : Not a scalar element
			-1 : Scalar element with scalar points only
	12	I	Number of words per entry on EST
	13	I	Position of first grid point in ECT entry
	14	I	Temperature data
	15	I	Temperature data count
	16	I	2 Hollerith symbols for plotting symbol. If "xx" the element is not plottable. Word 10 of group 2 thru (NENT+1) must be 2 to 20 and word 11 of group 2 thru (NENT+1) must be zero if the element is plottable.
	17	I	Number of words SDR2 passes from Phase 1 element routines to Phase 2 element routines
	18	I	Count of words SDR2 outputs for real stresses
	19	I	Count of words SDR2 outputs for real forces
	20	I	Pointer into an SDR2D table for combining of real stresses to form complex stress outputs

MISCELLANEOUS TABLE DESCRIPTIONS

<u>Group</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
2-(NENT+1)	21	I	Pointer into an SDR2D table for combining of real forces to form complex force outputs
	22	I	SMA1 element overlay limb
	23	I	SMA2 element overlay limb
	24	I	SMA3 element overlay limb
	25	I	Number of degrees-of-freedom per grid point for element

Note

Subroutine DELSET should be called prior to any usage of this block data program to set 'dummy' elements (see Section 3.4.72).

DATA BLOCK AND TABLE DESCRIPTIONS

2.5.2.2 BITPOS

Purpose

To provide pointers into USET and USETD words for interpreting the nested vector sets in NASTRAN.

Words 1-32 are the bit numbers and words 33-64 are the corresponding alphanumeric mnemonics.

Description

<u>Word</u>	<u>Item</u>	<u>Value</u>	<u>Word</u>	<u>Item</u>	<u>Value</u>
1	UM bit number	32	33	M	
2	UO bit number	30	34	Ø	
3	UR bit number	29	35	R	
4	USG bit number	23	36	SG	
5	USB bit number	22	37	SB	
6	UL bit number	24	38	L	
7	UA bit number	25	39	A	
8	UF bit number	26	40	F	
9	US bit number	31	41	S	
10	UN bit number	27	42	N	
11	UG bit number	28	43	G	
12	UE bit number	21	44	E	
13	UP bit number	20	45	P	
14	UNE bit number	19	46	NE	
15	UFE bit number	18	47	FE	
16	UD bit number	17	48	D	
17	UPS bit number	16	49	PS	
18	USA bit number	15	50	SA	
19	UK bit number	14	51	K	
20	UPA bit number	13	52	PA	
21-32	Empty	0	53-64	Empty	**

Notes:

1. The first 32 words are integer and the second 32 are alphanumeric.
2. An example of usage is shown below:

```

COMMON / TWØ / ITWØ(32)
COMMON / BITPOS / K(32,2)
C   DETERMINE IF USET ENTRY IS A MEMBER ØF THE Ø-SET.
      L = K(2,1)
      IF (ANDF(USET(I),ITWØ(L)).EQ.0) GØ TØ α

```


2.5.2.3 PLTDAT

Purpose

To define plotter-dependent parameters.

Description

This table is defined in the PLØTBD block data subprogram. The table is divided into N+2 20-word sections, where N = number of plotters acceptable by the NASTRAN plotting software. Sections 3 to N+2 are the only sections initialized, because each contains values which are dependent upon the plotter hardware. Section 1 contains values which may vary within the limits of the hardware, and Section 2 is simply a duplicate of one of the last N sections corresponding to the plotter of interest.

Section 2 must be filled in by the module writer. The format of Sections 2 to N+2 is as follows:

<u>Word</u>	<u>Type</u>	<u>Name</u>	<u>Description</u>
1-2	R	XYMAX	Maximum x and y coordinate values acceptable by the plotter.
3	R	CNTSIN	Number of plotter counts/inch on paper.
4-5	R	CNTCHR	Number of plotter counts per character in the x and y directions.
6	R	MAXLEN	Maximum length of a line segment.
7	I	NPENS	Maximum number of pens or line density available on the plotter.
8-9	R	ØRIGIN	For incremental plotters, the current pen position relative to the lower left corner of the plot. <u>Otherwise</u> , the location of the lower left corner of the plotter relative to its true physical origin.
10	I	PLTYPE	{ +1, +2, or +3 if the plotter is a microfilm, table or drum plotter, respectively, with typing capability. -1, -2, or -3 if the plotter is a microfilm, table or drum plotter, respectively, with no typing capability (i.e., all characters must be drawn).
11	B	PLTAPE	{ PLT1 if an <u>even</u> parity plot tape is to be generated for this plotter. PLT2 if an <u>odd</u> parity plot tape is to be generated for this plotter.
12	I	PBFSIZ	Plot tape physical record size (number of characters)
13	I	EØF	{ 0 if an end-of-file is to be written after every plot. 1 if no end-of-file is to be written on the plot tape.
14-20			Undefined

DATA BLOCK AND TABLE DESCRIPTIONS

Section 1 must also be filled in by the module writer. However, unlike Section 2, some of the parameters may vary from plot to plot, as long as they remain within the limitations imposed by the plotter hardware. The format of Section 1 is as follows.

<u>Word</u>	<u>Type</u>	<u>Name</u>	<u>Description</u>
1	I	MØDEL	Plotter model index
2	I	PLØTER	Plotter index
3-6	R	REG	Plotter region (x_{min} , y_{min} , x_{max} , y_{max}) in which the current picture is being drawn. These values must be some fraction (between 0 and 1) of words 7 and 8 (AXYMAX).
7-8	R	AXYMAX	Size of the paper used (x,y), less the borders, in plotter units.
9-10	R	XYEDGE	Size of the borders (x,y) in plotter units.
11	I	CAMNUM	Current selected camera. This word need not be filled in, because it is set and used as a communication between the SELCAM and SKPFRM subroutines.
12-20			Undefined

Usage

Sections 1 and 2 are normally setup by the FNDPLT subroutine, except for the plotter region values (REG). These values must be setup by the module writer himself. It is essential that both these sections be correctly setup, because they are referenced by the entire NASTRAN plotter software package.

If Sections 1 and 2 are correctly setup by the module writer, he need not subsequently worry about such things as compensating for paper margins or different physical plotter origins. He need only assume that the plotter origin is located at the lower left corner of the paper where the left and bottom borders intersect. The NASTRAN plotter software will automatically compensate for the borders and the physical origin.

MISCELLANEOUS TABLE DESCRIPTIONS

2.5.2.4 CHAR94

Purpose

To provide a table of characters used to generate plot tapes as if the computer were always an IBM 7094. This table however is independent of the actual computer used.

Description

This is a 240 word table defined in the PLØTBD block data subprogram. It is divided into four equal sections of 60 words each. Each entry in each section has a parallel entry in the other three sections.

Section I is a string of all the Hollerith characters acceptable by the plot modules of the form 1Hx, where x is a Hollerith character.

Section II contains the integer equivalents of the IBM 7094 internal binary characters in the same order as Section I. However, near the end of this section are integers representing various additional characters not in Section I. These additional characters cannot be expressed in the form 1Hx and are used for special plotter commands. Each entry in this table is a right-adjusted two-digit integer with leading zeroes.

Section III contains the integer equivalents of the IBM 7094 BCD characters as they would appear on an even parity tape written on an IBM 7094, in the same order and form as in Section II.

Section IV contains the integer equivalents of the CDC display character codes so as to produce an even parity BCD plot tape as if written on an IBM 7094, in the same order and form as in Section II.

The sequence of characters in each section is as follows:

0	1	2	3	4	5	6	7	8	9
A	B	C	D	E	F	G	H	I	J
K	L	M	N	Ø	P	Q	R	S	T
U	V	W	X	Y	Z	()	+	-
*	/	=	.	,	\$	'	b		

character 49 = end of record mark

character 50 = end of file mark

characters 51-53 = special characters.

characters 54-60 = 0

Note

In Section I, characters 49-60 = 0.

Usage

Section I is basically used for calculating an index into the other two sections by comparing an arbitrary Hollerith character with each character in Section I until a match is found. Once this is done, the index is used to extract the corresponding entry from either Section II or III, depending on whether an odd or even parity plot tape is being generated. If the computer is an IBM 7094, only Section II is used, and if the computer is a CDC 6600 and an even parity plot tape is being generated, Section IV is used instead of Section III.

DATA BLOCK AND TABLE DESCRIPTIONS

2.5.2.5 XXPARM

Purpose

To define the plot tape buffer size, the camera to be used, the number of blank frames of film to be inserted between plots, the plotter model name, and the paper size to be used on table plotters.

Description

This table is defined as follows in the PLØTBD block data subprogram.

<u>Word</u>	<u>Type</u>	<u>Name</u>	<u>Description</u>
1	I	BUFSIZ	Plot tape buffer size
2	I	CAMERA	Plotter camera to be used (appropriate only on a <u>microfilm</u> plotter).
3	I	BFRAMS	Number of blank frames of <u>film</u> to be inserted between plots (appropriate only on a <u>microfilm</u> plotter).
4-5	I, or B	PLTMDL	Plotter model identification.
8-9	R	PAPSIZ	Width and height of the paper to be used (appropriate only on <u>table</u> plotters).

Usage

The initial values of these variables are as follows:

BUFSIZ = must be set by the module writer

CAMERA = 2 (paper output only)

BFRAMS = 1

PLTMDL = 4020, 0 (integer)

PAPSIZ = 8.5, 11.0

This table's actual size is 157 words. The remainder of the table is initialized for the structural plotter module, PLØT, but may be used by the programmer for anything he desires in other plotting modules.

MISCELLANEOUS TABLE DESCRIPTIONS

2.5.2.6 SYMBLS

Purpose

To provide a table of indices into the CHAR94 and CHRDRW tables used to type or draw pre-defined plotter symbols.

Description

The table is defined in the PLØTBD block data subprogram. There is room for up to 20 indices for each plotter. However, the same number of indices must be defined for each plotter. The format of the table is as follows:

<u>Word</u>	<u>Type</u>	<u>Description</u>
0	I	Number of symbols defined for each plotter (currently = 9).
1-20	I	Symbol indices for plotter 1.
21-40	I	Symbol indices for plotter 2.
41-60	I	Symbol indices for plotter 3.
:	:	:

There are as many groups of symbol indices as there are available plotters. The symbols defined for each plotter are as follows:

Symbol 1 = x
 Symbol 2 = *
 Symbol 3 = +
 Symbol 4 = -
 Symbol 5 = . (dot, not a period).
 Symbol 6 = circle
 Symbol 7 = square
 Symbol 8 = diamond
 Symbol 9 = triangle

Should any of these symbols not be available on a plotter, a substitution of another symbol must be made.

Usage

This table is used by the SYMBØL subroutine.

2.5.2.7 CHRDRW

Purpose

To define the combination of lines needed to draw alphanumeric characters and symbols.

Description

This table is defined in the PLØTBD block data subprogram. The table is divided into two sections. Section I is a list of indices into Section II, used to locate the data needed to draw characters. The first 48 indices in Section I correspond to the 48 characters listed in Section I of the CHAR94 table. The last 7 indices are used for drawing the special characters listed in the SYMBLS table. If an index is negative, it is an index into Section I instead of Section II. This occurs when a duplicate character exists (e.g., a zero, the letter "Ø", and the symbol for a circle).

Section II of this table defines the (integer) coordinates of the starting and ending points of the straight lines to be drawn in order to draw a character or symbol. In general, the necessary straight lines are contiguous, so that the end point of one line is the starting point of the next, etc. In some cases, this is either impractical or impossible (e.g., *, +, =, etc.). In such a situation, the starting point of a line is negative, meaning that it is not to be connected to the end point of the preceding line.

The characters defined in Section II are based upon 6x6 square characters. The values in this section are simply integer coordinates within a 6x6 square.

The format of this table is as follows:

<u>Word</u>	<u>Type</u>	<u>Name</u>	<u>Description</u>
0	I	LSTCHR	Name of characters and symbols referenced in Section I (=52).
1-60	I	CHRIND	<u>Section I</u> - "LSTCHR" indices into Section II.
61-760	I	CHR	<u>Section II</u> = (x,y) pairs defining the lineal representation of 6x6 square characters.

Usage

This table is used by the DRWCHR subroutine.

MISCELLANEOUS TABLE DESCRIPTIONS

2.5.2.8 NTIME

Purpose

Defines timing constant data for all machine configurations.

Description

This table is defined in the NTMXBD block data subprogram. Values defined in this table are in microseconds and are obtained by running the TIMETEST module (see NASTRAN User's Manual Section 5.5 and the Programmer's Manual Section 4.127) on different computer configurations.

The format of this table is as follows:

	<u>Word</u>	<u>Type</u>	<u>Name</u>	<u>Description</u>
	0	I	NMACHS	Number of machine configurations supported by NASTRAN.
	1	I	NITEMS	Number of timing data constants items in table.
	2	I	TMIO	Average time for a read and write operation per word.
	3	I	TMBPAK	Average time for a BLDPK operation.
	4	I	TMIPAK	Average time for an INTPK operation.
	5	I	TMPAK	Average time for a PACK operation.
	6	I	TMUPAK	Average time for an UNAPCK operation.
	7	I	TMGSTR	Average time for a GETSTR operation.
	8	I	TMPSTR	Average time for a PUTSTR operation.
	9	I	TMTRSP	Average time for a multiply-tight loop operation in single precision.
Repeated for each Machine Configuration	10	I	TMTRDP	Average time for a multiply-tight loop operation in real double precision.
	11	I	TMTCSF	Average time for a multiply-tight loop operation in complex single precision.
	12	I	TMTCDP	Average time for a multiply-tight loop operation in complex double precision.
	13	I	TMLRSP	Average time for a multiply-loose loop operation in real single precision.
	14	I	TMLRDP	Average time for a multiply-loose loop operation in real double precision.
	15	I	TMLCSF	Average time for a multiply-loose loop operation in complex single precision.
	16	I	TMLCDP	Average time for a multiply-loose loop operation in complex double precision.
	17	I	-	Unused.

DATA BLOCK AND TABLE DESCRIPTIONS

Each of the timing constant arrays are sixteen words in length and each array is associated with one computer configuration. The following lists the computer configuration associated with each word.

<u>Word</u>	<u>Configuration</u>
2	Vacant
18	IBM 360/91,95
34	UNIVAC 1108
50	CDC CYBER 175
66	IBM 360/50
82	IBM 360/65
98	IBM 360/75
114	IBM 360/85
130	IBM 360/195
146	CDC 6400
162	IBM 370/155
178	IBM 370/165
194	IBM 370/145
210	IBM 370/158
226	IBM 370/168
242	CDC CYBER 174
258	UNIVAC 1110
274	CDC CYBER 173
290	CDC CYBER 176
306	CDC 6600

Usage

This table is used by modules like MPYAD and SDCOMP for calculating timing estimates for operations such as matrix multiplication and decomposition. (See module MPYAD - Section 3.5.12 for an example of the use of this table).

DATA BLOCK DESCRIPTIONS

2.6 SUBSTRUCTURE DATA ITEMS DESCRIPTION

The format for substructure data blocks to be stored on the Substructure Operating File (SØF) is similar to the storage of NASTRAN data blocks with the following exceptions:

1. The SØF is a separate physical file subdivided into data "items." Each "item" is equivalent to a NASTRAN data block.
2. Each substructure item may be subdivided into several "groups" of arbitrary length. Each group is equivalent to a logical record in NASTRAN.
3. In order to operate on the separate items in the SØF, a random access pointer table is maintained in the MDI (Master Data Index) item on the SØF. The storage space on the SØF file is dynamically allocated so that the user may maintain control over the use of that space. As processing continues the SØF will rapidly fill up. Provisions have been made, therefore, to give the user control over what is stored on the SØF, over what is removed from the SØF for storage elsewhere, and over what is removed from the SØF and destroyed.

SUBSTRUCTURE DATA ITEM DESCRIPTIONS

2.6.1 Substructure Data "Items" Description

The following is an alphabetical index of Substructure data "items" descriptions.

<u>Section Number</u>	<u>Item</u>	<u>Page Number</u>
2.6.1.1	BGSS	2.6-3
2.6.1.9	BMTX	2.6-15
2.6.1.2	CSTM	2.6-4
2.6.1.3	EQSS	2.6-5
2.6.1.9	GIMS	2.6-15
2.6.1.9	HLFT	2.6-15
2.6.1.9	HØRG	2.6-15
2.6.1.9	KMTX	2.6-15
2.6.1.9	K4MX	2.6-15
2.6.1.8	LAMS	2.6-14
2.6.1.9	LMTX	2.6-15
2.6.1.7	LOAP	2.6-13
2.6.1.4	LØDS	2.6-6
2.6.1.9	MMTX	2.6-15
2.6.1.9	PAPP	2.6-15
2.6.1.9	PHIL	2.6-15
2.6.1.9	PHIS	2.6-15
2.6.1.5	PLTS	2.6-7
2.6.1.9	PØAP	2.6-15
2.6.1.9	PØVE	2.6-15
2.6.1.9	PVEC	2.6-15
2.6.1.9	QVEC	2.6-15
2.6.1.6	SØLN	2.6-9
2.6.1.9	UPRT	2.6-15
2.6.1.9	UVEC	2.6-15

DATA BLOCK DESCRIPTIONS

2.6.1.1 BGSS

Description

The BGSS data defines the coordinates of the grid points in a substructure. The local coordinate system identification number (originally given in the basic substructure on the GRID data card) is followed by the X, Y, and Z locations in the basic coordinate system of the substructure.

The BGSS item is generated by modules SUBPH1, CØMB1, and REDUCE.

Item Format

<u>Group</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>	
0	1,2	B	Name of substructure	
	3		N_p - Number of internal points	
1	1		Coordinate system	} Repeated N_p times
	2-4		Basic coordinates (X,Y,Z)	
2			End-of-item	

SUBSTRUCTURE DATA ITEM DESCRIPTIONS

2.6.1.2 CSTM

Description

Coordinate System Transformation Matrices

The CSTM data is equivalent to the data block with the same name. The CSTM item for any substructure contains a transformation for each coordinate system defined in Phase 1 for each basic substructure which is a component of the substructure. The transformation is from global of the basic substructure to basic of the pseudostructure.

The CSTM item is generated in SUBPH1, COMB1, and REDUCE.

Item Format

<u>Group</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>	
0	1,2	B	Name of the substructure	
1	1	I	Coordinate system ID	} Repeated for each coordinate system
	2	I	Coordinate system type	
			1 - rectangular	
			2 - cylindrical	
			3 - spherical	
	3-5	R	Translation vector	
	6-14	R	Transformation matrix	
2			End-of-item	

DATA BLOCK DESCRIPTIONS

2.6.1.3 EQSS

Description

The EQSS data is used to convert basic substructure grid points on the bulk data cards to internal point numbers for substructure operations. In addition, the current grid point components are listed (i.e., those components which remain after reduction and constraint operations have been performed). The last group of data on the EQSS table is the scalar indices and components for each internal point number.

The EQSS item is generated in SUBPH1, COMB1, and REDUCE.

Item Format

<u>Group</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>	
0	1,2	B	Name of substructure	
	3	I	N_S - Number of contributing substructures	
	4	I	Number of internal points in combination	
	5,6	B	Name of basic substructure 1	
	7,8	B	Name of basic substructure 2	
	⋮		⋮	
1	1	I	Grid point ID	} Repeated for all points (external sort)
	2	I	I_p - Internal point index	
	3	I	Component code	
2,3,... N_S	(Group 1 is repeated for all contributing basic substructures)			
	⋮			
$N_S + 1$	1	I	Scalar index (row number)	} Repeated for all points (internal sort, position = I_p)
	2	I	Component code	
$N_S + 2$	End-of-item			

Notes:

1. If one of the component substructures in Group 0 is not a basic substructure, it represents a component of modal dof created in MREDUCE or CREduce. The following grid point numbers are assigned.

<u>Grid</u>	<u>Component</u>	<u>Description</u>
1-6	1	optional inertial dof
101-	1	modal dof

SUBSTRUCTURE DATA ITEM DESCRIPTIONS.

2.6.1.4 LØDS

Description

Directory of set ID's for loads defined in Phase 1. For each Phase 1 subcase of each basic substructure, LØDS contains the set ID identifying the applied load. It may be zero if no load was defined for a subcase.

The LØDS item is generated in SUBPH1, CØMB1, and REDUCE.

Item Format

<u>Group</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>	
0	1,2	B	Name of substructure	
	3	I	Total number of load vectors for this substructure	
	4	I	N_S - Number of basic substructures in this substructure	
	5,6	B	Name of basic substructure 1	
	7,8	B	Name of basic substructure 2	
	⋮	⋮	⋮	
1	1	I	NL - Number of load vectors for basic substructure 1	} Repeated for each basic substructure
	2	I	Set ID for load vector (subcase) 1	
	3	I	Set ID for load vector (subcase) 2	
	⋮			
	NL + 1		Set ID for load vector (subcase) NL	
$N_S + 1$			End-of-item	

DATA BLOCK DESCRIPTIONS

2.6.1.5 PLTS

Description

The PLTS item contains data necessary to produce undeformed substructure plots. The PLTS item of a pseudostructure contains only the names of the component basic substructures and the basic coordinate systems transformation data. The PLTS item of a basic substructure contains basic grid point coordinates, external grid point ID's, and grid point and element connection sets for plotting. Therefore, to plot a pseudostructure, the PLTS items of the pseudostructure and its component basic substructures must exist.

Basic substructure PLTS items are generated by module SUBPH1. Pseudostructure PLTS items are generated by modules COMB1 and REDUCE.

Item Format

<u>Group</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>	
0	1,2	B	Substructure name	
	3	I	Number of basic substructures	
	4,5	B	Basic substructure name	} Repeated for each basic substructure
	6-8	R	Translation vector	
	9-17	R	Transformation matrix	

The following groups exist for PLTS items of basic substructures only.

<u>Group</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>	
1	1	I	Coordinate system ID	} Repeated for all grid points
	2-4	R	Basic grid point coordinates	
2	1	I	External grid or scalar number	} Repeated for all grid and scalar points
	2	I	Internal number	
3	1	I	Number of grid points in element set	
	2-(NGP+1)	I	Pointers to the grid points in this element set	
			If = 0, the grid point is not in this set	
			If ≠ 0, this is an internal index relative to only the grid	
			points in this element set. If negative, this grid point	
			is to be excluded when applying grid labels or symbols.	

31

SUBSTRUCTURE DATA ITEM DESCRIPTIONS

<u>Group</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>	
4	1	B	Element plot symbol (2 BCD characters)	}
	2	I	NGPEL - Number of grid points per element of this type. 1. NGPEL < 3 - one dimensional element 2. $3 \leq \text{NGPEL} \leq 4$ and plot symbol is not "TE" - the first and last grid are also connected. 3. $4 < \text{NGPEL}$ or plot symbol is "TE" - special line connection pattern in LINEL is used.	
	3	I	Element identification number. If zero, there are no more elements of this type.	
	4		Index of this element type in ECT.	
	5-(NGPEL+3)	I	Grid point connection indices for the grid point sets array in group 3.	}
5			End of item	

Repeated for all element types in the set

Repeated for elements of this type

DATA BLOCK DESCRIPTIONS

2.6.1.6 SØLN

Description

The SØLN item contains data defining the solution vectors obtained from the NASTRAN analysis. The data may be of two forms, if a static analysis has been performed, the data consists of the load vectors defined for each basic substructure load and each solution subcase. If a real eigenvalue analysis has been performed the data consists of eigenvalue and eigenvector parameters (similar to data block LAMA).

The SØLN item is created by module RCØVR.

Item Format

The contents of SØLN for static analysis are:

<u>Group</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>	
0	1,2	B	Name of the substructure which was solved	
	3	I	Rigid Format number (1 or 2)	
	4	I	NS - Number of basic substructures	
	5	I	NC - Number of subcases	
	6,7	B	Basic substructure name	} Repeated for all component basic substructures
	8	I	Number of loads for this substructure	
1 - NC	1	I	Number of load vectors participating in this subcase. If negative, this is a SYMCØM or SUBCØM subcase.	} Repeated for all subcases
	2	I	Internal load vector number	
	3	R	Scale factor	
NC + 1			End-of-item	

The contents of SØLN for real eigenvalue analysis are:

<u>Group</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0	1,2	B	Name of the substructure which was solved
	3	I	Rigid Format number (3)
	4	I	Number of eigenvalues

SUBSTRUCTURE DATA ITEM DESCRIPTIONS

<u>Group</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>	
1	1	I	Mode number	} Repeated for each eigenvalue
	2	I	Extraction order	
	3	R	λ - eigenvalue	
	4	R	$\omega = \sqrt{ \lambda }$	
	5	R	$f = \omega/2\pi$	
	6	R	Generalized mass	
	7	R	Generalized stiffness	
			End-of-item	

The contents of S0LN for complex eigenvalue analysis are:

<u>Group</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>	
0	1,2	B	Name of the substructure which was solved	
	3	I	Rigid Format Number (3)	
	4	I	Number of eigenvalues	
1	1	I	Mode number	} Repeated for each eigenvalue
	2	I	Extraction order	
	3	R	Real part of Eigenvalue	
	4	R	Imaginary part of Eigenvalue	
	5	R	$ \text{Im}(\lambda) / 2\pi$	
	6	R	$-2 + \text{Re}(\lambda) / \text{Im}(\lambda) $	
	7	R	Not used	

The contents of S0LN for dynamic analysis are:

<u>Group</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0	1,2	B	Name of substructure which was solved
	3	I	Rigid Format Number (8 or 9)
	4	I	NS - Number of basic substructures
	5	I	NSTEP - Number of time or frequency steps

DATA BLOCK DESCRIPTIONS

<u>Group</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>	
	6,7	B	Basic Substructure Name	} repeated for all component basic substructures
	8	I	Number of loads for this substructure	
	6+3*NS	I	Number of static load vectors used in this solution	
	6+3*NS+1	I	Internal load vector number	} repeated for each load vector
1	1	R	Time or frequency value	} repeated for each step
2	1	R	Scale factor	} repeated for each load
NSTEP + 2			End-of-Item	

Notes:

1. The name of the substructure which was solved (first 2 words of group 0) is the name which appeared on a SØLVE substructure command.
2. During the RECOVER operation, the SØLN items of lower level substructures are created by editing the SØLN item of the substructure which was solved.
3. The internal load vector number in the statics or dynamics version is the column number of the load matrix stored as the PVEC item of the same substructure for which the SØLN item is stored.
4. The scale factors in the statics version are computed from the factors on LOADC bulk data cards and/or SUBSEQ or SYMSEQ case control cards.
5. Group 1 in the eigenvalue version is identical to record 2 of the LAMA or CLAMA data block.
6. If the number of eigenvalues is zero, group 1 does not exist in the eigenvalue version.
7. Group 1 in the dynamics version is identical to record 0 of the PPF data block for R.F. 8 or the TOL data block for R.F. 9.
8. Groups 2 through NSTEP+1 in the dynamics version each form one column of the F_i matrix. This matrix gives the load factor for each static load set for time step.

SUBSTRUCTURE DATA ITEM DESCRIPTIONS

9. The scale factors in the dynamics version are computed from the TLOADi, RLOADi, TABLEDi and LOADC bulk data cards.
10. The scale factors in the dynamics version are real for R.F. 9 and complex for R.F. 8.

DATA BLOCK DESCRIPTIONS

2.6.1.7 LØAP

Description

Directory of set ID's for loads defined in Phase 1 with option PA. For each Phase 1 subcase of each basic substructure, LØAP contains the set ID identifying the applied load. It may be zero if no load was defined for a subcase.

The LØAP item is generated in SUBPH1, COMB1, and REDUCE.

Item Format

See 2.6.4 LØDS

SUBSTRUCTURE DATA ITEM DESCRIPTIONS

2.6.1.8 LAMS

The LAMS item contains data defining the eigenvalues used in the modal reduction. This data is identical to the real or complex eigenvalue analysis SØLN item. An additional group is also present which describes the manner in which each mode was used in the reduce.

The LAMS item is created by module MRED2 or CMRED2.

Item Format

<u>Group</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>
0	see SØLN	data (sec 2.6.1.6)	
1	see SØLN	data (sec 2.6.1.6)	
2	1-NE	I	Modal use descriptor for each eigenvector

Notes:

1. The name of the substructure being reduced (first 2 words of group 0) is the name on a MREDUCE or CREduce command.
2. All eigenvalues found by module READ or CEAD are placed in the LAMS item.
3. The modal use descriptors have the following values
 - 1 - mode was used in modal reduce
 - 2 - mode was selected by NMAX and RANGE parameter but was rejected because of nonparticipation.
 - 3 - mode was excluded because it was outside the specified NMAX and RANGE parameters.

DATA BLOCK DESCRIPTIONS

2.6.1.9 Matrix Items

Description

All matrices stored on the SØF are direct copies of the matrices stored in packed form on GINØ files.

The following matrix items are stored on the SØF:

BMTX	Viscous damping
GIMS	G transformation matrix for boundary matrix in modal reduce
HLFT	Left hand H transformation matrix
HØRG	H or G transformation matrix
KMTX	Stiffness matrix
K4MX	Structure damping matrix
LMTX	Lower triangular factor of stiffness matrix
MMTX	Mass matrix
PAPP	Appended load vectors
PHIL	Left hand eigenvector matrix
PHIS	Eigenvector matrix
PØAP	Load vectors on points omitted during a REDUCE operation
PØVE	Load vectors on points omitted during a REDUCE operation
PVEC	Load vectors
QVEC	Reaction force vectors
UPRT	Partitioning vector from a REDUCE operation
UVEC	Displacement vectors

3. SUBROUTINE DESCRIPTIONS

3.1 INTRODUCTION

Section 3 contains descriptions of subroutines not an integral part of a module. Those subroutines which are an integral part of a module are discussed in section 4, Module Functional Descriptions. Section 3.2 contains an alphabetical index of entry points of routines documented in section 3. A similar index of entry points documented in section 4 can be found in section 4.1.3.

Subroutine descriptions have been partitioned into 3 classifications: executive, utility and matrix subroutine descriptions, documented in sections 3.3, 3.4, and 3.5 respectively.

Descriptions of the plotting utility routines (e.g., AXIS, section 3.4.40; AXISi, section 3.4.41) refer to plotters by number or the letter "i", and to plotter models by number only. The correspondence of these numbers to plotter hardware is given in Table 1. Further details can be found in section 4 of the User's Manual.

SUBROUTINE DESCRIPTIONS

Table 1. Correspondence Between External and Internal Plotter and Model Names and Numbers.

<u>External</u> plotter name	<u>External</u> model name	<u>Internal</u> plotter number	<u>Internal</u> model number
SC	4020,0	3	1
CALCOMP	7651,78i	4	41i
	7651,77j	4	31j
	5651,76k	4	21k
	5651,75ℓ	4	11ℓ
	7655,78i	5	45i
	7655,77j	5	35j
	5655,76k	5	25k
	5655,75ℓ	5	15ℓ
	7631,78i	6	41i
	7631,77j	6	31j
	5631,76k	6	21k
	5631,75ℓ	6	11ℓ
	7635,78i	7	45i
	7635,77j	7	35j
	5635,76k	7	25k
	5635,75ℓ	7	15ℓ
NASTPLT	M,0	10	+1
	T,0	11	+2
	D,0	11	+3
	M,1	10	-1
	T,1	11	-2
	D,1	11	-3

SC = Stromberg Carlson

CALCOMP = California Computing

NASTPLT = NASTRAN General Purpose Plotter

i = Number of spacers (0 or 1), 780 transport

j = Number of spacers (0,1,2 or 3), 770 transport

k = Number of spacers (0,1,2 or 3), 760 transport

ℓ = Number of spacers (0 or 1), 750,470,570 and 580 transports

INTRODUCTION

Table 1. Correspondence Between External and Internal Plotter and Model Names and Numbers
(Cont'd).

SC	=	Stromberg Carlson
CALCOMP	=	California Computing
NASTPLT	=	NASTRAN General Purpose Plotter

ALPHABETICAL INDEX OF ENTRY POINTS FOR SUBROUTINE DESCRIPTIONS

3.2 ALPHABETICAL INDEX OF ENTRY POINTS FOR SUBROUTINE DESCRIPTIONS

<u>Section Number</u>	<u>Entry Point</u>	<u>Subroutine Descriptions</u>	<u>Page Number</u>
3.5.10	ADD	ADD	3.5-15
3.4.1	ANDF	MAPFNS	3.4-1
3.4.40	AXIS	AXIS	3.4-70
3.4.41	AXIS3	AXISi	3.4-72
3.4.41	AXIS10	AXISi	3.4-72
3.4.12	BCKREC	GINØ	3.4-15
3.3.5	BGNSYS	ENDSYS	3.3-6
3.4.88	BISHEL	BISHEL	3.4-142
3.4.87	BISLØC	BISLØC	3.4-140
3.4.74	BISRCH	BISRCH	3.4-123
3.5.1	BLDPK	PAKUNPK	3.5-1
3.5.1	BLDPKI	PAKUNPK	3.5-1
3.5.1	BLDPKN	PAKUNPK	3.5-1
3.3.2	BTSTRP	BTSTRP	3.3-2
3.4.89	BUG	BUG	3.4-144
3.5.5	CALCV	CALCV	3.5-8
3.5.16	CDCØMP	CDCØMP	3.5-74
3.4.12	CLØSE	GINØ	3.4-15
3.4.18	CLSTAB	CLSTAB	3.4-26
3.4.1	CØMPLF	MAPFNS	3.4-1
3.3.12	CØNMSG	CØNMSG	3.3-16
3.4.1	CØRWDS	MAPFNS	3.4-1
3.4.93	CPYFIL	CPYFIL	3.4-149
3.4.92	CPYSTR	CPYSTR	3.4-148
3.4.76	DADØTB	DADØTB	3.4-127
3.4.77	DAXB	DAXB	3.4-128
3.5.15	DDLØØP	DLØØP	3.5-72
3.4.81	DECØDE	DECØDE	3.4-132
3.5.15	DECØMP	DECØMP	3.5-56

SUBROUTINE DESCRIPTIONS

<u>Section Number</u>	<u>Entry Point</u>	<u>Subroutine Descriptions</u>	<u>Page Number</u>
3.4.72	DELSET	DELSET	3.4-121
3.5.5	DLØØP	DLØØP	3.5-72
3.4.84	DMPFIL	DMPFIL	3.4-135
3.5.21	DMPY	DMPY	3.5-83
3.4.68	DRWCHR	DRWCHR	3.5-115
3.4.82	ECTLØC	ECTLØC	3.4-133
3.4.62	EJECT	EJECT	3.4-105
3.5.22	ELIM	ELIM	3.5-85
3.4.12	ENDGET	GINØ	3.4-15
3.4.12	ENDGTB	GINØ	3.4-15
3.4.12	ENDPUT	GINØ	3.4-15
3.3.5	ENDSYS	ENDSYS	3.3-6
3.4.12	EØF	GINØ	3.4-15
3.5.23	FACTØR	FACTØR	3.5-86
3.5.17	FBS	FBS	3.5-76
3.5.17	FBS1,2,3 & 4	FBSI	3.5-77
3.4.12	FILPØS	GINØ	3.4-15
3.5.12	FILSWI	FILSWI	3.5-29
3.5.15	FINDC	FINDC	3.5-69
3.5.15	FINWRT	ØNETWØ	3.5-66
3.4.17	FNAME	FNAME	3.4-25
3.4.69	FNDPLT	FNDPLT	3.4-117
3.4.75	FØRFIL	FØRFIL	3.4-126
3.4.15	FREAD	FREAD	3.4-23
3.4.12	FWDREC	GINØ	3.4-15
3.5.15	GENVEC	GENVEC	3.5-66
3.4.12	GETSTB	GINØ	3.4-15
3.4.12	GETSTR	GINØ	3.4-15
3.4.12	GETURN	GINØ	3.4-15
3.5.19	GFBS	GFBS	3.5-79

ALPHABETICAL INDEX OF ENTRY POINTS FOR SUBROUTINE DESCRIPTIONS

<u>Section Number</u>	<u>Entry Point</u>	<u>Subroutine Descriptions</u>	<u>Page Number</u>
3.4.12	GINØ	GINØ	3.4-15
3.4.32	GMMATD	GMMATD	3.4-49
3.4.33	GMMATS	GMMATS	3.4-52
3.3.4	GNFIAT	GNFIAT	3.3-5
3.3.9	GNFIST	GNFIST	3.3-12
3.4.14	GØPEN	GØPEN	3.4-22
3.4.80	HBDY	HBDY	3.4-131
3.4.71	HEAD	HEAD	3.4-120
3.4.73	HMAT	HMAT	3.4-122
3.4.44	IDPLØT	IDPLØT	3.4-75
3.4.45	INTGPT	INTGPX	3.4-76
3.4.45	INTGPX	INTGPX	3.4-76
3.4.46	INTLST	INTLST	3.4-77
3.5.1	INTPK	PAKUNPK	3.5-1
3.5.1	INTPKI	PAKUNPK	3.5-1
3.4.34	INVERD	INVERD	3.4-53
3.4.35	INVERS	INVERS	3.4-54
3.4.1	KØRSZ	MAPFNS	3.4-1
3.4.47	LINE	LINE	3.4-78
3.4.48	LINE1	LINEi	3.4-79
3.4.48	LINE2	LINEi	3.4-79
3.4.48	LINE3	LINEi	3.4-79
3.4.48	LINE4	LINEi	3.4-79
3.4.48	LINE9	LINEi	3.4-79
3.4.48	LINE10	LINEi	3.4-79
3.4.30	LØCATE	PRELØC	3.4-44
3.4.1	LSHIFT	MAPFNS	3.4-1
3.4.86	LSPLIN	LSPLIN	3.4-138
3.5.30	MAKMCB	MAKMCB	3.5-93
3.4.36	MAT	PREMAT	3.4-55
3.4.28	MATDUM	MATDUM	3.4-42

SUBROUTINE DESCRIPTIONS

<u>Section Number</u>	<u>Entry Point</u>	<u>Subroutine Descriptions</u>	<u>Page Number</u>
3.5.6	MERGE	PARTN - MERGE	3.5-9
3.4.25	MESSAGE	MESSAGE	3.4-39
3.5.9	MPART	UPART	3.5-14
3.5.12	MPYAD	MPYAD	3.5-18
3.5.12	MPY1	MPYQ	3.5-29
3.5.12	MPY2NT	MPYQ	3.5-29
3.5.12	MPY2T	MPYQ	3.5-29
3.5.12	MPY3T	MPYQ	3.5-29
3.4.83	MRGE	MRGE	3.4-134
3.4.26	MSGWRT	MSGWRT	3.4-40
3.3.16	NASCAR	NASCAR	3.3-21
3.4.10	NASTIØ	NASTIØ	3.4-12
3.5.15	ØNETWØ	ØNETWØ	3.5-66
3.4.11	ØPEN	ØPEN	3.4-14
3.4.1	ØRF	MAPFNS	3.4-1
3.5.1	PACK	PAKUNPK	3.5-1
3.4.24	PAGE	PAGE	3.4-38
3.4.24	PAGE1	PAGE	3.4-38
3.4.24	PAGE2	PAGE	3.4-38
3.5.6	PARTN	PARTN - MERGE	3.5-9
3.4.22	PEXIT	PEXIT	3.4-36
3.4.70	PHDMIA	PHDMIA	3.4-118
3.4.70	PHDMIB	PHDMIA	3.4-118
3.4.70	PHDMIC	PHDMIA	3.4-118
3.4.70	PHDMID	PHDMIA	3.4-118
3.4.63	PLAMAT	PLAMAT	3.4-106
3.4.67	PLTSET	PLTSET	3.4-113
3.4.30	PRELØC	PRELØC	3.4-44
3.4.36	PREMAT	PREMAT	3.4-55
3.4.39	PRETAB	PRETAB	3.4-67
3.4.37	PRETRD	PRETRD	3.4-64
3.4.38	PRETRS	PRETRS	3.4-66

ALPHABETICAL INDEX OF ENTRY POINTS FOR SUBROUTINE DESCRIPTIONS

<u>Section Number</u>	<u>Entry Points</u>	<u>Subroutine Descriptions</u>	<u>Page Number</u>
3.4.49	PRINT	PRINT	3.4-81
3.4.12	PUTSTR	GINØ	3.4-15
3.4.12	QØPEN	GINØ	3.4-15
3.4.20	RCARD	RCARD	3.4-32
3.5.15	RCØRE	TIMEEQ	3.5-70
3.4.50	RDMØDE	RDMØDX	3.4-83
3.4.50	RDMØDX	RDMØDX	3.4-83
3.4.50	RDMØDY	RDMØDX	3.4-83
3.4.16	RDTRL	WRTTRL	3.4-24
3.4.50	RDWORD	RDMØDX	3.4-83
3.4.12	READ	GINØ	3.4-15
3.4.12	RECTYP	GINØ	3.4-15
3.3.15	RETURN	RETURN	3.3-20
3.4.12	REWIND	GINØ	3.4-15
3.4.91	RE2AL	RE2AL	3.4-146
3.4.1	RSHIFT	MAPFNS	3.4-1
3.5.6	RULER	PARTN - MERGE	3.5-10
3.5.26	SADD	SADD	3.5-90
3.4.78	SADØTB	SADØTB	3.4-129
3.4.12	SAVPØS	GINØ	3.4-15
3.4.79	SAXB	SAXB	3.4-130
3.4.51	SCLØSE	SGINØ	3.4-85
3.5.14	SDCIN	SDCIN	3.4-53
3.5.14	SDCØM1	SDCØM	3.5-53
3.5.14	SDCØM2	SDCØM	3.5-53
3.5.14	SDCØM3	SDCØM	3.5-53
3.5.14	SDCØM4	SDCØM	3.5-53
3.5.14	SDCØMP	SDCØMP	3.5-32
3.5.14	SDCØUT	SDCØUT	3.5-53
3.5.8	SDR1B	SDR1B	3.5-13

SUBROUTINE DESCRIPTIONS

<u>Section Number</u>	<u>Entry Point</u>	<u>Subroutine Descriptions</u>	<u>Page Number</u>
3.3.6	SEARCH	SEARCH	3.3-8
3.4.43	SELCAM	SELCAM	3.4-74
3.3.3	SEMINT	SEMINT	3.3-3
3.3.14	SEMTRN	SEMTRN	3.3-19
3.4.51	SEØF	SGINØ	3.4-85
3.4.94	SETFND	SETFND	3.4.150
3.4.12	SKPFIL	GINØ	3.4-15
3.4.42	SKPFRM	SKPFRM	3.4-73
3.4.90	SKPREC	SKPREC	3.4-145
3.5.20	SØLVER	SØLVER	3.5.81
3.4.51	SØPEN	SGINØ	3.4-85
3.4.31	SØRT	SØRT	3.4-46
3.5.7	SSG2A	SSG2A	3.5-12
3.5.13	SSG2B	SSG2B	3.5-31
3.5.11	SSG2C	SSG2C	3.5-16
3.5.18	SSG3A	SSG3A	3.5-78
3.4.85	SSPLIN	SSPLIN	3.4-136
3.3.11	SSWTCH	SSWTCH	3.3-15
3.4.52	STPLØT	STPLØT	3.4-87
3.4.51	SWRITE	SGINØ	3.4-85
3.4.53	SYMBØL	SYMBØL	3.4-88
3.5.15	T	TIMEEQ	3.5-70
3.4.39	TAB	PRETAB	3.4-67
3.4.29	TABPRT	TABPRT	3.4-43
3.4.21	TAPBIT	TAPBIT	3.4-35
3.5.15	TFIN	TIMEEQ	3.5-70
3.4.54	TIPE	TIPE	3.4-90
3.4.23	TMTØGØ	TMTØGØ	3.4-37
3.5.24	TRANP1	TRANP1	3.5-87
3.4.37	TRANSD	PRETRD	3.4-64

ALPHABETICAL INDEX OF ENTRY POINTS FOR SUBROUTINE DESCRIPTIONS

<u>Section Number</u>	<u>Entry Point</u>	<u>Subroutine Descriptions</u>	<u>Page Number</u>
3.5.15	TRANSP	TRANSP	3.5-65
3.4.38	TRANSS	PRETRS	3.4-66
3.5.25	TRNSP	TRNSP	3.5-88
3.3.13	TTLPGE	TTLPGE	3.3-17
3.4.55	TYPE1	TYPEi	3.4-92
3.4.55	TYPE2	TYPEi	3.4-92
3.4.55	TYPE3	TYPEi	3.4-92
3.4.55	TYPE9	TYPEi	3.4-92
3.4.55	TYPE10	TYPEi	3.4-92
3.4.56	TYPFLT	TYPFLT	3.4-94
3.4.57	TYPINT	TYPINT	3.4-96
3.5.1	UNPACK	PAKUNPK	3.5-1
3.5.9	UPART	UPART	3.5-14
3.4.27	USRMSG	USRMSG	3.4-41
3.4.58	WPLT1	WPLT1	3.4-98
3.4.59	WPLT2	WPLT2	3.4-100
3.4.60	WPLT3	WPLT3	3.4-102
3.4.64	WPLT4	WPLT4	3.4-108
3.4.65	WPLT9	WPLT9	3.4-110
3.4.66	WPLT10	WPLT10	3.4-111
3.4.12	WRITE	GINØ	3.4-15
3.4.16	WRTTRL	WRTTRL	3.4-24
3.3.10	XEØT	XEØT	3.3-14
3.5.15	XLØØP	DLØØP	3.5-72
3.4.1	XØRF	MAPFNS	3.4-1
3.4.19	XRCARD	XRCARD	3.4-27
3.3.8	XSEMXX	XSEMXX	3.3-11
3.3.1	XSEM1	XSEM01	3.3-1
3.3.7	XSEM2	XSEM02	3.3-9
3.3.7	XSEM3	XSEM03	3.3-9

SUBROUTINE DESCRIPTIONS

<u>Section Number</u>	<u>Entry Point</u>	<u>Subroutine Descriptions</u>	<u>Page Number</u>
3.3.7	XSEM4	XSEM04	3.3-9
3.3.7	XSEM5	XSEM05	3.3-9
3.3.7	XSEM6	XSEM06	3.3-9
3.3.7	XSEM7	XSEM07	3.3-9
3.3.7	XSEM8	XSEM08	3.3-9
3.3.7	XSEM9	XSEM09	3.3-9
3.3.7	XSEM10	XSEM10	3.3-9
3.3.7	XSEM11	XSEM11	3.3-9
3.3.7	XSEM12	XSEM12	3.3-9
3.3.7	XSEM13	XSEM13	3.3-9
3.3.7	XSEM14	XSEM14	3.3-9
3.5.1	ZBLPKI	PAKUNPK	3.5-1
3.5.1	ZNTPKI	PAKUNPK	3.5-1

EXECUTIVE SUBROUTINE DESCRIPTIONS

3.3 EXECUTIVE SUBROUTINE DESCRIPTIONS.

3.3.1 XSEM1 (Executive Sequence Monitor, Preface).

3.3.1.1 Entry Point: XSEM1.

3.3.1.2 Purpose

To initiate the execution of the NASTRAN Preface.

3.3.1.3 Calling Sequence

CALL XSEM1

3.3.1.4 Method

Subroutine BTSTRP is called to initialize machine dependent data, and then subroutine SEMINT is called to execute the program Preface (i.e. input file processors and DMAP program compiler). After initiating the problem, modules are called as directed by the ØSCAR until a module is encountered in the ØSCAR that does not reside in link 1 at which time XSEM1 calls subroutine ENDSYS to load a new link.

3.3.1.5 Design Requirements

XSEM1 must reside in the core resident portion of link 1. Link 1 is not re-entrant which means that once the program leaves link 1 it can never transfer control back to link 1. Functional DMAP modules can not reside in link 1. Open core is used for a GINØ buffer with named common block /ESFA/ defining the beginning of open core. See the second paragraph of the design requirements section of the subroutine description XSEMi (see section 3.3.7) for details on files, data blocks, and common blocks necessary for operation.

3.3.2 BTSTRP (Bootstrap Generator).

3.3.2.1 Entry Point: BTSTRP.

3.3.2.2 Purpose

Determines the machine type and initializes the machine dependent constants and masks within the NASTRAN system block data program (SEMDBD).

3.3.2.3 Calling Sequence

CALL BTSTRP

3.3.2.4 Method

The machine type (IBM 7094, IBM S/360, Univac 1108, CDC 6600) is determined by inspection of the machine binary word length and the known methods of representing negative integers (sign and magnitude or ones/twos complement) using the following algorithm:

1. If the ones complement (CØMPLF see section 3.4.1) of -1 is greater than 2, the machine is the IBM 7094. If not, the machine is an IBM S/360, Univac 1108 or CDC 6600. (i.e., only the sign and magnitude representation of -1 on the 7094 will yield a large (> 2) positive value when complemented.)
2. Shift (RSHIFT see section 3.4.1) a binary machine word of all 1's to the right thirty-two binary places. Compare the resulting value to 15. If the value is less than fifteen, the machine is the 32 bit IBM S/360; equal to fifteen, the 36 bit Univac 1108; and greater than fifteen, the 60 bit CDC 6600.

Once the machine type is known, the proper constants and masks are selected from assembled tables.

3.3.2.5 Design Requirements

This subroutine must be modified if it is to operate with other than the four machine types listed above.

EXECUTIVE SUBROUTINE DESCRIPTIONS

3.3.3 SEMINT (Sequence Monitor Initialization).

3.3.3.1 Entry Point: SEMINT.

3.3.3.2 Purpose

To execute the Preface of a NASTRAN problem solution.

3.3.3.3 Calling Sequence

CALL SEMINT

3.3.3.4 Method

The first card of the NASTRAN data deck is read from the system input file and its image stored in blank COMMON. XRCARD is called to convert the card. If the name of the card is NASTRAN, the card is echoed and keywords are identified and appropriate words of /SYSTEM/ are reset to the input values. If an unidentified keyword is detected, or the card has a format error, a message is printed and the NØGGØ flag is turned on. The first word of blank COMMON is set to one if the card was a NASTRAN card, to zero otherwise. Then GNFIAT is called to generate the initial FIAT. XCSA is called to read and process the Executive Control Deck. IFP1 is called to read and process the Case Control Deck. XSØRT is called to read and sort the Bulk Data Deck. If bulk data is present, IFP is called to process it. If the problem is a conical shell problem, IFP3 is called to further process the bulk data. If the current run is to prepare a User's Master File, UMFEDT is called and control is returned to XSØRT for each new problem to be written on the UMF. Otherwise, XGPI is called to perform General Problem Initialization and then return is made to XSEMI signifying completion of the Preface.

3.3.3.5 Design Requirements

If the NASTRAN card is present, it must be the first card of the data deck. For a description of the keywords, see the NASTRAN User's Manual, Section 2.1.

3.3.3.6 Diagnostic Messages

UNIDENTIFIED NASTRAN KEYWORD _____. ACCEPTABLE KEYWORDS FOLLOW--

BUFSIZE	SYSTEM
CØNFIG	FILE
MAXFILES	FILES
MAXØPEN	

SUBROUTINE DESCRIPTIONS

Self-explanatory.

NASTRAN CARD DOES NOT HAVE CORRECT FORMAT.

Typical errors include non-integer values or continuation of the card following an = sign.

See section 6.3.1 for further details on the NASTRAN card.

EXECUTIVE SUBROUTINE DESCRIPTIONS

3.3.4 GNFIAT (Generate FIAT).

3.3.4.1 Entry Point: GNFIAT.

3.3.4.2 Purpose

Determines the number of logical files available within the computer hardware and software configuration and places an entry for each into FIAT or XFIAT.

3.3.4.3 Calling Sequence

CALL GNFIAT

GNFIAT must be called once and only once as the first call from the preface.

3.3.4.4 Method

Each computer configuration has its own independent subroutine to accomplish the necessary functions of GNFIAT. The subroutine interrogates unit blocks, data definition cards, file tables, etc. to determine the number of logical files available within the configuration. As each logical file is sensed, it is determined whether the file has been assigned to a physical magnetic tape or some bulk storage device such as disk or drum. Each file has a logical name and/or number for identification. These file ID's are stored in FIAT, XFIAT or both depending on several factors. As the file ID is stored, a physical tape flag is set where appropriate. The factors that determine FIAT vs. XFIAT storage are as follows: 1) the first PFIST (see section 2.4 for a description of the FIST) files sensed are always entered into XFIAT, 2) except for the first file (always the PØØL), all of the first PFIST files without tape flags are also entered into FIAT, and 3) all other files are entered into FIAT only.

3.3.4.5 Design Requirements

Since GNFIAT routines are computer hardware/software dependent, operational requirements may differ at various times. See appropriate commented assembly listing if difficulties or error codes are encountered.

SUBROUTINE DESCRIPTIONS

3.3.5 ENDSYS (End-of-Link).

3.3.5.1 Entry Points: ENDSYS, BGNSYS.

3.3.5.2 Purpose

For ENDSYS, to save various NASTRAN core-resident Executive Tables on a scratch file for use in communicating with the next link requested.

For BGNSYS, to restore the NASTRAN Executive Tables saved by ENDSYS and to position the ØSCAR at the correct entry to be executed in the resident link.

3.3.5.3 Calling Sequences

CALL ENDSYS(LINK,X,REWFLG)

LINK - BCD name of the link. The naming convention is: NS01 = link 1, NS02 = link 2, etc.

X - Dependent on machine type. For the IBM 7094 only, X (6 BCD characters) specifies the unit where the links are stored. Not used on other machines.

REWFLG = 0 indicates LINK is ahead of current link (i.e. we are going from link N to link N + K, K > 0). IBM 7094 only; not used on other machines.

REWFLG = 1 indicates LINK is behind current link (i.e. we are going from link N to link N + K, K < 0). IBM 7094 only; not used on other machines.

CALL BGNSYS.

3.3.5.4 Method

For ENDSYS, a search is made for an empty file and when found the Executive Tables are written (saved) on it. A pointer to the save file is saved in blank common or written on a system file for use by BGNSYS when the new link is loaded. Subroutine SEARCH is then called to load the requested link.

BGNSYS is called after a new link is loaded. The pointer to the save file containing the Executive Tables is obtained from either blank common or a system file, and the Executive Tables are reloaded into core. The ØSCAR is positioned at the correct entry to be executed, and a RETURN is made to the calling routine.

EXECUTIVE SUBROUTINE DESCRIPTIONS

3.3.5.5 Design Requirements

Program links are usually considered to be physically separate programs, essentially independent of one another except for the fact that the operating system executive (not the NASTRAN executive) provides a means by which control can be transferred from one link to another when requested by the user. The means by which the operating system executive transfers control from one link to another is dependent upon the machine and the system being used. For some future machines there may be no means for building physically separate links so the links become logical subsets of one huge program.

No matter how the links are formed it is necessary, when transferring from one link to another, that all file assignments be preserved as well as their status (i.e. don't rewind the tapes).

Open core is used for GINØ buffer area and the beginning of open core is defined by named common block /ESFA/.

If no save file is available or if any of the Executive Tables to be written exceeds 900 words, the job is terminated.

SUBROUTINE DESCRIPTIONS

3.3.6 SEARCH (Search, Load, and Execute Link).

3.3.6.1 Entry Point: SEARCH.

3.3.6.2 Purpose

SEARCH locates (searches for) a particular link of the NASTRAN system on the Link Storage File, loads the link into the computer memory and transfers execution control to the link entry point XSEMi, i = 2,3,....

3.3.6.3 Calling Sequence

CALL SEARCH(LKNAM,LKFIL,REW)

LKNAM = 4 character symbolic name of link, i.e., NS01, NS02 for link 1, link 2, etc.

LKFIL = symbolic name of the Link Storage File (IBM 7094 only)

REW = set non-zero to position a sequential Link Storage File to its beginning
(IBM 7094 only)

3.3.6.4 Method

SEARCH is machine dependent. It interfaces with the machine operating system to provide a multi-link capability. Each link is a somewhat arbitrary part of the complete NASTRAN system. The division into links was necessary only because of the size limitation for program complexes imposed by the various operating systems. The linking technique for each machine is discussed in section 5 of the Programmer's Manual.

3.3.6.5 Design Requirements

Only the IBM 7094 system requires the Link Storage File to be named (LKFIL) and, since it is sequential, provides the capability of rewinding it following a SEARCH call. All other systems provide random access (disk, drum) Link Storage Files.

3.3.6.6 Diagnostic Messages

Individual SEARCH subroutines may abnormally terminate due to hardware malfunction. See appropriate commented assembly listing if difficulties or error codes are encountered.

EXECUTIVE SUBROUTINE DESCRIPTIONS

3.3.7 XSEMi (Link i Main Program, i = 2,3, ...)

3.3.7.1 Entry Point: XSEMi.

3.3.7.2 Purpose

To get the next module to be executed from the ØSCAR, initialize the module and call it if it is in link i, or transfer to the link in which module resides if it is not in link i.

3.3.7.3 Calling Sequence

Example: CALL XSEM2, where XSEM2 is the entry point of link 2

3.3.7.4 Method

Subroutine BTSTRP is called to initialize machine dependent data, and then subroutine BGNSYS is called to reload Executive Tables saved from the previous link.

The next ØSCAR entry is read into core and processed. If the entry is for a functional module, subroutine GNFISt is called to link files with input, output and scratch data blocks needed by the module. Variable parameter values needed by the module are transferred to blank common from table VPS which resides in named common block /XVPS/. Constant values in the ØSCAR entry parameter section are transferred to blank common.

The link specification table in named common block /XLINK/ is examined to see if the module resides in this link. If it does, the module is called. Upon returning from the module, the diagnostic message queue is checked and message writer MSGWRT is called if there are messages queued. Begin and end execution times are printed out for functional modules.

The next ØSCAR entry is read and the process is repeated until a module is encountered which does not reside in this link, at which time subroutine ENDSYS is called to initiate loading of the link containing the module.

3.3.7.5 Design Requirements

XSEMi must reside in the core resident portion of link i. Link i is re-entrant which means program control can be transferred to this link as often as needed. Open core is used for a GINØ buffer with named common block /ESFA/ defining the beginning of open core. An ØSCAR entry cannot be greater than 200 words.

SUBROUTINE DESCRIPTIONS

Files, data blocks and named common blocks needed by XSEMI are listed below, along with type of access required (i.e. fetch and/or store data) and reasons for use.

1. Data Pool File - fetch. Contains XØSCAR data block.
2. XØSCAR - fetch. Contains ØSCAR entry to be processed.
3. Common /XLINK/ - fetch. Contains link specification table.
4. Common /XFIST/ - store. Initialized prior to calling GNFIST.
5. Common /XPFIST/ - fetch. Contains parameter needed to initialize FIST table.
6. Common /ØSCENT/ - fetch. Contains ØSCAR entry to be processed.
7. Common /ESFA/ - store. Defines beginning of open core area used by GINØ buffer.
8. Common /XVPS/ - fetch. Contains variable parameter values needed to initialize module to be executed.
9. Common /MSGX/ - fetch. Contains diagnostic message queue.
10. Common /SEM/ - fetch. Contains BCD names of links NS01, NS02,

3.3.7.6 Diagnostic Messages

A message is written if the module to be executed required more files than are available. The job is then terminated.

EXECUTIVE SUBROUTINE DESCRIPTIONS

3.3.8 XSEMXX (Sequence Monitor - Deck Generator).

3.3.8.1 Entry Point: XSEMXX.

3.3.8.2 Purpose

To provide a model from which all other XSEMi (i = link number) subroutines except XSEM1 can be made.

See section 6.11, which discusses how to generate a link driver subroutine.

SUBROUTINE DESCRIPTIONS

3.3.9 GNFIST (Generate FIST)

3.3.9.1 Entry Point: GNFIST.

3.3.9.2 Purpose

To set up the proper linkage between data blocks and the files they reside on in preparation for executing the functional module requiring the data blocks.

3.3.9.3 Calling Sequence

CALL GNFIST(DBN,FISTNM,MØDNØ)

DBN - Data block name (Two word BCD array - 8 characters total)

FISTNM - Data block identification (GINØ file number) used by functional module (integer).

MØDNØ - ØSCAR record number of functional module to be executed (integer). MØDNØ indicates to the calling routine what action was taken by GNFIST.

MØDNØ > 0, data block assigned a file or it was purged.

MØDNØ = 0, fatal error detected.

MØDNØ < 0, data block not assigned a file, GNFIST called Executive Segment File Allocator (XSFA)

3.3.9.4 Method

If the data block is purged, GNFIST returns to the calling routine with MØDNØ > 0. A data block is purged if it is an input which has not been generated or its status is purged or DBN = 0.

If an input data block resides on the Data Pool File and needs to be unpooled, GNFIST calls the file allocator (XSFA) to unpool all inputs to the module which reside on the Data Pool File that need to be unpooled. GNFIST then returns to the calling routine with MØDNØ < 0. The other condition under which XSFA is called is if a file has not been allocated to a non-purged output data block or scratch data block needed by the module.

A file is allocated to a data block when the data block name appears in the FIAT table, located in named common block /XFIAT/, as unpurged. Input, output and scratch data blocks which have been assigned to a file and are required by the functional module, have their FISTNM's entered

EXECUTIVE SUBROUTINE DESCRIPTIONS

in the FIST table which is located in named common block /XFIST/. FIST entries are linked to the DBN's in the FIAT table which in turn links the data block to a file. This completes the linking of functional module data blocks to their files.

Output data blocks cannot reside on the Data Pool File, so GNFIST checks for this and if found, the DBN and all DBN's equivalenced to it are deleted from the DPL table located in named common /XDPL/.

3.3.9.5 Design Requirements

GNFIST resides in the core resident portion of a link. It does not use open core and the only restriction is that the FIST table be large enough to hold all FISTNM's for a module.

The named common blocks needed by GNFIST are listed below, along with type of access required (i.e. fetch and/or store data) and reasons for use.

1. COMMON/XFIST/ - Store.

Used to store FISTNM's and link FISTNM's with their corresponding DBN's in FIAT.

2. COMMON/XFIAT/ - Fetch and store.

Used to determine status of DBN's. The FIAT table is altered if unpooling of input data blocks is necessary.

3. COMMON/XDPL/ - Fetch and store.

Used to determine status of input DBN's and is altered if output DBN's appear in it.

4. COMMON/ØSCENT/ - Fetch.

Contains ØSCAR entry for functional module to be executed. Used to alter FIAT when input DBN's need to be unpooled.

3.3.9.6 Diagnostic Messages

GNFIST detects overflow in FIST table and sends message to terminate job.

SUBROUTINE DESCRIPTIONS

3.3.10 XEØT (End-of-Tape).

3.3.10.1 Entry Point: XEØT.

3.3.10.2 Purpose

To prepare and send to the computer operator, messages instructing him what to do when end-of-tape has been encountered on the Old Problem Tape (ØPTP) or the New Problem Tape (NPTP).

3.3.10.3 Calling Sequence

CALL XEØT(ID,ØREEL,NREEL,BUF)

ID - BCD name NPTP or ØPTP

ØREEL - Number of reel to be dismounted - integer.

NREEL - Number of new reel to be mounted - integer.

BUF - GINØ buffer used by NPTP or ØPTP.

3.3.10.4 Method

A check is made to see if tape has multi-reel capability. If not, a fatal message is issued and job is terminated. The operator messages are generated and issued and the old reel is re-wound and unloaded. A check is made to see if correct new reel has been mounted and then a return is made to calling program.

3.3.10.5 Design Requirements

XEØT must be accessible to routines XGPI and XCHK.

3.3.10.6 Diagnostic Messages

A message is issued if tape does not have multi-reel capability.

EXECUTIVE SUBROUTINE DESCRIPTIONS

3.3.11 SSWTCH (Sense Switches)

3.3.11.1 Entry Point: SSWTCH.

3.3.11.2 Purpose

To indicate to the calling routine whether or not a specified sense switch is set.

3.3.11.3 Calling Sequence

CALL SSWTCH(SS,F)

SS - Sense switch number - integer. $1 \leq SS \leq 31$.

F - Flag indicating status of SS

F = 0 if SS not on

F = 1 if SS is on

3.3.11.4 Method

Named common block /SYSTEM/ contains the word which contains the sense switch settings.

Bit 1 of the word corresponds to sense switch 1, bit 2 corresponds to sense switch 2, etc. If the bit corresponding to SS is on then F = 1, if not then F = 0.

Note that sense switches are set by the user via the DIAG card in the Executive Control Deck and not through physical sense switches set by the computer operator.

The following sense switches are currently in use:

Switch

- | | |
|---|---|
| 1 | Dump core when subroutine DUMP or PDUMP is called. This will cause a core dump on any nonpreface fatal error. |
| 2 | Print the FIAT after each call to XSFA. |
| 3 | Print the Data Pool Dictionary after each call to XSFA. |
| 4 | Print the ØSCAR at the end of XGPI. |
| 5 | Type a message to signify the beginning of each module on the operator's console. |
| 6 | Type a message to signify the ending of each module on the operator's console. |
| 7 | Print eigenvalue extraction diagnostics for real inverse power and real and complex determinant methods. |

SUBROUTINE DESCRIPTIONS

Switch

8	Print matrix trailers as the matrices are generated and data block trailers as data blocks are generated.
9	Not used.
10	Use alternate nonlinear loading in TRD. (Replace $\{N_{n+1}\}$ by $\frac{1}{3} \{N_{n+1} + N_n + N_{n-1}\}$)
11	Print all active row and column possibilities for the decomposition algorithm.
12	Print eigenvalue extraction diagnostics for complex inverse power.
13	Print open core length.
14	Print the Rigid Format (NASTRAN SOURCE PROGRAM COMPILATION) for all non-Restart runs.
15	Trace GINØ ØPEN/CLØSE operations on CDC 6000 series.
16	Trace real inverse power eigenvalue extraction operations.
17	Punch the DMAP sequence that is compiled.
18	Print internal grid points picked by SET2 cards.
19	Print data for MPYAD method selection.
20	Generate de-bug printout (For NASTRAN programmers who include CALL BUG in their subroutines).
21	Print GP4 set definition.
22	Print GP4 degree of freedom definition.
23	Print DMAP alters generated during multi-stage substructuring.
24	Punches DMAP alters generated during multi-stage substructuring.
25-26	Not used.
27	Input File Processor (IFP) table dump.
28	Punch out the link specification table - deck XBSBD.
29	Process link specification table update deck.
30	Punch out alters to XSEMI decks.
31	Print link specification table.
32-38	Not used.
39	FA1 additional printout (trace operations)

For a further explanation of switches 28-31 see Section 6.11 in the Programmer's Manual.

EXECUTIVE SUBROUTINE DESCRIPTIONS

3.3.11.5 Design Requirements

SSWTCH resides in the core resident portion of a link.

SUBROUTINE DESCRIPTIONS

3.3.12 CØNMSG (Console Message Writer).

3.3.12.1 Entry Point: CØNMSG

3.3.12.2 Purpose

Writes the current time and a NASTRAN system message onto the system output device and (if the computer configuration permits) onto the on-line operator's console device.

3.3.12.3 Calling Sequence

CALL CØNMSG(MSG,CNT,YN)

MSG - Array name containing message.

CNT - Number of 4-character words in message (integer).

YN - 1 = yes, 0 = no. Print on-line device if yes and available.

3.3.12.4 Method

A computer real-time and/or job clock is interrogated. The number of message words indicated is sent to the system output device (usually printer) along with the clock reading(s). If the computer configuration permits and the yes/no switch is set yes, the same clock reading(s) and message is sent to the operator's console device (usually typewriter).

3.3.12.5 Design Requirements

Only the left-most four characters from each computer word are extracted and sent to the output device(s).

EXECUTIVE SUBROUTINE DESCRIPTIONS

3.3.13 TTLPGE (Title Page Writer)

3.3.13.1 Entry Point: TTLPGE

3.3.13.2 Purpose

To print on the system output file title page information as follows:

- the NASTRAN symbol
- the machine type and model
- the system generation date
- the level identification
 - major level number (corresponds to the basic archive Source Library)
 - minor level number (corresponds to the object library for a machine type)
 - local level number
 - variations of a local level
- the Rigid Format series identification, including modifications, if any.

3.3.13.3 Calling Sequence

CALL TTLPGE (K)

3.3.13.4 Method

TTLPGE is called from NASCAR following reading of the NASTRAN card (if any).

The variable K is stored as a local variable in subroutine NASCAR and may be set at execution time by the user on the NASTRAN data card by

NASTRAN TITLEOPT = k

where the default value for k is 1 as defined by a DATA statement in subroutine NASCAR. The action taken by TTLPGE depends on the integer option parameter K (whose value is k) as shown below.

<u>k</u>	<u>TTLPGE action</u>
<0	Print one (1) copy of an abbreviated title page
=0	Supress any title page printout
1	Print one (1) copy of the full title page
2	Print two (2) copies of the full title page
3	Print a single copy of a locally annotated title page
>3	Supress any title page printout

SUBROUTINE DESCRIPTIONS

Whenever changes are incorporated into NASTRAN, the TTLPGE routine should be updated to reflect these changes. This is particularly important when official updates are made since runs may only be identifiable by the information contained in this printout. The basic identification of a given version of NASTRAN is called the Level number, a code of the form

i.j.k

where i is the current major level number, j is the minor level number and k is local level number.

The major level corresponds to a complete recompilation of the entire system on each machine from a single archive source library maintained for all machines. It is through this mechanism that the machine-independent nature of the NASTRAN code is guaranteed. Major levels of NASTRAN will probably only be issued at intervals of once a year or longer due to the expense involved.

Minor levels correspond to changes that are made on one given type of machine, say the CDC 6000 machines. These changes are reflected in the object library for the given machine class, and may be reflected in the source by either alters to the basic source library or by an updated source library. Minor levels will probably be issued every few months for each machine class as alters to the basic or previous source library.

Local levels are reserved for locally made changes and provides a mechanism for the local NASTRAN system programmer to keep track of several versions of NASTRAN that may exist at his installation. This would probably consist of a digit or a digit and a typed letter (e.g., Level 15.1.2A).

The Rigid Format series is designated by a letter. Minor modifications will be identified by a digit (e.g., Rigid Format Series M.2). It is anticipated that new series of Rigid Formats will only be available concurrently with major levels of the program.

EXECUTIVE SUBROUTINE DESCRIPTIONS

3.3.14 SEMTRN (Transliterater) (IBM 360-370 only)

3.3.14.1 Entry Point: SEMTRN

3.3.14.2 Purpose

To read the system input stream and convert EBCDIC characters to BCD.

3.3.14.3 Calling Sequence

CALL SEMTRN (KIN, KØUT)

3.3.14.4 Method

An I/O activity is done using FØRTRAN. One eighty (80) column card image at a time is read from FØRTRAN unit KIN, transliterated, and written out on FØRTRAN unit KØUT. FØRTRAN unit KØUT is rewound before writing and before returning. FØRTRAN unit KIN is not rewound before reading and is not manipulated further once an end-of-file condition is detected. Any EBCDIC characters other than the standard NASTRAN set defined in Section 2.1 of the User's Manual are transliterated to the blank character. BCD punched characters are transliterated into themselves. Thus, for the standard character set, either BCD, EBCDIC or mixed BCD and EBCDIC may be used on the IBM 360-370 computer systems. It should be emphasized that decks containing EBCDIC characters will not run on the other NASTRAN computers.

3.3.14.5 Design Requirements

The FØRTRAN unit KØUT must be defined in the JCL and sufficient space must be allocated to hold the transliterated input stream. The actual unit numbers used are defined by the calling program (SEMINT) and are currently set to KIN = 5 and KØUT = 1. If the 2314 disk facility is used for KØUT, the space can be estimated by

$$\text{No. Tracks} = \frac{\text{No. Cards}}{91}$$

if full track blocking is used. This is accomplished by specifying the DCB as

DCB = (RECFM=FB, LRECL=80, BLKSIZE=7280) .

The transliteration is effected by using the character read in as an index into a 256 byte table containing the desired BCD representations. In this way, no look-up expense is involved.

SUBROUTINE DESCRIPTIONS

3.3.15 RETURN (Return)

3.3.15.1 Entry Point: RETURN

3.3.15.2 Purpose:

To allow inclusion of calls to non-existing decks. Linkage Editor data changes are required to use this capability.

3.3.15.3 Calling Sequence

CALL RETURN

3.3.15.4 Method

The only executable statement is a RETURN to the calling program.

3.3.15.5 Design Requirements

RETURN should be located in LINK 0 or in the root segment.

3.3.15.6 Diagnostic Messages

None.

3.3.16 NASCAR (Read the NASTRAN card)

3.3.16.1 Entry Point: NASCAR

3.3.16.2 Purpose

To read the NASTRAN card (if any) and then to call TTLPGE (see Section 3.3.13).

3.3.16.3 Calling Sequence

CALL NASCAR

3.3.16.4 Method

NASCAR is called as the first executable statement in the Preface driver SEMINT following transliteration (IBM 360,370 only). NASCAR reads the first data card from the input stream. If it is a NASTRAN card, NASCAR decodes its contents and sets the values thereby defined. Following this, NASCAR calls TTLPGE and returns to SEMINT.

3.3.16.5 Design Requirements

NASCAR is intended to be overlayed, along with TTLPGE, under both SEMINT and XRCARD. Any common blocks having entries which can be overridden via the NASTRAN card must be accessible to the routine.

3.3.16.6 Diagnostic Messages

Messages 17 and 43 may be issued.

3.4 UTILITY SUBROUTINE DESCRIPTIONS.

3.4.1 MAPFNS (Machine Word Functions).

3.4.1.1 Entry Points: LSHIFT, RSHIFT, ANDF, ØRF, XØRF, CØMPLF, KØRSZ, CØRWDS.

3.4.1.2 Purpose

To perform basic computer word manipulations by standard binary digit (bit) operations. The manipulations are performed over the complete memory word length for the particular hardware. Also, to determine the size of open core (CØRSZ) and the absolute difference between locations in core (CØRWDS).

3.4.1.3 Calling Sequence

All machine word functions are executed as FØRTTRAN integer function subroutines with integer arguments.

3.4.1.4 Method

The method employed within each function will be described following the separate function examples.

3.4.1.5 Entries

$$K = \text{LSHIFT} (I, N)$$

The entire bit structure of word I is shifted left N places and the resulting word replaces word K. Word I is unchanged. High-order bits shifted out are lost. Zeros are supplied to vacated low-order positions. The shift is logical; no special provision is made for the sign position.

$$K = \text{RSHIFT} (I, N)$$

The entire bit structure of word I is shifted right N places and the resulting word replaces word K. Word I is unchanged. Low-order bits shifted out are lost. Zeros are supplied to vacated high-order positions. The shift is logical; no special provision is made for the sign position.

$$K = \text{ANDF} (I, J)$$

A logical product of the bits within word I and word J is formed and stored into word K. Words I and J are unchanged.

SUBROUTINE DESCRIPTIONS

$K = \text{ORF}(I, J)$

A logical sum of the bits within word I and word J is formed and stored into word K. Words I and J are unchanged.

$K = \text{XORF}(I, J)$

The modulo-two sum (exclusive or) of the bits within word I and word J is formed and stored into word K. Words I and J are unchanged.

$K = \text{COMPLF}(I)$

The ones complement of the bits within word I is formed and stored into word K. Word I is unchanged.

$K = \text{CORSZ}(I, J)$

The size of open core is computed and stored in location K through this function. Location I is normally the address of a labeled common cell defining the beginning of a particular open core area. Location J is normally the address of blank common (usually thought to be the end of a particular open core area). On computer memory configurations where blank common does not define the end of open core, CORSZ ignores location J and substitutes a correct end value. The arguments I and J may be interchanged without affecting results.

$K = \text{COWDS}(I, J)$

The absolute difference plus 1 between the addresses of locations I and J is computed and stored into word K. Words I and J are unchanged.

3.4.1.6 Design Requirements

MAPFNS is written in assembly language.

UTILITY SUBROUTINE DESCRIPTIONS

THE MATERIAL PREVIOUSLY ON PAGE 3.4-3 HAS BEEN DELETED

SUBROUTINE DESCRIPTIONS

THE MATERIAL PREVIOUSLY ON PAGE 3.4-4 HAS BEEN DELETED

UTILITY SUBROUTINE DESCRIPTIONS

THE MATERIAL PREVIOUSLY ON PAGE 3.4-5 HAS BEEN DELETED

SUBROUTINE DESCRIPTIONS

THE MATERIAL PREVIOUSLY ON PAGE 3.4-6 HAS BEEN DELETED

UTILITY SUBROUTINE DESCRIPTIONS

THE MATERIAL PREVIOUSLY ON PAGE 3.4-7 HAS BEEN DELETED

SUBROUTINE DESCRIPTIONS

THE MATERIAL PREVIOUSLY ON PAGE 3.4-8 HAS BEEN DELETED

UTILITY SUBROUTINE DESCRIPTIONS

THE MATERIAL PREVIOUSLY ON PAGE 3.4-9 HAS BEEN DELETED

SUBROUTINE DESCRIPTIONS

THE MATERIAL PREVIOUSLY ON PAGE 3.4-10 HAS BEEN DELETED

UTILITY SUBROUTINE DESCRIPTIONS

THE MATERIAL PREVIOUSLY ON PAGE 3.4-11 HAS BEEN DELETED

SUBROUTINE DESCRIPTIONS

3.4.10 NASTIØ (NASTRAN Input/Output Manager)

3.4.10.1 Entry Point: NASTIØ

3.4.10.2 Purpose: To manage core storage in which data blocks may be core resident.

3.4.10.3 Calling Sequence:

CALL NASTIØ (AREA, LENGTH, NAMES, CØUNT), where three types of calls exist, depending on LENGTH:

1. LENGTH > 0 implies a call to furnish NASTIØ with working storage for the purpose of holding NASTRAN data blocks in core. In this case,
AREA = Address of the area of working storage.
LENGTH = Number of words in the area.
NAMES = Address of a list of 1-word entries defining the GINØ reference names of candidates for core files.
CØUNT = Number of 1-word entries in NAMES list.
2. LENGTH = 0 implies a call to modify the candidates list. In this case, the NAMES list specifies GINØ reference names to be dropped from the previously supplied NAMES list.
AREA = Address of the area of working storage supplied in previous LENGTH > 0 call.
LENGTH = 0
NAMES = Address of a list of 1-word entries defining GINØ reference names to be dropped from previously supplied list.
CØUNT = Number of 1-word entries in NAMES list.
3. LENGTH < 0 implies a call to return working storage previously furnished to NASTIØ by the user.
AREA = Address of working storage to be released (must be the same as the previous call).
LENGTH = Absolute value - number of words to be released (must be the same as the previous LENGTH > 0 call).
NAMES = Address of NAMES list in previous LENGTH > 0 call (not used in this call).
CØUNT = Number of 1-word entries in NAMES list (not used in this call).

Design Notes for NASTIØ

1. The order of calls must be as follows:

1 LENGTH > 0 call

0-n LENGTH = 0 calls

1 LENGTH < 0 call

2. A module should release all working storage (LENGTH < 0 type call) before returning control to XSEM.
3. It is permissible for the sequence of calls listed in Note 1 above to occur more than once within a module.

3.4.10.4 Method:

Implementation of NASTIØ is computer dependent. The basic logic is, however, similar on each of the NASTRAN computers for which NASTIØ has been implemented.

The amount of core furnished through a LENGTH > 0 call is divided into equal blocks according to the GINØ block length. These blocks are chained in a threaded list. Initially all blocks belong to the "free" chain.

Data blocks are written in the normal manner to secondary storage. When a data block scheduled for core residency is opened to read, all GINØ blocks in the data block which can fit into available core are read into core. For each GINØ block the corresponding entry in the threaded list of free blocks is removed and added to a threaded list of blocks belonging to the particular data block. Note that this activity occurs at the time of CALL ØPEN. Subsequent calls from GINØ to read a block result in a substitution of a new buffer address only.

When a data block is dropped from core status through a LENGTH = 0, LENGTH < 0 or a write operation on the same unit, all core blocks previously assigned are added to the threaded list of "free" blocks.

3.4.10.5 Diagnostic Messages

Messages generated by NASTIØ are computer dependent. See Section 5 for a detailed list.

SUBROUTINE DESCRIPTIONS

3.4.11 OPEN (Initiate Activity on a File)

3.4.11.1 Entry Point: OPEN

3.4.11.2 Purpose: To initiate activity for a file.

3.4.11.3 Calling Sequence:

CALL OPEN (\$n,NAME,BUFF,OP)

n - FORTRAN statement number defining the return to be taken in the event NAME is not in the FIST.

NAME - GINØ file name.

BUFF - Address of buffer to be used for file activity. Length of the buffer is defined by the contents of the first word of /SYSTEM/.

OP {
-2, write end-of-file and rewind if file was previously opened to write
and is now off the load point
0, open file to read with rewind
1, open file to write with rewind
2, open file to read without rewind
3, open file to write without rewind

Note

For OP = -2 call, no action takes place if stated conditions are not met. The file is closed on return from OPEN when OP = -2.

3.4.11.4 Method

If the file is currently positioned off the load point and was previously opened to write and OP = -2 or OP = 0 or OP = 2, OPEN calls QOPEN to write with no rewind, calls EOF and then calls CLOSE. If OP = -2, OPEN returns. Otherwise, OPEN calls QOPEN with OP set as requested. If OP = 2, BCKREC is called to position prior to the end-of-file which was just written.

If any of the above conditions are not met, QOPEN is called directly.

UTILITY SUBROUTINE DESCRIPTIONS

3.4.12 GINØ (General Input/Output Routine)

3.4.12.1 Entry Points: GINØ, QØPEN, WRITE, READ, CLØSE, BCKREC, FWDREC, SKPFIL, REWIND, EØF, SAVPØS, GETURN, FILPØS, RDBLK, WRTBLK, PUTSTR, ENDPUT, GETSTR, ENDGET, GETSTB, ENDGTB, RECTYP

3.4.12.2 Purpose: To provide general input/output services for all NASTRAN data block operations.

3.4.12.3 Calling Sequence:

```
CALL GINØ (NAME, ØPCØDE)
```

This entry point is provided in Level 16 as a general service entry but no service operations have been defined. Therefore, in Level 16, it results in no operation.

To initiate activity on the requested file:

```
CALL QØPEN ($n,NAME,BUFF,ØP)
```

n - FØRTRAN statement number defining the return to be taken in the event NAME is not in the FIST (i.e., the data block is purged).

NAME - GINØ file name of the data block which is to be read or written (see Section 1.6.4.1).

BUFF - An array whose dimension equals the contents of the first word of /SYSTEM/ which will be used by GINØ while the file is open.

ØP = $\begin{cases} 0, & \text{open file to read with rewind} \\ 1, & \text{open file to write with rewind} \\ 2, & \text{open file to read without rewind} \\ 3, & \text{open file to write without rewind} \end{cases}$

To write a logical record, or portion of a logical record:

```
CALL WRITE (NAME,BLØCK,N,EØR)
```

NAME - GINØ file name of the data block which is to be written (see Section 1.6.4.1).

BLØCK - An array of dimension $\geq N$ containing the data words to be written.

N - The number of words to be written - integer - input.

EØR = $\begin{cases} 0, & \text{the N words to be written by this call do not end the logical record,} \\ & \text{i.e., subsequent WRITE calls will provide additional data to be written} \\ & \text{in the current logical record.} \\ 1, & \text{the N words to be written by this call end the logical record.} \end{cases}$

SUBROUTINE DESCRIPTIONS

To read a logical record, or portion of a logical record:

CALL READ (\$n₁, \$n₂, NAME, BLØCK, N, EØR, M)

- n₁ - FØRTRAN statement number defining the return to be taken in the event an end-of-file is encountered by this READ operation.
- n₂ - FØRTRAN statement number defining the return to be taken at the end of the READ operation whenever the end-of-logical-record is encountered prior to transmitting the requested number of data words.
- NAME - GINØ file name of the data block which is to be read (see Section 1.6.4.1).
- BLØCK - An array of dimension $\geq N$, where the words read will be stored.
- N $\left\{ \begin{array}{l} N > 0: \text{ The number of words to be read and stored at BLØCK - integer - input.} \\ N \leq 0: \text{ The number of words to be skipped, i.e., read but not stored at BLØCK. Integer - input.} \end{array} \right.$
- EØR $\left\{ \begin{array}{l} 0, \text{ subsequent calls to READ for the current logical record are expected.} \\ 1, \text{ the current call is the last call for the current logical record. The file will be positioned to the beginning of the next logical record before returning.} \end{array} \right.$
- M - If return to n₂ is given, the number of words actually read is stored in M. In no other case are the contents of M changed.

Whenever return to n₂ is given, the file is positioned to the beginning of the next logical record regardless of the setting of EØR.

A return to n₁ is possible only when a call to READ is given when the file is positioned at the beginning of a logical record.

To terminate activity on the requested file:

CALL CLØSE (NAME, ØP)

NAME - GINØ file name of the data block to be closed (see Section 1.6.4.1).

- ØP $\left\{ \begin{array}{l} 1, \text{ if file was opened to write, write end-of-file and rewind.} \\ \quad \text{If file was opened to read, rewind only.} \\ 2, \text{ close file at current file position (no end-of-file, no rewind).} \\ 3, \text{ if file was opened to write, write end-of-file and position file in} \\ \quad \text{front of end-of-file mark. If file was opened to read, same as} \\ \quad \text{ØP = 2.} \end{array} \right.$

If the requested file is not in the FIST or is not currently open, a normal return is given and no operation takes place.

If the file was opened for output and the last logical record has not been written, it is written prior to honoring the ØP request.

UTILITY SUBROUTINE DESCRIPTIONS

The buffer assigned when the file was opened is released and is available to the user on return.

To position the requested file backward one logical record:

CALL BCKREC (NAME)

NAME - GINØ file name of data block to be positioned backward (see Section 1.6.4.1).

If the file is positioned in the middle of a logical record, the file is repositioned to the beginning of that record. Otherwise, the file is positioned to the beginning of the previous logical record.

If the file is positioned at the beginning of file, no operation occurs and a normal return is given.

To position the requested file forward one logical record:

CALL FWDREC (\$n,NAME)

n - FORTRAN statement number defining the return to be taken in the event an end-of-file is encountered.

NAME - GINØ file name of data block to be positioned forward (see Section 1.6.4.1).

This call will always position the file to the beginning of the next logical record unless an end-of-file is encountered.

To position the requested file forward or backward a stated number of files:

CALL SKPFIL(NAME,N)

NAME - GINØ file name of the data block to be repositioned (see Section 1.6.4.1).

N - The number of files to be skipped. N > 0 means forward skip, N < 0 means backward skip, N = 0 means no operation - integer - input.

Notes:

1. Following a forward skip, the file is positioned at the beginning of the first logical record (i.e., immediately after the end-of-file mark).
2. Following a backward skip, the file is positioned immediately in front of the end-of-file mark (or at the beginning-of-unit).
3. Request to skip backward from the beginning-of-unit is ignored and the file remains positioned at the beginning-of-unit.
4. SKPFIL backward on a file opened to write is not permitted.

SUBROUTINE DESCRIPTIONS

To rewind the requested file:

CALL REWIND(NAME)

NAME - GINØ file name of the data block to be rewound (see Section 1.6.4.1).

Rewind given for an output file has the effect of erasing any data which has been written on the file.

To write an end-of-file on the requested file:

CALL EOF(NAME)

NAME - GINØ file name of data block on which end-of-file is to be written (see Section 1.6.4.1).

The file must be open to write at the time of this call.

To save the "GINØ address" of the current logical record on the requested file:

CALL SAVPOS (NAME,POS)

NAME - GINØ file name of data block.

POS - Address of the word into which SAVPOS will return the GINØ address of the current logical record - integer - output.

The "address" returned by SAVPOS contains the block number and position within the block in which the logical record begins. The bit positions within the word in which these two quantities reside is machine dependent.

To find the unit reference number corresponding to the GINØ file name:

CALL GETURN (NAME)

NAME - GINØ file name.

Notes:

1. NAME must be stored in /GINØX/ prior to entry.
2. FILEX in /GINØX/ must be set to zero prior to entry.
3. The unit reference number is returned in FILEX in /GINØX/. If FILEX = 0 on return, NAME is not in the FIST.
4. GETURN will also return the contents of UNITAB (FILEX) in EØR, XYZ(1) and XYZ(2) in /GINØX/. If GETURN is called when the file is not open, EØR will contain the read/write bit from the previous open, XYZ(1) and XYZ(2) will contain the block number and record pointer defining the position at which the file was closed.

UTILITY SUBROUTINE DESCRIPTIONS

To position the requested file to the stated file position:

CALL FILPOS (NAME,POS)

NAME - GINØ file name.

POS - "GINØ address" of requested logical record. This word should have been furnished through a previous CALL SAVPOS for the file - integer - input.

The requested file must be open for input at the time of a CALL FILPOS.

To determine the type of the logical record at which the requested file is currently positioned:

CALL RECTYP (NAME,TYPE)

NAME - GINØ file name.

TYPE - Output - integer { 0: normal GINØ logical record
 1: record written in string format

The requested file must be open for input at the time of a CALL RECTYP.

Two special entry points are provided for efficient data transfers between the Substructuring Operating File and GINØ.

To read the next physical block of a file into the buffer:

CALL RDBLK (\$n,NAME,FIRST,LEFT)

n - FORTRAN statement number defining the return to be taken if block read is the last in the file.

NAME - GINØ file name of the data block to be read (see Section 1.6.4.1).

FIRST - { 1, this is the first call to read a block after opening the file.
 0, this is a call to read additional blocks from the file.

LEFT - If return to n is given, the number of unused words remaining in the last buffer.

- Notes:
1. The first block is read into the buffer on the call to OPEN, but RDBLK must be called with FIRST non-zero to check if it is the last block.
 2. RDBLK terminates reading the file and takes the non-standard return when the first end-of-file is encountered.

To write the current contents of the buffer as the next physical block of a file:

CALL WRTBLK (NAME,EØF)

NAME - GINØ file name of the data block to be written (see Section 1.6.4.1)

EØF - { 1, this is the last call to write a block before closing the file.
 2, subsequent calls to WRTBLK will provide additional data for the file.

SUBROUTINE DESCRIPTIONS

- Notes:
1. The last block will be written on the file by the call to CLØSE, but WRTBLK should be called first with EOF non-zero to properly set the buffer pointers.
 2. The file should be closed with ØP = 1 or 3 to write an end-of-file mark on the file.

The entry points PUTSTR, ENDPUT, GETSTR, ENDGET, GETSTB, and ENDGTB allow for matrix string manipulation directly from the buffer. Associated with string calls is a communication block.

The communication block for all string calls is as follows:

<u>Word</u>	<u>Description</u>	<u>Input or Output Entry Points</u>
1	GINØ file name	input to all
2	Type of elements (1,2,3,4) 1 = real single precision 2 = real double precision 3 = complex single precision 4 = complex double precision	{ input to PUTSTR output from GETSTR, GETSTB
3	Format of strings (0,1) 0 implies no string trailers are written in the buffer 1 implies string trailers are written in the buffer	{ input to PUTSTR output from GETSTR, GETSTB
4	Row position of {first} {last} element in string	input to ENDPUT output from {GETSTR/ GETSTB}
5	Pointer to string	output from PUTSTR, GETSTR, GETSTB
6	Number of terms {available} {actual} in string	output from PUTSTR, GETSTR, GETSTB
7	Number of terms written in string	input to ENDPUT

UTILITY SUBROUTINE DESCRIPTIONS

Word	Description	Input or Output Entry Points
8	Begin/end flag (-1,0,+1) -1 = first call 0 = intermediate call 1 = last call	{ input to all { output from ENDGET, ENDGTB
9-11	Reserved for internal use by the various entry points	
12	Column number	{ input to PUTSTR { output from GETSTR, GETSTB

Notes:

The pointer returned in word 5 is relative to /XNSTRN/ and depends on the precision of the elements (word 2). For double precision elements, a double precision reference is made to the element in the buffer. Thus, for complex double precision elements:

```
DOUBLE PRECISION XND,SSQR
COMMON /XNSTRN/ XND(1)
EQUIVALENCE ( BLOCK(5), JP0INT )
.
.
.
CALL GETSTR ($n,BLOCK)
SSQR = XND(JP0INT)**2 + XND(JP0INT+1)**2
.
.
.
```

For GETSTB calls, the pointer returned is to the last element in the string. The string should be processed "backwards".

For the convenience of assembly language programmers, it is possible to have the true machine address returned in word 5. This option is machine dependent and the user is referred to comments in assembly listings for specific details.

To initiate writing a string directly into the buffer of the requested file:

```
BLOCK(1) = GIN0 file name
BLOCK(2) = Type of elements
BLOCK(3) = Format of strings
BLOCK(8) = -1
BLOCK(12) = Column number
CALL PUTSTR (BLOCK)
```

} prior to first string
for column only

To terminate processing a string furnished by PUTSTR:

```
BLOCK(7) = number of terms written in string
BLOCK(8) = 1 (last string in column)
CALL ENDPUT (BLOCK)
```

SUBROUTINE DESCRIPTIONS

THIS PAGE HAS BEEN LEFT BLANK INTENTIONALLY.

UTILITY SUBROUTINE DESCRIPTIONS

To initiate reading a string directly from the buffer on the requested file:

```
BLØCK(1) = GINØ file name      }      prior to first string  
BLØCK(8) = -1                  }      for column only  
CALL GETSTR ($n,BLØCK)
```

n - FØRTRAN statement number defining return taken when there does not exist another string in the column.

To terminate processing a string furnished by GETSTR:

```
CALL ENDGET (BLØCK)
```

3.4.12.4 Method

To provide maximum efficiency, GINØ is coded in assembly language on each of the NASTRAN computers. The logic of each of the versions is similar, however.

GINØ blocks all logical records into fixed length blocks (physical records). A buffer area is furnished with each call ØPEN. The format of the buffer is machine dependent and are described in Figure 1 for the CDC, IBM and UNIVAC computers respectively.

The actual address of the buffer is either that furnished by the CALL ØPEN or the address plus one. The criterion is that the displacement in words from /XNSTRN/ be even (so that accesses to double precision words is insured).

A description of the blocking techniques used for matrix and non-matrix data is found in Figure 2.

Formats of the various control words are computer dependent and are defined in Table 1 for each computer. Their contents are the same.

- The record header indicates the type of record (regular or "string" data) and the number of words in the record.
- The record trailer contains the "GINØ address" (block number and position) of the record and continuation indicators.
- The last control word is either an end-of-file flag or an indicator that more information is contained in the next block.

An example of string packing of matrix data is found in Figure 2. A string is defined as a set of contiguous non-zero rows in a matrix column. The example in Figure 2 shows two strings implying all terms of the column are "0" except for rows 2, 3, 9 and 10.

SUBROUTINES DESCRIPTIONS

The following is a description of the control words:

- The column header and column trailer contain the type of elements in the strings, format of the strings and the column number.
- The string {header } contains the row position of the {first } term in the string and the number of terms in the string. On the CDC and UNIVAC computers the string header or trailer is one word while on the IBM 360 each is two words.
- Two additional control words may appear in the place of a string header:
 1. dummy string word. This is used as a pad to insure proper alignment of double precision terms.
 2. "last" string word. This indicates no more strings in the current block.

On the 360 and 6000 series, current buffer pointer and current logical record pointer are maintained as offsets relative to the address of the buffer. On the 1108, current buffer pointer and current logical record pointer are maintained as actual addresses.

Most key parameters are maintained in /GINØX/ as follows:

<u>Word</u>	<u>Variable</u>	<u>Description</u>
1	LGINØX	Length of /GINØX/
2	FILEX	Current unit reference number
3	EØR	End-of-record indicator for READ/WRITE
4	ØP	Operation code for ØPEN/CLØSE
5	ENTRY	Code for current entry point
6	LSTNAM	GINØ file name on previous entry
7	N	Number of words to transmit on READ/WRITE
8	NAME	GINØ file name on current entry
9	NTAPE	Device flag (0 implies disk, ≠ 0 implies tape)
10-11	XYZ	Scratch
12-86	UNITAB	Position maintenance table
87-161	BUFADD	Buffer address maintenance table
162	NBUFF3	Block size
163-166		(computer dependent)

The Position maintenance table has a one word entry for every file that is maintained by GINØ.

The format of each word is as follows:

R/W	BLØCK NØ.	LØGICAL RECØRD
-----	-----------	----------------

UTILITY SUBROUTINE DESCRIPTIONS

where

R/W	= 0	if file last opened for read
	= 1	if file last opened for write
BLØCK NØ.		block number of position of file
LØGICAL RECØRD		current logical record pointer

The buffer address maintenance table has a one word entry for every file that is currently opened.

The format of each word is as follows:

INCØRE BUFF	R/W	BUFF ADDR
-------------	-----	-----------

where

INCØRE BUFF		buffer address supplied when ØPEN was called and the file is to be held in core
R/W	= 0	if file opened for read
	= 1	if file opened for write
BUFF ADDR		buffer address from which GINØ will be blocking or unblocking

3.4.12.5 Design Requirements

1. Maximum block size = 4095 words
2. Minimum block size 360 - 64 words
 1180 - 143 words
3. Maximum number of blocks in a single file (NASTRAN data block) = 65535
4. QØPEN should only be called from SUBRØUTINE ØPEN.

3.4.12.6 Diagnostic Messages

Section 5 contains details of the various messages which may occur in GINØ.

SUBROUTINE DESCRIPTIONS

CDC
NASTRAN
I/O
Subsystem
Buffer

Physical
Record
Block
to be
written

Word Count	Contents
1	Data Block
2	Current Buffer Pointer (bits 35-18) Current Logical Record Pointer (bits 17-0)
3	Not used
4	Current Block Number
5	Pointer to Last Control Word
6	Logical Record Header : :
NBUFF	End of Block Definition Word
NBUFF + 30	File Environment Table (FET)
NBUFF + 30 + 126	Subindex Array

IBM
NASTRAN
I/O
Subsystem
Buffer

Physical
Record
to be
written

Word Count	Contents
1	Data Block Reference Name
2	Current Buffer Pointer
3	Current Logical Record Pointer
4	Current Block Number
5-6	8 Bytes Containing Full Disk Address in the Form of MBCCCHRR (DCBFDA0)
7	1 Byte Containing the DCB Open Flags (DCBOFLGS) 3 Bytes of Track Balance (DCBTRBAL)
8	Pointer to Last Control Word
9	Logical Record Header : :
NBUFF	End of Block Definition Word

UNIVAC
NASTRAN
I/O
Subsystem
Buffer

Physical
Record
to be
written

Word Count	Contents
1	Current Buffer Pointer
2	Current Logical Record Pointer
3	Address of End of Buffer - 2
4	Current Block Number
5	Pointer to Last Control Word
6	Logical Record Header : :
NBUFF	End of Block

Figure 1. Buffer Format for the CDC, IBM and UNIVAC I/O Subsystem

UTILITY SUBROUTINE DESCRIPTIONS

Example	Operation	Number of Words	Description
1	Call to WRITE	1	Record Header
		10	Logical Data Record
		1	Record Trailer
2	Call to PACK	1	String Data
		1	Column Header
		1	String Header
		2	Rows 2 and 3 Stored in Real Single Precision
		1	String Trailer
		1	String Header
		2	Rows 9 and 10 Stored in Real Single Precision
		1	String Trailer
		1	Column Trailer
		1	Record Trailer
3	Call to CLØSE	1	End of File Definition Word
		1	End of Block Definition Word

Figure 2. An example of Blocked Logical Records in a Physical Record Block

Table 1. GINØ Definition Words

Definition Word	CDC				IBM				UNIVAC			
	59 54	53 36	35 18	17 0	0 3	4 19	20 31	35 30	29 12	11 0		
Record Header	0	11111	0	NBR WDS	F	1111	NBR WDS	77	11111	NBR WDS		
String Data	0	22222	0	NBR WDS	F	2222	NBR WDS	77	22222	NBR WDS		
Record Trailer												
A. Contained in Block	0	33330	BLK NØ	CLR	2	-	CLR	11	BLK NØ	CLR		
B. Last of Continued Record	0	33332	BLK NØ	CLR	2	-	CLR	12	BLK NØ	CLR		
C. Record Continued in Next Block	0	33333	BLK NØ	CLR	3	-	CLR	13	BLK NØ	CLR		
End of Block	0	55555	0	0	F	5555	0	77	55555	0		
End of File	0	77777	0	0	F	7777	0	77	77777	0		
Column Header	0	CØL NBR	7	8*TYPE+FØRM	4	CØL NBR	16*TYPE+FØRMAT	21	CØL NBR	8*TYPE+FØRM		
Column Trailer	0	CØL NBR	377777	8*TYPE+FØRM	8	CØL NBR	16*TYPE+FORMAT	27	CØL NBR	8*TYPE+FØRM		
String Header * IBM	0	RØW NBR	17	NBR TERMS	F	8888	NBR TERMS	31	RØW NBR	NBR TERMS		
String Trailer	0	RØW NBR	37	NBR TERMS	F	9999	NBR TERMS	37	RØW NBR	NBR TERMS		
Dummy String Def. Word	0	0	777	0	F	AAAA	0	77	33333	0		
End of Block String Word	0	0	77	0	F	BBBB	0	77	44444	0		

* For IBM the next word contains the row number.

SUBROUTINE DESCRIPTIONS

THE MATERIAL PREVIOUSLY ON PAGE 3.4-20 HAS BEEN DELETED

UTILITY SUBROUTINE DESCRIPTIONS

THE MATERIAL PREVIOUSLY ON PAGE 3.4-21 HAS BEEN DELETED

SUBROUTINE DESCRIPTIONS

3.4.14 GOPEN (Short Form for Subroutine OPEN With Header Record Processing).

3.4.14.1 Entry Point: GOPEN.

3.4.14.2 Purpose

To provide a short form (without the non-standard return of subroutine OPEN) for opening a GINØ file, and to write a two-word header record if the data block is opened as output with rewind or to skip the header record if the data block is opened as input with rewind.

3.4.14.3 Calling Sequence

CALL GOPEN(FILE,BUFFER,ØPT)

where:

FILE = GINØ file name (see section 1.6.4.1).

BUFFER = GINØ buffer location.

ØPT = any of the open options permitted by subroutine OPEN (see section 3.4.2).

3.4.14.4 Method

Open the file (subroutine OPEN). If ØPT = input with rewind (0), skip the first record of the data block. If ØPT = output with rewind (1), write the two word BCD name of the data block as returned by subroutine FNAME.

3.4.14.5 Design Requirements

The data block must exist (must not be purged). If ØPT = input with rewind (0), the first record of the data block must be at least two words long. Subroutines used: OPEN, READ, WRITE, FNAME, MESSAGE.

3.4.14.6 Diagnostic Messages

If the data block is purged or if an end-of-file or end-of-record condition is encountered when reading the data block, subroutine MESSAGE will be called with internal message numbers 1, 2, or 3, respectively (external message numbers are 3001, 3002 and 3003).

UTILITY SUBROUTINE DESCRIPTIONS

3.4.15 FREAD (Short Form for Subroutine READ).

3.4.15.1 Entry Point: FREAD.

3.4.15.2 Purpose

To provide a short form (without the non-standard returns of subroutine READ) of reading a GINØ file.

3.4.15.3 Calling Sequence

CALL FREAD(FILE,BLØCK,N,EØR)

where:

FILE = GINØ file name (see section 1.6.4.1).

BLØCK= array into which N items are to be read.

N = number of items to be read.

EØR = any end of record option permitted by subroutine READ (see section 3.4.5).

3.4.15.4 Method

Read the N items from FILE into BLØCK. If subroutine READ returns an end-of-file or end-of-record condition, subroutine MESSAGE is called with a fatal error condition.

3.4.15.5 Design Requirements

In addition to those imposed by READ, there must be N items remaining in the record to be read. Subroutines used: READ, MESSAGE.

3.4.15.5 Diagnostic Messages

Subroutine MESSAGE may be called with internal message number 2 or 3 (external message numbers 3002,3003).

SUBROUTINE DESCRIPTIONS

3.4.16 WRTTRL (Write Trailer).

3.4.16.1 Entry Points: WRTTRL. RDTRL.

3.4.16.2 Purpose

WRTTRL will pack six words of trailer information into three words and store them in the FIAT.

RDTRL will retrieve and unpack the trailer information.

3.4.16.3 Calling Sequence

CALL WRTTRL(FILBLK)

FILBLK(1) - GINØ file name (see section 1.6.4.1).

FILBLK(2-7) - Trailer information to be stored.

CALL RDTRL(FILBLK)

FILBLK(1) - GINØ file name.

FILBLK(2-7) - Storage space for trailer information.

3.4.16.4 Method

The index into the FIAT for the specified file is located in the FIST. The three packed words are stored in or retrieved from the FIAT. The information is also stored for all files equivalenced to the GINØ file name. For RDTRL, if the file is purged, FILBLK(1) is set negative. If the file is a matrix, word 7 is converted to a density (10000 = 100% dense). Matrix trailers can be displayed as they are written by activating DIAG 8.

3.4.16.5 Design Requirements

Each word of trailer information is assumed to be a positive integer less than $2^{16}-1$. Trailers may not be written on GINØ files 101-199.

3.4.16.6 Diagnostic Messages

If the file did not exist in the FIST when WRTTRL was called, fatal error 3011 occurs.

UTILITY SUBROUTINE DESCRIPTIONS

3.4.17 FNAME (File Name).

3.4.17.1 Entry Point: FNAME.

3.4.17.2 Purpose

Given a GINØ file name, FNAME returns the two BCD words which describe the data block.

3.4.17.3 Calling Sequence

CALL FNAME(FILE,NAME)

FILE - GINØ file name (see section 1.6.4.1).

NAME(2) - Storage for the two BCD words.

3.4.17.4 Method

The GINØ file name is first located in the FIST. The index in the FIST is used to find the BCD descriptors in the FIAT. If the file does not exist in the FIST, "^(NONE)^" is returned as the two words, ^ indicating a BCD blank.

SUBROUTINE DESCRIPTIONS

3.4.18 CLSTAB (Close a GINØ File and Write a Non-zero Trailer).

3.4.18.1 Entry Point: CLSTAB.

3.4.18.2 Purpose

To close a GINØ file and generate a table trailer by calling WRTTRL.

3.4.18.3 Calling Sequence

CALL CLSTAB(FILE,ØPT)

where:

FILE = GINØ file number - integer - input.

ØPT = any close option permitted by subroutine CLØSE (see section 3.4.4) - integer - input.

3.4.18.4 Method

CALL CLØSE(FILE,ØPT)

Generate the table control block, ITABCB:

ITABCB(1) = FILE

ITABCB(7) = 1

DØ 10 I = 2,6

10 ITABCB(I) = 0

CALL WRTTRL (ITABCB).

3.4.18.5 Design Requirements

Same as those for subroutines CLØSE and WRTTRL. Subroutines used: CLØSE, WRTTRL.

UTILITY SUBROUTINE DESCRIPTIONS

3.4.19 XRCARD (Executive Free-Field Card Data Conversion Routine)

3.4.19.1 Entry Point: XRCARD.

3.4.19.2 Purpose

To interpret NASTRAN free-field card input data as follows:

1. Identify BCD alpha and numeric data fields as they are converted and placed in the user's buffer;
2. Flag and output special data field delimiters;
3. Convert BCD numeric fields to binary integer or binary floating point;
4. Indicate when the data extends beyond one 72 column card.

3.4.19.3 Calling Sequence

CALL XRCARD(OUTBUF,L,INBUF)

Where:

- OUTBUF = The buffer which is to contain the converted card image.
- L = The length of OUTBUF available to XRCARD.
- INBUF = The buffer containing the card image to be converted.

3.4.19.4 Method

XRCARD's design is based on the necessity of having to function on a variety of computing machines having a variety of computer word structures, and a variety of differences in hollerith handling imposed by differing FORTRAN compilers.

XRCARD analyzes the twenty hollerith words input through INBUF as follows:

Data Field Delimiters

Type A:

The following symbols signify the end of an alpha field or numeric field on the card. As these symbols are encountered, they will be flagged and placed in the output buffer to aid the user in identifying the data.

- (LEFT PAREN
- / SLASH
- = EQUAL

SUBROUTINE DESCRIPTIONS

Type B:

The following symbols are identical to those listed above except that the symbol is not flagged or placed in the output buffer:

, COMMA

) RIGHT PAREN

When successive type A or type B delimiters are encountered, a null field indication (two BCD blank words) is output. A null field is generated for each successive delimiter. A null field is also generated when a type A or type B delimiter is followed by a \$ indicating the end of data condition.

Type C:

The following symbol is identical to the COMMA except that no null field indication is output when they are encountered in succession.

^ BLANK

End of Data Indication

There are three means by which end-of-data may be specified on the card:

- The last data field ends in column 72, or is followed by blanks out through column 72;
- \$ is encountered, after which comments may be included out to column 80; or
- Continuation cards ending in (, /, = or , will result in a continuation flag (0 mode word).

Format of Output Data

A mode word, N, is placed in the output buffer to distinguish between BCD data and numeric data.

Numeric Mode Word: A new mode word is output each time a numeric field is converted and output. (All numeric mode words are negative).

- N = -1 integer data (1 data word)
- = -2 floating point single precision (1 data word)
- = -4 floating point double precision (2 data words)

N indicates the type of numeric data and where to look for the next mode word.

UTILITY SUBROUTINE DESCRIPTIONS

Alpha Mode Word: When processing alpha data, only one mode word is output for successive alpha fields, i.e., an alpha mode word will never follow another alpha mode word.

N = The number of successive alpha fields encountered on the card. Each alpha field consists of two 4-character computer words (left adjusted). Thus N can be used to compute the location of the next mode word.

The type A delimiters are output as alpha data and are 'covered' by the alpha mode word. Since data output in the alpha mode must consist of two words a type A delimiter will appear as:

Word 1 = Delimiter flag, all bits of the word are on.

Word 2 = BCD delimiter, left adjusted, followed by BCD blanks.

End-of-Data: The end-of-data flag is placed last in the output buffer and appears in place of an expected mode word. There are two end-of-data flags:

- A word with all bits off, indicating that more data is to follow on a continuation card.
- A word with all bits on except for the sign, indicating that no more data is to follow for this card type.

Sample Input Card

CARDA A=1,B=1.0,ABC/CDEFGH GOOD DATA

[illegible]

SUBROUTINE DESCRIPTIONS

Resulting Output Buffer for IBM 7094 or Univac 1108

		+ (alpha mode word) 3	Number of successive alpha fields (including Type A delimiters)
BCD Field {		C A R D ^ ^	
		A ^ ^ ^ ^ ^	
BCD Field {		A ^ ^ ^ ^ ^	
		^ ^ ^ ^ ^ ^	
Output Delimiter {		all bits on	
		= ^ ^ ^ ^ ^	
		-(numeric mode word) 1	
		integer 1	
		+ (alpha mode word) 2	
		B ^ ^ ^ ^ ^	
		^ ^ ^ ^ ^ ^	
Output Delimiter {		all bits on	
		= ^ ^ ^ ^ ^	
		-(numeric mode word) 2	
		single-precision 1.0	
		+ (alpha mode word) 5	
		A B C ^ ^ ^	
		^ ^ ^ ^ ^ ^	
		all bits on	
		/ ^ ^ ^ ^ ^	
		C D E F ^ ^	
		G H ^ ^ ^ ^	
		G Ø Ø D ^ ^	
		^ ^ ^ ^ ^ ^	
		D A T A ^ ^	
		^ ^ ^ ^ ^ ^	
		all bits on sign bit off	End of data for this card

NOTE: For the IBM S/360 the output buffer shown here looks the same except that the right two blanks shown in the BCD fields here do not exist. For the CDC 6600 there are an additional four trailing blanks in each word of a BCD field than shown here.

^ Indicates blank.

UTILITY SUBROUTINE DESCRIPTIONS

3.4.19.5 Design Requirements

An alpha field must be eight characters or less. Long alpha fields will be truncated to eight characters.

All data must be placed in card columns 1-72.

A data field may not be split between two cards.

The specification of all numeric data fields must conform to FORTRAN IV standards.

If an error condition is encountered, e.g., data bad, XRCARD will write a message, turn on the NOGG flag in /SYSTEM/, set the first word of OUTBUF = 0, and make a normal return to the calling program.

SUBROUTINE DESCRIPTIONS

3.4.20 RCARD (Fixed Field Card Data Conversion Routine).

3.4.20.1 Entry Point: RCARD.

3.4.20.2 Purpose

To interpret NASTRAN fixed-field (bulk data) card input as follows:

- Identify BCD alpha and numeric data fields as they are converted and placed in the users buffer; and
- Convert BCD numeric fields to binary integer or binary floating point.

3.4.20.3 Calling Sequence

```
CALL RCARD(ØUTBUF,FRMTBF,NFLAG,INBUF)
```

Where:

- ØUTBUF = The buffer which is to contain the converted card image.
- FRMTBF = A buffer which contains identification flags for the converted data in ØUTBUF.
- NFLAG = Contains number of words returned in ØUTBUF.
- INBUF = The buffer containing the card image to be converted.

Definition of Data Identification Flags Placed in FRMTBF

- 0 = output for a blank data field.
- 1 = output for an integer field.
- 2 = output for a floating point field.
- 3 = output for a BCD field.
- 4 = output for a double precision floating point field.
- 1 = error.

3.4.20.4 Method

RCARD's design is based on the necessity of having to function on a variety of computing machines having a variety of computer word structures, and a variety of differences in Hollerith handling imposed by differing FØRTRAN compilers.

Twenty 4-Hollerith words are received by RCARD on any particular call to RCARD. RCARD first determines from field 1 (words 1 and 2) if the data card is a continuation card, and whether the

UTILITY SUBROUTINE DESCRIPTIONS

fields are single (2 words each) or double (4 words each) in length. Fields 2 through 9 (for single field cards) or 2 through 5 (for double field cards) are then considered one at a time. No consideration is made for the last field of any card (words 19 and 20).

3.4.20.5 Design Requirements

1. All BCD fields must begin with an alphabetic character.
2. All BCD fields are defined to be eight characters in length. Names with less than eight characters will be filled with BCD blanks.
3. When placed in the user output buffer, each BCD field will be divided into two four-character words (left adjusted) and stored in two successive locations of the output buffer. The remainder of the words is filled with BCD blanks.
4. Special characters are not to be used as part of a BCD field except for * and + in field 1 (column 1) which indicate a double field or single field (respectively) continuation card.
5. The data fields will be stored successively in the users output buffer as they are encountered in scanning the card image from left to right. The number of output core locations required per field type varies:
 - a. Integer field = 1 core word (right adjusted).
 - b. BCD field = 2 core words.
 - c. Real single precision = 1 core word.
 - d. Real double precision = 2 core words.
 - e. Blank field = 1 core word (integer 0).
6. The card type field (field 1) of a continuation card will not be passed along to the user. Two zero words will replace the ID field in the output buffer. Thus the user can easily distinguish the difference between a continuation card and a new card type.
7. A check for bulk data card types SEQGP and SEQEP is made by RCARD. Fields 3, 5, 7, and 9 of these card types are processed by a special conversion.

The input within these special fields will be similar to the Dewey decimal notation and consists of a multiple digit integer and up to three single digit sub-integers; e.g., (354.1.2) and (267.5). The special fields will be converted to a single integer by dropping any decimal points and appending a number of zeros equal to three minus the number of decimal points in the original number; e.g., (354120) and (267500).

SUBROUTINE DESCRIPTIONS

8. RCARD does not know the length of the users output buffer, therefore, no check is made for exceeding the length of the buffer. However, the number of data words placed in the output buffer will be specified in NFLAG.

9. Field 10 will not be passed along to the user.

3.4.20.6 Diagnostic Messages

Fields appearing to be incorrect to RCARD will cause a diagnostic to be written on the system output file followed by a card format heading, a card image echo, and an underlining of the field in question. Also, the /SYSTEM/ NØGØ flag is set .TRUE., a zero is placed in the output buffer for the field, and a -1 is placed in the format buffer for the field. RCARD will print diagnostics for all fields appearing incorrect and make a normal return.

UTILITY SUBROUTINE DESCRIPTIONS

3.4.21 TAPBIT (Tape Bit Test).

3.4.21.1 Entry Point: TAPBIT.

3.4.21.2 Purpose

To examine the tape bit for a permanent GINØ file to determine the existence of a physical tape for that file.

3.4.21.3 Calling Sequence

IF (TAPBIT(FILE)) GØ TØ ...

FILE is the GINØ file name (one of 'PØØL', 'ØPTP', 'NPTP', 'UMF ', 'NUMF', 'PLT1', 'PLT2', 'INPT').

3.4.21.4 Method

The permanent FIST is searched and the tape bit in the corresponding FIAT entry is examined. If the bit is on (indicating the presence of a physical tape), TAPBIT will be set .TRUE.. Otherwise it will be set .FALSE..

3.4.21.5 Design Requirements

The type of TAPBIT must be declared LØGICAL.

3.4.21.6 Diagnostic Messages

A fatal call to MESSAGE occurs if a GINØ file name other than those listed is used.

SUBROUTINE DESCRIPTIONS

3.4.22 PEXIT (Problem Exit).

3.4.22.1 Entry Point: PEXIT.

3.4.22.2 Purpose

To terminate the program.

3.4.22.3 Calling Sequence

CALL PEXIT.

3.4.22.4 Method

The diagnostic message queue is checked and if not empty the message writer MSGWRT is called. If the checkpoint flag is set a card is punched indicating the end of the restart checkpoint dictionary. The system output buffers are flushed and then the job is terminated.

3.4.22.5 Design Requirements

PEXIT must have access to the FORTRAN I/O routines.

PEXIT should not be called by module writers. Termination should be via a call to MESSAGE (i.e., CALL MESSAGE(-61,0, NAME)).

UTILITY SUBROUTINE DESCRIPTIONS

3.4.23 TMTØGØ (Time-To-Go).

3.4.23.1 Entry Point: TMTØGØ.

3.4.23.2 Purpose

Computes the running time remaining for this NASTRAN problem.

3.4.23.3 Calling Sequence

CALL TMTØGØ (TIME)

TIME = Remaining time in integer seconds.

3.4.23.4 Method

During NASTRAN problem initialization, one system cell is set to the problem starting time (PSTART) while another is set to the maximum running time (MXTIME) contained on the Executive Control Deck TIME card. TIME-TØ-GØ is then found by reading the clock (NØW) and solving the following:

$$\text{TIME-TØ-GØ} = \text{MXTIME} - (\text{NØW} - \text{PSTART}).$$

The CPU clock is utilized on all machines except the IBM 7094 where none is available.

SUBROUTINE DESCRIPTIONS

3.4.24 PAGE (Page Heading).

3.4.24.1 Entry Points: PAGE, PAGE1, PAGE2

3.4.24.2 Purpose

To provide a standard page heading for NASTRAN output.

3.4.24.3 Calling Sequence

CALL PAGE

CALL PAGE1

CALL PAGE2(N)

COMMON/SYSTEM/XXX,ØTPE,SPACE(6),IPAGE,LINE,ITLINE,MAXLIN,DATE(3)

ØTPE - System output unit - integer.

IPAGE - Current page number - increased by 1 on each call to PAGE.

LINE - Number of data lines on previous page - LINE is set to zero by PAGE.

ITLINE - Total number of lines of printout in run - ITLINE = ITLINE + LINE.

MAXLIN - Maximum number of data lines allowed - if ITLINE > MAXLIN, PEXIT will be called.

DATE(3)- Today's date: month, day, year - integer.

N - Number of lines to be written - integer - input.

COMMON/OUTPUT/TITLE(32),SUBTIT(32),LABEL(32),HEAD1(32),HEAD2(32),HEAD3(32)

3.4.24.4 Method

PAGE writes a standard 6 line heading from TITLE, SUBTIT, LABEL, HEAD1, HEAD2, HEAD3.

PAGE1 writes only the first 3 lines of a standard header.

PAGE2 restores the page if N lines will not fit on the current page.

3.4.24.5 Design Requirements

ITLINE must be less than MAXLIN. PAGE must have access to the FØRTRAN I/Ø routines.

UTILITY SUBROUTINE DESCRIPTIONS

3.4.25 MESSAGE (Message).

3.4.25.1 Entry Point: MESSAGE.

3.4.25.2 Purpose

To queue nonfatal messages during the execution of a module; and for fatal messages give a core dump (CALL PDUMP), print the message queue (CALL MSGWRT), and call PEXIT.

3.4.25.3 Calling Sequence

CALL MESSAGE(NØ,PARM,NAME)

Where

NØ = Internal message number. NØ positive defines the message as nonfatal;

NØ negative defines the message as fatal.

PARM = Parameter used in the printed message (usually the GINØ file number)

NAME(2) = Two words used in the printed message (usually two BCD words containing the name of the subroutine calling MESSAGE).

3.4.25.4 Method

Non-fatal messages are queued in common block /MSGX/ until the maximum number is reached. All non-fatal messages after this are lost. When a fatal message is encountered, it is queued and appropriate action taken to terminate the run.

3.4.25.5 Design Requirements

The size of common block /MSGX/ limits the number of messages stored.

SUBROUTINE DESCRIPTIONS

3.4.26 MSGWRT (Message Writer).

3.4.26.1 Entry Point: MSGWRT.

3.4.26.2 Purpose

To print NASTRAN error messages on the system output file.

3.4.26.3 Calling Sequence

CALL MSGWRT

COMMON/MSGX/N,M,MSG(4,10)

where:

N - is the total number of messages to be printed.

M - maximum number of messages that can be queued by subroutine MESSAGE in the array MSG.

MSG - array where message parameters are queued.

MSG(1,I) - the internal message number of the Ith message.

MSG(2,I) - if |MSG(1,I)| \neq 30, MSG(2,I) is a GINØ file number.
If |MSG(1,I)| = 30, then MSG(2,I) is an internal message number and
USRMSG is called.

MSG(3,I), MSG(4,I) = parameters for the Ith message.

3.4.26.4 Method

The internal message number, M(1,I), if not equal to 30 in absolute value, is used by MSGWRT to print out the error message along with external message number, which is 3000 plus the internal message number. If the internal message number, M(1,I), is 30, subroutine USRMSG is called.

3.4.26.5 Design Requirements

External message numbers output by MSGWRT at present are 3001 through 3057.

MSGWRT is called only by MESSAGE.

UTILITY SUBROUTINE DESCRIPTIONS

3.4.27 USRMSG (User Message Writer).

3.4.27.1 Entry Point: USRMSG.

3.4.27.2 Purpose

To print most NASTRAN user error messages on the system output file.

3.4.27.3 Calling Sequence

CALL USRMSG(I)

COMMON/MSGX/N,M,MSG(4,10)

where:

I - Pointer into the MSG array.

N - Not used in USRMSG.

M - Not used in USRMSG.

MSG(1,I) - If |MSG(1,I)| = 30, MSGWRT will call USRMSG.

MSG(2,I) - Used by USRMSG as the internal message number.

MSG(3,I), MSG(4,I) - Parameters for the Ith message.

3.4.27.4 Method

USRMSG will print appropriate error message along with external message number, which is 2000 plus internal message number.

3.4.27.5 Design Requirements

External message numbers output by USRMSG at present are: 2001--2140.

USRMSG is called only by MSGWRT.

SUBROUTINE DESCRIPTIONS

3.4.28 MATDUM (Matrix Dump (Print) Routine).

3.4.28.1 Entry Point: MATDUM.

3.4.28.2 Purpose

To print a general NASTRAN matrix.

3.4.28.3 Calling Sequence

CALL MATDUM(FILEA)

FILEA - Seven-word array (matrix control block) - integer

Word

- 1 GINØ name
- 2 Number of columns
- 3 Number of rows
- 4 Form of matrix
- 5 Type of matrix
- 6 Maximum number of non-zero terms in any column
- 7 Undefined

3.4.28.4 Method

The non-zero terms of each column are unpacked and printed.

If the matrix control block does not contain legal values the table printer (see section 3.4.29) is called.

3.4.28.5 Design Requirements

Open core at /TABPRX/.

MATDUM must hold the non-zero band of the matrix in this area.

Subroutine TABPRT and the FØRTRAN I/Ø routines must be available to MATDUM.

UTILITY SUBROUTINE DESCRIPTIONS

.4.29 TABPRT (Table Printer).

3.4.29.1 Entry Point: TABPRT.

3.4.29.2 Purpose

To print any NASTRAN Data Block (especially tables).

3.4.29.3 Calling Sequence

CALL TABPRT(FILEN)

FILEN - GINØ name of data block - integer - input.

3.4.29.4 Method

Each word is read, identified as to type -- integer, BCD, or real number and printed 10 characters per word, 10 numbers per line. Note that the identification method varies from machine to machine and is not 100% certain, i.e., certain words may be misidentified.

3.4.29.5 Design Requirements

Open core at /TABPRX/.

Double precision numbers will not be correctly interpreted on the Univac 1108.

SUBROUTINE DESCRIPTIONS

3.4.30 PRELØC (Position Data Block to Requested Record).

3.4.30.1 Entry Points: PRELØC, LØCATE.

3.4.30.2 Purpose

To provide a convenient means of locating data records in data blocks output by the Input File Processor (IFP).

3.4.30.3 Calling Sequence

CALL PRELØC(\$n,BUFF,NAME)

n - FØRTRAN statement number defining return taken in the event NAME is not in the FIST (i.e., data block is purged).

BUFF - An array whose dimension equals the contents of the first word of /SYSTEM/ plus one. Used as a GINØ buffer by PRELØC and LØCATE.

NAME - GINØ file name of data block to be read (integer).

CALL LØCATE(\$n,BUFF,ID,IDX)

n - FØRTRAN statement number defining return taken in the event that the requested record (defined by ID) is not present in the data block.

BUFF - The same BUFF assigned when PRELØC was called.

ID - The address of a two-word array. The first word is the integer record identification and the second word is the bit position in the trailer for the data block where the presence or absence of the record is defined.

IDX - The contents of the third word of the record found will be stored in IDX (internal card number generated by IFP).

Notes:

1. If the data block is not purged, PRELØC will open the file and skip the header record.
2. If the requested record is not present (as determined by the appropriate trailer bit), no I/Ø activity will occur. Otherwise, LØCATE will position the file to read the first data entry of the requested record (i.e., after the 3-word header for the record). See 2.3.2 for format of records and trailer.

UTILITY SUBROUTINE DESCRIPTIONS

3. If the user does not read all data in a record and he wishes to use LØCATE to find another record, he should use FWDREC to skip the remainder of the current record prior to calling LØCATE.

4. For optimum efficiency in processing a data block, the user should call LØCATE in the order in which the records appear on the data block, i.e. NASTRAN collating order.

3.4.30.4 Method

PRELØC stores NAME in BUFF(1) and then calls ØPEN using BUFF(2) as the buffer address. If the data block is purged, the non-standard return is given to the user. Otherwise, FWDREC is called to skip the header record and return is made to the user. LØCATE calls RDTRL to read the data block trailer. The bit position identified by ID(2) is tested using ANDF. If zero, the non-standard return is given. Otherwise, three words from the file are read. If the first word equals ID(1), IDX is set to the third word and return is made. Otherwise, the first word is saved and the remainder of the record is skipped. The first three words of each successive record are read and the test for match on first word is made until (1) an end-of-file occurs in which case the file is rewound, the header record skipped and the process is continued, (2) a match is found in which case IDX is set and return is given or (3) a match with the first record read is found in which case the record is skipped, a warning message is queued and the non-standard return is given.

3.4.30.5 Diagnostic Messages

The following messages may be issued by PRELØC:

2072

3002

3003

SUBROUTINE DESCRIPTIONS

3.4.31 SØRT (Sort a Table).

3.4.31.1 Entry Point: SØRT.

3.4.31.2 Purpose

To sort a core contained table, or to sort a logical record from a specified input file, on a specified keyword in each entry.

3.4.31.3 Calling Sequences

To sort a core contained table:

```
CALL SØRT(0,0,NWDS,KEYWD,TABLE,NTABLE)
```

NWDS - The number of words in each entry of the table. Restriction: $NWDS \leq 20$.

KEYWD - The word position within each entry on which the sort is to take place.

TABLE - Address where the table is stored.

NTABLE - Total number of words in the table (NTABLE must be an integral multiple of NWDS).

To sort a logical record:

```
COMMON/SETUP/NFILE(6),BUF
```

```
CALL SØRT(INPFL,ØUTFL,NWDS,KEYWD,BLOCK,NBLOCK)
```

NFILE - The first three words must be set by the user prior to CALL SØRT with the GINØ file names of three scratch files for use by SØRT. Upon return to the user, NFILE(6) will contain the GINØ file name of the file containing the sorted record.

BUF - If INPFL = ØUTFL, then BUF points to an area in BLOCK where a GINØ buffer is available for SØRT, i.e., BLOCK(BUF) is the buffer address.
Restriction: $BUF > NBLOCK$.

INPFL - GINØ file name of data block containing the logical record to be sorted.

ØUTFL - GINØ file name of data block where the sorted record is to be written.
If ØUTFL = 0, the sorted record will remain on NFILE(6).

UTILITY SUBROUTINE DESCRIPTIONS

NWDS - The number of words in each entry of the record. Restriction: $NWDS \leq 20$.

KEYWD - Defined as above.

BLØCK - An area in core to be used by SØRT to perform the sort phase.

NBLØCK - The number of computer words available at BLØCK.

Notes:

1. INPFL must be opened and positioned to the logical record by the user prior to entry to SØRT. The file is not closed by SØRT.
2. If ØUTFL \neq 0, this file must be opened and positioned by the user prior to entry to SØRT. The file is not closed by SØRT.
3. If INPFL = ØUTFL, the file is closed by SØRT, opened to write with rewind, and the sorted logical record is written as the first logical record on the file. The file is not closed by SØRT.
4. NFILE(6) is always closed with rewind.

3.4.31.4 Method

1. CØRE SØRT. The method used is a shuttle exchange or bubble sort which is optimum for data which is nearly in sort. The method is as follows:
 - a. The key words of two successive entries are compared. If currently in sort, the process is repeated. If not,
 - b. A search toward the beginning of the table is made to determine the position of the out-of-sort entry.
 - c. From this position, the table is shifted one entry and the out-of-sort entry is inserted at its proper position.
 - d. If the last pair of entries have not been analyzed, the process returns to step (a). Otherwise the sort is complete.
2. FILE SØRT. One GINØ buffer is allocated at the end of BLØCK and a scratch file is opened to write. As many entries as can be held in the remaining core in BLØCK are read and sorted using the algorithm above. The sorted data is written as a logical record on the scratch file. This process is repeated until all data in the input record has been read and the sorted

SUBROUTINE DESCRIPTIONS

strings written on the scratch file. If only one such sort was required, the sort is complete except for copying onto ØUTFL if requested. Otherwise, an optimum distribution of sorted records on two scratch files is computed using a Fibonacci sequence. The sorted strings are redistributed between two scratch files and the merge phase is entered. The two scratch files are read one entry at a time, merged, and new sorted entries written on a third scratch file. Note that, using the Fibonacci sequence, one of the files containing sorted strings will have a greater number of strings (records) than the other. On each pass in the merge phase, the merge occurs until the file with fewer strings is exhausted. At this point, the merged file becomes the file with the larger number of sorted strings, the previous larger file becomes the file with the fewer strings, and the previous file with fewer strings (which was exhausted) becomes the file onto which the merged strings are written. The process continues until the sort is complete. The resulting sorted record is copied onto ØUTFL if requested.

3.4.31.5 Design Requirements

The number of words per entry may not exceed 20. (A change in the dimension of the local variable TEMP may be made to relax this restriction.)

The amount of core available at BLØCK must be at least one GINØ buffer plus 2*NWDS during the core sort phase and three GINØ buffers plus 2*NWDS during the merge phase.

The core table or logical record to be sorted must contain an integral number of entries.

3.4.31.6 Diagnostic Messages

The following messages may be issued by SØRT:

3001

3002

3008

UTILITY SUBROUTINE DESCRIPTIONS

3.4.32 GMMATD (General Matrix Multiply and Transpose - Double Precision).

3.4.32.1 Entry Point: GMMATD.

3.4.32.2 Purpose

To perform any one of the following matrix operations:

$$[A] [B] = [C] \quad (1)$$

$$[A]^T [B] = [C] \quad (2)$$

$$[A] [B]^T = [C] \quad (3)$$

$$[A]^T [B]^T = [C] \quad (4)$$

$$[A] [B] + [D] = [C] \quad (5)$$

$$[A]^T [B] + [D] = [C] \quad (6)$$

$$[A] [B]^T + [D] = [C] \quad (7)$$

$$[A]^T [B]^T + [D] = [C] \quad (8)$$

where $[A]$, $[B]$, $[C]$, and $[D]$ are real double precision matrices. This routine is used for small in-core matrices, in non-NASTRAN packed format, in such modules as SMA1, SMA2, SMA3 and DSMG1.

3.4.32.3 Calling Sequence

CALL GMMATD(A,IRQWA,ICOLA,MTA,B,IRQWB,ICOLB,MTB,C)

A - A real double precision matrix of IRQWA rows and ICOLA columns stored in the singly dimensioned double precision variable A.

N.B. A must be stored by rows. For example, if

$$[A] = \begin{bmatrix} 1.0 & 4.0 \\ 2.0 & 5.0 \\ 3.0 & 6.0 \end{bmatrix},$$

then the matrix must be stored in the FORTRAN double precision array A as follows:

$$A(1) = 1.0$$

$$A(2) = 4.0$$

$$A(3) = 2.0$$

$$A(4) = 5.0$$

$$A(5) = 3.0$$

SUBROUTINE DESCRIPTIONS

$$A(6) = 6.0$$

(A is input only).

IRØWA - number or rows of [A] - input.

ICØLA - number of columns of [A] - input.

MTA - Flag used to determine if [A] is to be transposed and to determine if the output matrix, [C], is to be zeroed out; that is, to determine if a matrix product only, of the form $[A] [B] = [C]$, will be performed or if a product and (in effect) a sum, of the form $[A] [B] + [D] = [C]$, will be performed.

1. If $MTA = 0$, then [A] is not transposed and hence either Equation (1) or (3) will be performed, depending upon MTB.

If $MTA = +1$ then [A] is transposed and hence either Equation (2) or (4) will be performed, depending upon MTB.

MTA is input only.

2. If MTA is less than zero, [C] is not zeroed out. Hence the routine, in this case, computes

$$[A] [B] + [D] = [C] \text{ if } MTA = -2 \text{ and } MTB = 0.$$

$$[A] [B]^T + [D] = [C] \text{ if } MTA = -2 \text{ and } MTB = 1.$$

$$[A]^T [B] + [D] = [C] \text{ if } MTA = -1 \text{ and } MTB = 0.$$

$$[A]^T [B]^T + [D] = [C] \text{ if } MTA = -1 \text{ and } MTB = 1.$$

(see MTB definition below)

where D is a real double precision matrix of IRØWA rows and ICØLB columns if $MTA = -2$ and D is ICØLA x ICØLB if $MTA = -1$. D must be stored row-wise at the location of C by the calling program.

B - real double precision matrix, stored row-wise. See comments for A above - input.

IRØWB - the number or rows of [B] - input.

ICØLB - the number of columns of [B] - input.

MTB - Transpose flag for [B]. If $MTB = 0$, [B] is not transposed. If $MTB = 1$, [B] is transposed. Note that MTA and MTB are independent and that only MTA controls whether or not

UTILITY SUBROUTINE DESCRIPTIONS

[C] will be zeroed out. MTB is input only.

C - real double precision matrix. Input (if MTA < 0) and output.

Examples on the use of the routine:

1. If [A] is 3x3 and [B] is 3x1 and [C] = [A] [B] is desired then:

CALL GMMATD(A,3,3,0,B,3,1,0,C). [C] is 3x1.

2. If [A] is nx1 and [B] is nx1 and the dot product is desired ($[A]^T [B]$) then:

CALL GMMATD(A,N,1,1,B,N,1,0,C). [C] is 1x1, a scalar.

3. Compute $[C] = ([X] [Y])^T$ where [X] is 5x4 and [Y] is 4x7:

CALL GMMATD(Y,4,7,1,X,5,4,1,C). C is 7x5.

4. Compute $D = [A] [B]^T + [C]$ where [A], [B] and [C] are 3x3:

DO 10 I = 1, 9

10 D(I) = C(I)

CALL GMMATD(A,3,3,-2,B,3,3,1,D).

3.4.32.4 Method

The first phase of the subroutine sets up integer loop limits which are functions of the two transpose flags. If MTA is not less than zero, the C array is zeroed out. Then the classical mathematical definitions of the above matrix products are carried out.

3.4.32.5 Design Requirements

The orders of the [A] and the [B] matrices in combination with the transpose flags must define a conformable matrix product.

3.4.32.6 Diagnostic Messages

The subroutine examines the transpose flags in combination with the orders of the matrices to make sure that a conformable matrix product is defined by this input data. This test clearly is made for purposes of calling routine checkout only. No tests are made, nor can they be made, to insure that the calling routine has provided sufficient storage for arrays. If a conformable matrix product is not defined by the input arguments, fatal error message 2021 is printed.

SUBROUTINE DESCRIPTIONS

3.4.33 GMMATS (General Matrix Multiply and Transpose - Single Precision).

3.4.33.1 Entry Point: GMMATS.

3.4.33.2 Purpose

To perform any one of the following matrix operations:

$$[A] [B] = [C] \quad (1)$$

$$[A]^T [B] = [C] \quad (2)$$

$$[A] [B]^T = [C] \quad (3)$$

$$[A]^T [B]^T = [C] \quad (4)$$

$$[A] [B] + [D] = [C] \quad (5)$$

$$[A]^T [B] + [D] = [C] \quad (6)$$

$$[A] [B]^T + [D] = [C] \quad (7)$$

$$[A]^T [B]^T + [D] = [C] \quad (8)$$

where [A], [B], [D] and [C] are real single precision matrices. This routine is used for small in-core matrices in non-NASTRAN packed format in such modules as SDR2 and PLA3 and in the utility routine PREMAT.

3.4.33.3 Calling Sequence

CALL GMMATS(A,IR0WA,IC0LA,MTA,B,IR0WB,IC0LB,MTB,C)

This routine is exactly the same as subroutine GMMATD except that GMMATD operates on real double precision matrices, while GMMATS operates on real single precision matrices. See subroutine description for GMMATD (see section 3.4.32) for details on subroutine arguments, method, design requirements and diagnostic messages.

UTILITY SUBROUTINE DESCRIPTIONS

3.4.34 INVERD (Double Precision In Core Inverse Routine).

3.4.34.1 Entry Point: INVERD.

3.4.34.2 Purpose

To compute the inverse of a real double precision matrix $[A]$ and on option to solve the matrix equation $[A] [X] = [B]$. This routine is used to invert small in-core double precision matrices in non-NASTRAN packed format and is used as a utility routine in such modules as SMA1, SMA3 and DSMG1.

3.4.34.3 Calling Sequence

CALL INVERD (NDIM,A,N,B,M,DETERM,ISING,INDEX)

- NDIM - The actual row dimension of the doubly subscripted arrays A and B in the calling program - integer - input.
- A - The square matrix to be inverted. $[A]^{-1}$ upon return from INVERD is stored at A. Double precision - input and output
- N - The order of the matrix being inverted (the size of the upper left hand corner actually being inverted). $N \leq \text{NDIM}$ - integer - input.
- B - The column(s) of constants in the above equation. If $[A]$ is to be inverted, then B is a dummy argument. The solution matrix $[X]$ is returned at B. Double precision - input and output.
- M - The number of columns of constants. If $M \leq 0$, $[A]^{-1}$ is computed - integer - input.
- DETERM - The determinant of $[A]$. Double precision - output.
- ISING - Singularity indicator. If $[A]$ is non-singular, ISING is set to 1; if $[A]$ is singular, ISING is set to 2 - integer - output.
- INDEX - Doubly subscripted array of row dimension N and column dimension 3 used for the row and column interchanges - integer - internal working storage.

3.4.34.4 Method

The classical Gauss-Jordan method with full row and column interchanges is used. All arithmetic operations are double precision.

SUBROUTINE DESCRIPTIONS

3.4.35 INVERS (Single Precision In Core Inverse Routine).

3.4.35.1 Entry Point: INVERS.

3.4.35.2 Purpose

To compute the inverse of a real single precision matrix [A] and on option to solve the matrix equation $[A] [X] = [B]$. This routine is used to invert small in-core single precision matrices in non-NASTRAN packed format and is used as a utility routine in such modules as SDR2.

3.4.35.3 Calling Sequence

CALL INVERS (NDIM,A,N,B,M,DETERM,ISING,INDEX)

This routine is exactly the same as subroutine INVERD except that INVERD operates on real double precision matrices, while INVERS operates on real single precision matrices. All arithmetic operations are single precision. DETERM is real single precision. See subroutine description for INVERD (see section 3.4.34) for details on subroutine arguments and method.

UTILITY SUBROUTINE DESCRIPTIONS

3.4.36 PREMAT (Material Property Utility).

3.4.36.1 Entry Points: PREMAT, MAT.

3.4.36.2 Purpose

To provide a utility routine for obtaining material properties used by structural element subroutines. The first entry point, PREMAT, is called once by a module for initialization purposes, and then MAT can be called by the module's element subroutines repeatedly to fetch required material properties.

3.4.36.3 Calling Sequence

CALL PREMAT (Z,ZZ,BFR,N1MAT,N2MAT,MPTF,DITF)

- Z - Integer array of open core given to the subroutine to store the material properties and the direct input tables - input and output.
- ZZ - Same address as Z. Used as real in this routine - input and output.
- BFR - A GINØ buffer (plus one cell) used by subroutine PRELØC as a buffer - input only.
- N1MAT - The length of open core, the Z array, given to PREMAT and MAT - integer - input only.
- N2MAT - The length of open core used by PREMAT and MAT - integer - output only.
- MPTF - GINØ file number of the Material Properties Table (IPT) data block - input only.
- DITF - GINØ file number of the Direct Input Tables data block. If DITF is negative, the routine assumes that the calling module is a Piecewise Linear Analysis module which implies material properties cannot be temperature dependent and that MATS1 cards are to be read.

PREMAT uses the 10th word of /SYSTEM/ which is the temperature set identification number for material properties chosen by the user in his Case Control Deck. PREMAT also uses /NAMES/ for various GINØ options.

SUBROUTINE DESCRIPTIONS

CALL MAT (ELEMID)

ELEMID - Integer element identification number; used only for diagnostic messages
(see below) - input and output.

COMMON/MATIN/MATID,INFLAG,TEMP,PLAARG,SINTH,COSTH

MATID - Material property identification number - integer - input.

INFLAG - Integer input flag which determines which sets of input data cards, MAT1, MAT2, or MAT3, the routine will search in order to find MATID. Also INFLAG determines in what format the output will be placed in the MATOUT common block. Currently INFLAG may assume the values 1 through 7 defined as follows:

INFLAG = 1 -- The material properties corresponding to the MATID are output in "MAT1" or isotropic material format (see /MATOUT/ below). One dimensional elements such as RØD, BAR, SHEAR etc. require isotropic materials. If the MATID is not found among all the MAT1 material cards read by PREMAT, a fatal error occurs.

INFLAG = 2 -- If INFLAG = 2, the material properties corresponding to the MATID are output in "MAT2" or anisotropic material format. Two-dimensional elements such as TRMEM, TRIA1, QDPLT, QUAD1 etc. may use isotropic or anisotropic materials. First, the routine will try to find the MATID among the MAT1 cards. If it is found among the MAT1 cards, the variables E (modulus of elasticity), ν (Poisson's ratio) and G (shear modulus) are used to construct the 3x3 symmetric matrix $[G_e]$ needed by two-dimensional elements, and the matrix is stored in /MATOUT/:

$$[G_e] = \begin{bmatrix} \frac{E}{1-\nu^2} & \frac{\nu E}{1-\nu^2} & 0 \\ \frac{\nu E}{1-\nu^2} & \frac{E}{1-\nu^2} & 0 \\ 0 & 0 & G \end{bmatrix}$$

UTILITY SUBROUTINE DESCRIPTIONS

If the MATID is not found among the MAT1 cards, the MAT2 cards are searched. If the MATID is not found among the MAT2 cards a fatal error occurs. If it is found, $[G_m]$, the 3x3 symmetric matrix input on the MAT2 card, is transformed by the matrix equation $[G_e] = [U]^T [G_m] [U]$ and $\{\alpha\} = [V] \{\alpha_m\}$, where $\{\alpha_m\}$ is the temperature expansion coefficient vector input on a MAT2 card. $[U]$ and $[V]$ are functions of $\sin \theta$ and $\cos \theta$ (see SINTH and CØSTH below).

$$[U] = \begin{bmatrix} \cos^2 \theta & \sin^2 \theta & \cos \theta \sin \theta \\ \sin^2 \theta & \cos^2 \theta & -\cos \theta \sin \theta \\ -2 \cos \theta \sin \theta & 2 \cos \theta \sin \theta & (\cos^2 \theta - \sin^2 \theta) \end{bmatrix}$$

$$[V] = \begin{bmatrix} \cos^2 \theta & \sin^2 \theta & -\cos \theta \sin \theta \\ \sin^2 \theta & \cos^2 \theta & \cos \theta \sin \theta \\ 2 \cos \theta \sin \theta & -2 \cos \theta \sin \theta & (\cos^2 \theta - \sin^2 \theta) \end{bmatrix}$$

INFLAG = 3 -- If INFLAG = 3, it implies the inverse of the symmetric 2x2 transverse shear matrix J will be stored in locations 16, 17 and 18 of /MATØUT/. There are two cases: (1) the current MATID is not equal to the most recent MATID, MATIDØ, and (2) the current MATID is equal to the most recent MATID.

1. If the current MATID is not equal to the most recent material identification number (MATIDØ), the MAT1 cards are searched. If the MATID is found among the MAT1 cards, then locations 16, 17 and 18 of /MATØUT/ are set to G, 0.0 and G respectively, where G is the shear modulus. If the MATID is not found among the MAT1 cards, the MAT2 cards are searched. If the MATID is not found among the MAT2 cards, a fatal error occurs. If it is found among the MAT2 cards, locations 16, 17 and 18 are set to zero.
2. The current MATID is equal to the most recent MATID. If INFLGØ, the most recent INFLAG is not 2, this is the same as case (1). If it is 2, then (a) if the MATID was found on a MAT1 card, locations 16, 17 and 18

SUBROUTINE DESCRIPTIONS

are set to G, 0.0 and G respectively; or (b) if the MATID was found on a MAT2 card, locations 16, 17 and 18 are set to 0.0.

INFLAG = 4 -- If INFLAG is 4, this implies that only the density of the material, $RH0$, will be returned in /MATOUT/ and this is the first location. The MATID can be either on a MAT1 or MAT2 card. If the MATID cannot be found among all MAT1 and MAT2 cards, a fatal error occurs.

INFLAG = 5 -- INFLAG = 5 is reserved for use only by module PLA1. This option determines if the MATID is such that E, the modulus of elasticity, is defined as stress dependent by MATS1 and TABLES1 cards. If it is stress dependent, INDSTR, equivalenced to the first word of /MATOUT/, is set to +1. If not stress dependent, INDSTR is set to 0. Only MAT1 cards are admissible for INFLAG = 5.

INFLAG = 6 -- INFLAG = 6 is reserved for use by modules PLA3 and PLA4. The fourth word of /MATIN/, PLAARG (see below), is strain and is used as the independent variable in a table look-up for stress, which is stored in the first word of /MATOUT/. Only MAT1 cards are searched for match the input MATID.

INFLAG = 7 -- INFLAG = 7 implies that the material properties corresponding to MATID will be output in MAT3 format. This format is a combination of both orthotropic and anisotropic material format. If the MATID is found in the MAT1 set, the data is stored in MAT3 format. If not found in the MAT1 set, then the MAT2 set is searched. If MATID is not found in the MAT2 set, then the MAT3 cards are searched. If not found here, a fatal error occurs. If it is found, $[G_M]$, the 3×3 symmetric matrix input on the MAT2 card is transformed by the matrix equation: $[G_e] = [u]^T [G_M] [u]$ and $\{\alpha\} = [u] \{\alpha_m\}$, where $\{\alpha_m\}$ is the temperature expansion coefficient vector input on a MAT2 card. $[u]$ is a function of sine and cosine.

$$[u] = \begin{bmatrix} \cos^2\theta & \sin^2\theta & \cos\theta \sin\theta \\ \sin^2\theta & \cos^2\theta & -\cos\theta \sin\theta \\ -2 \cos\theta \sin\theta & 2 \cos\theta \sin\theta & \cos^2\theta - \sin^2\theta \end{bmatrix}$$

UTILITY SUBROUTINE DESCRIPTIONS

INFLAG = 8 -- INFLAG = 8 is used only by two-dimensional element subroutines in modules PLA3 and PLA4. The fourth word of /MATIN/, PLAARG (see below), is stress (σ) and is used as the ordinate in an inverse interpolation table look-up to obtain the abscissa which is strain (ϵ).

If either: a) the ordinate is in the range of the piecewise linear function defined by the table on a TABLES1 bulk data card, or b) the ordinate is greater than the maximum (which is also the last) ordinate in the table but the slope of the line segment joining the last two points of the table is nonzero, then the second word of /MATOUT/ is set to zero and the abscissa, obtained by inverse linear interpolation or extrapolation, is stored in the first word of /MATOUT/. If either: a) the ordinate is less than the minimum (which is also the first) ordinate in the table, or b) the ordinate is greater than the maximum ordinate in the table and the slope of the line segment joining the last two points of the table is zero, then the integer "1" is stored in the second word of /MATOUT/ (and the first word of /MATOUT/ is set to zero). Only MAT1 cards are searched to match the input MATID.

TEMP - Average element temperature. Used as the independent variable in a table look-up when it is determined that a material property is temperature dependent. Not used when INFLAG = 5 or 6.

PLAARG - Element strain. Used as the independent variable in a table look-up when E, the modulus of elasticity, is defined as the first derivative of a strain-stress curve. Used only in the Piecewise Linear Analysis Rigid Format and only by modules PLA3 and PLA4.

SINTH - Sine of the material property orientation angle. Used only when INFLAG = 2 and the MATID is found among the MAT2 cards. Used to construct the [U] matrix referenced above.

COSTH - Cosine of the material property orientation angle. The comments on SINTH, above, also apply here.

COMMON/MATOUT/ - (output Common Block). Length 20 words. Depending upon the values of INFLAG, the output common block is defined variously as follows:

SUBROUTINE DESCRIPTIONS

1. MAT1 Format (INFLAG = 1)

<u>Word</u>	<u>Symbol</u>	<u>Definition</u>
1	E	Young's modulus (modulus of elasticity)
2	G	Shear Modulus
3	ν	Poisson's ratio
4	ρ	Density
5	α	Thermal expansion coefficient
6	T_0	Thermal expansion reference temperature
7	g_e	Structural element damping coefficient
8	σ_t	Stress limit for tension
9	σ_c	Stress limit for compression
10	σ_s	Stress limit for shear
11-25	-	Undefined
26	TDEP	Temperature dependent flag (logical) T - material referenced was temperature dependent F - material referenced was not temperature dependent

2. MAT2 Format (INFLAG = 2)

<u>Word</u>	<u>Symbol</u>	<u>Definition</u>
1	G11	The 3x3 symmetric material property matrix
2	G12	
3	G13	
4	G22	
5	G23	
6	G33	
7	RHOY	Density
8	ALPH1	Thermal expansion coefficient vector
9	ALPH2	
10	ALPH12	
11	T0Y	Thermal expansion reference temperature
12	GEY	Structural element damping coefficient
13	SIGTY	Stress limit for tension
14	SIGCY	Stress limit for compression

UTILITY SUBROUTINE DESCRIPTIONS

<u>Word</u>	<u>Symbol</u>	<u>Definition</u>
15	SIGSY	Stress limit for shear
16-26	-	Undefined

3. Transverse Shear Inverse Matrix (INFLAG = 3)

<u>Word</u>	<u>Symbol</u>	<u>Definition</u>
1-15	-	Unchanged
16	J11	The 2x2 symmetric inverse of the transverse shear matrix
17	J12	
18	J22	
19-26	-	Undefined

4. RHØ Only Format (INFLAG = 4)

<u>Word</u>	<u>Symbol</u>	<u>Definition</u>
1	RHØ	Density
2-26	-	Undefined

5. PLA1 Use Only (INFLAG = 5)

<u>Word</u>	<u>Symbol</u>	<u>Definition</u>
1	INDSTR	Stress dependent flag
2-26	-	Undefined

6. Stress Functional Value (INFLAG = 6)

<u>Word</u>	<u>Symbol</u>	<u>Definition</u>
1	PLAANS	Value of stress (σ) as a function of ϵ (strain)
2-26	-	Undefined

SUBROUTINE DESCRIPTIONS

7. MAT3 Format (INFLAG = 7)

<u>Word</u>	<u>Symbol</u>	<u>Definition</u>
1	EX3	Young's moduli r, θ and z directions
2	EY3	
3	EZ3	
4	NUXY3	Poisson's ratios. Coupled strain ratios in the xy, yz, and zx directions
5	NUYZ3	
6	NUZX3	
7	RH03	Mass density
8	GXY3	Shear Moduli
9	GYZ3	
10	GZX3	
11	AX3	Thermal expansion coefficient
12	AY3	
13	AZ3	
14	TREF3	Thermal expansion reference temperature
15	GE3	Structural element damping coefficient
16	G113	The 3 x 3 symmetric material property matrix
17	G123	
18	G133	
19	G223	
20	G233	
21	G333	
22	SIGTY3	Stress limit for tension
23	SIGCY3	Stress limit for compression
24	SIGSY3	Stress limit for shear
25	MATSET	MATID type (1.0, 2.0, or 3.0)

UTILITY SUBROUTINE DESCRIPTIONS

8. Strain Functional Value (INFLAG = 8)

<u>Word</u>	<u>Symbol</u>	<u>Definition</u>
1	PLAANS	Value of strain (ϵ) as an inverse function of stress (σ)
2	ICELL2	} = 0 if the input stress is in the range of the function = 1 if the input stress is outside the range of the function
3-26		
		Undefined

3.4.36.4 Method

1. **PREMAT:** All the MAT1, MAT2 and MAT3 cards are read from the MPT data block into open core so that each card is assigned $1 + 3*N$ words of core where N, a function of the card type, is the number of material property data items on that card type. The first word is the material identification number and each material property is allocated 3 words: the first the input material property; the second a table (function) number which gives this material property as a function of temperature; the third a table number which gives this material property as a function of stress. Initially words 2 and 3 are set to zero. Although the third word is currently used only for MAT1 cards and for E, the modulus of elasticity, on that card, future development may make use of a more general application of stress dependent material properties. If there are no temperature dependent material properties for a non-Piecewise Linear Analysis problem, PREMAT is wrapped up and a RETURN to the calling routine is executed.

For a non-Piecewise Linear Analysis problem for which a temperature set for material properties was selected in the user's Case Control Deck, all MATT1, MATT2 and MATT3 cards are read into open core from the MPT data block. For a Piecewise Linear Analysis problem MATS1 cards are read into open core from the MPT. A sorted list, with duplicates discarded, of the table numbers referenced on these cards is constructed in open core. This table number list is constructed so that every referenced table has eleven locations allocated to it. These eleven locations are used as a dictionary for the tables. The contents are: the table number (word 1); the table type 1, 2, or 4 (word 2); pointers to the first and last entries in the table (words 3 and 4); parameters from the TABLE card (words 5 through 11). The DIT data block is then read. For each table read, it is determined by scanning the table number list whether or not the table is required for problem solution. If it is required, the table is read into open core and the dictionary entry for

UTILITY SUBROUTINE DESCRIPTIONS

the table is completed. For a required table which is a type 4 (polynomial) table, the functional values of the polynomial at the end points of the interval of the real line over which the polynomial is defined are calculated by an "internal subroutine" and stored in the table dictionary. If the table is not required, it is read until an end-of-file indicator is sensed. This process continues until all tables of the set TABLEM1, TABLEM2, TABLEM3 and TABLEM4 or of the set TABLES1, are exhausted. When all referenced tables have been read into core, PREMAT is wrapped up and a return is generated.

2. MAT: The basic logic of the MAT routine is straightforward. Eight types of table look-ups, described above for INFLAG = 1, 2, 3, 4, 5, 6, 7 and 8 are supported. A computed-go-to on INFLAG is executed and each option is carried out as described above. "Internal subroutines" which are entered via FORTRAN ASSIGN and GO TO statements and return to their correct "calling" locations via ASSIGNED GO TO's are used liberally by MAT. It should be noted that each time MAT is called, MATID, INFLAG and other applicable input items, are saved. On the next call if the input is identical with the input of the previous call, nothing is stored in /MATOUT/. Hence, the calling routine should use /MATOUT/ as a "read-only data set".

3.4.36.5 Design Requirements

Subroutine GMMATS is the only non-root segment subroutine used by this routine. There are no other special requirements.

3.4.36.6 Diagnostic Messages

The following messages can be output via PREMAT and/or MAT: 3008, 2017, 2018, 2019, 2041, 2042, 2103, 2112, 2113, 2114, 2115, 2116, and 2117.

3.4.37 PRETRD (Utility for Modules Which Use the CSTM Data Block - Double Precision Version).

3.4.37.1 Entry Points: PRETRD, TRANSD

3.4.37.2 Purpose

A utility routine for modules which use the CSTM (Coordinate System Transformation Matrices) data block, TRANSD generates a real double precision 3x3 direction cosine matrix which maps a vector from a local coordinate system to basic coordinates. PRETRD sets up eventual calls to TRANSD. For a module to use TRANSD a call to PRETRD is made once and only once.

3.4.37.3 Calling Sequence

CALL PRETRD(CSTM,NCSTM)

CSTM = array of coordinate system transformation matrices (see data block description for CSTM, section 2.3) - mixed - input.

NCSTM = length of the CSTM array. NCSTM = 14*the number of coordinate systems in the CSTM data block - integer - input.

CALL TRANSD(ECPT,TA)

ECPT = array of length 4. The first word is an integer coordinate system identification number and the next 3 words are the components of a vector in basic coordinates - input only.

TA = real double precision 3x3 direction cosine matrix which maps a vector from the local coordinate system designated by ECPT(1) to basic coordinates - output.

3.4.37.4 Method

The CSTM array is searched to find a coordinate system transformation identification number that matches ECPT(1). If the coordinate system is rectangular, the 3x3 matrix, call it T, which is in words 6 through 14 of the CSTM blocks, is stored in TA and a RETURN is generated. If the coordinate system is basic, the identity matrix is returned. If the coordinate system is spherical or cylindrical, the [T] matrix defines the rectangular system from which the angles are defined. In these cases calculate:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = [T]^T \begin{pmatrix} E - V \end{pmatrix}$$

where E is the input vector stored at ECPT(2) and V is the translation offset vector in basic

UTILITY SUBROUTINE DESCRIPTIONS

coordinates found in the CSTM block in words 3, 4 and 5; and

$$r = \sqrt{x^2 + y^2}$$

If the coordinate system is cylindrical define:

$$\begin{bmatrix} T_{\ell} \end{bmatrix} = \begin{bmatrix} x/r & -y/r & 0 \\ y/r & x/r & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

If the coordinate system is spherical define:

$$\ell = \sqrt{x^2 + y^2 + z^2},$$

$$\begin{bmatrix} T_{\ell} \end{bmatrix} = \begin{bmatrix} x/\ell & \frac{xz}{r\ell} & -y/r \\ y/\ell & \frac{yz}{r\ell} & x/r \\ z/\ell & -r\ell & 0.0 \end{bmatrix}$$

Then $[T_A] = [T][T_{\ell}]$ is computed and the subroutine returns to the calling program.

3.4.37.5 Design Requirements

The routine is designed so that a module which uses the CSTM data block can have a utility routine to fetch a coordinate system transformation matrix. Typically, a module driver will attempt to open the file which contains the CSTM data block. If the data block is not purged, the module will read the entire data block into open core, close the file and call PRETRD to transmit the address of the array and the length of the array. Once this initialization call has been made, TRANSD may be called in the module as many times as necessary. The routine does not perform any I/O operations. The routine assumes the format of the CSTM data block, as outlined in the Data Block Description for the CSTM (section 2.3 of the Programmer's Manual) is correct, and no numerical checks are made.

3.4.37.6 Diagnostic Messages

If the coordinate system identification number transmitted via ECPT(1) can not be found in the CSTM array user fatal message 2025 occurs. The user should check coordinate system numbers on GRID bulk data cards against those defined on CØRD1C, CØRD1R, etc., bulk data cards to insure that there are no undefined coordinate systems.

SUBROUTINE DESCRIPTIONS

3.4.38 PRETRS (Utility for Modules Which Use the CSTM Data Block - Single Precision Version).

3.4.38.1 Entry Points: PRETRS, TRANSS.

3.4.38.2 Purpose

A utility routine for modules which use the CSTM (Coordinate System Transformation Matrices) data block, TRANSS generates a real single precision 3x3 direction cosine matrix which maps a vector from a local coordinate system to basic coordinates. PRETRS sets up eventual calls to TRANSS. For a module to use TRANSS a call to PRETRS is made once and only once.

3.4.38.3 Calling Sequence

CALL PRETRS(CSTM,NCSTM)

CALL TRANSS(ECPT,TA)

This routine is exactly the same as subroutine PRETRD (see section 3.4.37) and TRANSD except that TRANSD, an entry point, returns a real double precision matrix TA and uses double precision arithmetic, while TRANSS returns a real single precision matrix TA and uses single precision arithmetic. See subroutine description for PRETRD for details on subroutine arguments, method, design requirements and diagnostic messages.

UTILITY SUBROUTINE DESCRIPTIONS

3.4.39 PRETAB (Table Look-up)

3.4.39.1 Entry Points: PRETAB, TAB, TAB1

3.4.39.2 Purpose

To read tables (functions) from the data block DIT, Direct Input Tables, into core and to set up table dictionaries which are subsequently used when the calling routine requests a functional value from a table via a call to the entry point TAB or TAB1. The routine is designed so that PRETAB is called once and only once by a module and so that TAB and TAB1 may be called many times as a table look-up routine.

3.4.39.3 Calling Sequence

CALL PRETAB(DITF,Z,IZ,BUF,LCRGVN,LCUSED,TABNØL,LIST)

- DITF - GINØ file number of the Direct Input Tables data block - integer-input.
- Z - Array of core given to the subroutine as working storage - real-input/output.
- IZ - Same address as Z. Used as integer in this routine.
- BUF - A GINØ buffer (plus one word) used by subroutine PRELØC - input.
- LCRGVN - The length of Z array, given to PRETAB and TAB - integer-input.
- LCUSED - The number of words of core used by PRETAB - integer-output.
- TABNØL - List of table numbers that the calling routine will be referencing via TAB calls. TABNØL(1) = N is the number of tables to be referenced. TABNØL(2), ..., TABNØL(N+1) contain the table numbers. Note that 0 is an admissible table number. Table 0 defines a function which is identically zero for all values of the independent variable - integer-input.
- LIST - Array of control words for subroutine LØCATE and table types. LIST(1)=M is the number of triples which follow in the list. The first two words of each triple are the subroutine LØCATE control words for the particular table being referenced and the third word is the table type: 1, 2, 3 or 4 - integer-input.

Note: If DITF is empty, PRETAB will exit with a GINØ fatal message.

CALL TAB(TABID,X,Y)

- TABID - Table number - integer-input..
- X - Abscissa for table number TABID at which the functional value is desired - real-input.
- Y - Functional value (ordinate) of abscissa X for table number TABID - real-output.

SUBROUTINE DESCRIPTIONS

CALL TAB1(TABID,X,Z)

TABID - Table number - integer-input.

X - Abscissa for table number TABID at which functional value is desired - real-input.

Z(1) - Functional value (ordinate) of abscissa X for table TABID(Real part) - real-output.

Z(2) - Imaginary part - real-output.

3.4.39.4 Method

PRETAB: For each table in the TABNØL list an 11 word table dictionary entry is defined in open core. The first word in each entry is the table number obtained from the TABNØL list. Then the DIT data block is read. For each entry of the DIT, it is determined whether or not this table number is in the TABNØL list. If it is not, then the table is read serially until an end-of-table indicator is sensed. If it is a table called for in the TABNØL list, the program sets words 2 and 3 of the table dictionary, the table type (1, 2, 3, 4 or 5) and the pointer to the first entry in the table respectively. The table is then read into core, and the fourth word of the table dictionary, the pointer to the last entry in the table, is set. Words 5 through 11 of the dictionary, the table parameters, are set. If the table type is 4, indicating a polynomial, the functional values of the polynomial at the end points of the interval of the real line over which the polynomial is defined are calculated. After the tables for an entry in the LIST array have been exhausted, a check is made to determine if all tables in the TABNØL list have been found (after each table is found the table number is set negative). If all tables have been found, the table numbers in TABNØL are set to their original positive status and the routine is wrapped up. If all tables have not been found, the next class of table cards, defined by the next triple in the LIST array, are located in the DIT data block and the process is repeated.

TAB: The table dictionary is searched until a match is found with the input argument TABID. The table type (1, 2, 3, 4, or 5) is determined, the (functional) argument is computed after a 4-way branch on table type, and a transfer is made to either the "internal subroutine" which performs linear interpolation--if the table type is 1, 2, or 3--or the "internal subroutine" which performs polynomial evaluation--if the table type is 4. A table ID of zero returns a y value of zero.

Table 5 (TABRNDG) defines

$$Sq(f) = 2(WG)^2 * (LU) \frac{1 + 2(p+1)k^2 LU^2 (2\pi f)^2}{[1 + k^2 (LU)^2 (2\pi f)^2]^{p+3/2}} \quad (1)$$

UTILITY SUBROUTINE DESCRIPTIONS

where

<u>Type</u>	<u>p</u>	<u>k</u>
1	1/3	1.339
2	1/2	1.0

and TYPE, LU, and WG are set by the TABRNDG card.

TAB1: Basically transforms a time function to a frequency function via a FOURIER transform. For TABLED1, 2, and 3 cards, $Y(t)$ transforms to $Z(\omega)$ by the following equations:

Let $Y(t)$ be a piecewise linear table for the $(N-1)$ intervals $(X_1, X_2) \dots (X_{n-1}, X_n)$.

Then,

$$Z(\omega) = X2 \sum_{i=1}^{n-1} \frac{X_{i+1} - X_i}{2} (L_i Y_i + R_i Y_{i+1}) \quad (2)$$

where

$$L_i = e^{-i\omega(X1 + X2 X_i)} E_2(-i\omega X2(X_{i+1} - X_i))$$

$$R_i = e^{-i\omega(X1 + X2 X_{i+1})} E_2(i\omega X2(X_{i+1} - X_i))$$

$E_2(i\theta)$ is given in module IFT. The formulas for the TABLED4 cards are given in Section 4.61, Equations 19-23.

3.4.39.5 Subroutines Called

TAB1 calls IFTE2 (see Section 4.139.8.1).

3.4.39.6 Design Requirements

DIT must not be purged. Enough open core must be made available to construct the table dictionaries and to contain all referenced tables in core. All table numbers must be unique. All table numbers input via the TABNØL array must be found in the DIT data block. A table number referenced by the TABID argument of the TAB must have been referenced previously in the TABNØL array.

3.5.39.7 Diagnostic Messages

The following diagnostic messages may appear: 3008, 2088, 2089, 2090.

SUBROUTINE DESCRIPTIONS

3.4.40 AXIS (Draw an Axis on a Plot).

3.4.40.1 Entry Point: AXIS.

3.4.40.2 Purpose

To draw an x or y axis on a plotter.

3.4.40.3 Calling Sequence

CALL AXIS(X1,Y1,X2,Y2,PEN,ØPT)

CØMMØN/PLTDAT/ - see PLTDAT Miscellaneous Table description, section 2.5.

where:

X1,Y1 = starting point of the axis line - real - input.

X2,Y2 = terminal point of the axis line - real - input.

PEN = pen number or line density to be used (its meaning depends on the plotter)
- integer - input.

ØPT = $\left\{ \begin{array}{l} -1 \text{ to initiate the line mode.} \\ +1 \text{ to terminate a series of plot commands.} \\ 0 \text{ to draw an axis.} \end{array} \right\}$ - integer - input

/PLTDAT/

MØDEL = plotter model number - integer - input.

PLØTER = plotter number (i) - integer - input.

NPENS = largest number of pens or maximum density for plotter i - integer - input.

3.4.40.4 Method

This subroutine calls LINE or AXISi, depending on whether the plotter has available a single command used for drawing an axis. At this writing, only plotters 3, 10 and 11 have a special axis command.

If ØPT ≠ 0, all other arguments are ignored, and LINE or AXISi is called. Otherwise, alternate pen number (PENX) is calculated modulo NPENS and is used as the pen number passed to

UTILITY SUBROUTINE DESCRIPTIONS

LINE or AXISi, as follows:

$$\text{PENX} = \text{PEN} - \text{NPENS} * ((\text{PEN}-1)/\text{NPENS})$$

3.4.40.5 Design Requirements

Generally, AXIS or LINE should be called with $\emptyset\text{PT} = -1$ before axes are generated, even though it is not necessary to specifically put all plotters in the line mode (e.g., plotter 3). Once this is done, it need not be repeated unless the plotter has been put into some other mode (e.g., the typing mode).

Subroutines used: LINE, AXISi.

SUBROUTINE DESCRIPTIONS

3.4.41 AXISi (Axis Routine for Plotter i).

3.4.41.1 Entry Point: AXISi.

3.4.41.2 Purpose

To set up a plot command to draw an x or y axis on plotter i.

3.4.41.3 Calling Sequence

CALL AXISi(X1,Y1,X2,Y2,PEN,ØPT)

COMMON/PLTDAT/ - see PLTDAT Miscellaneous Table description, section 2.5.

where:

X1,Y1 = starting point of the axis line - real - input.

X2,Y2 = terminal point of the axis line - real - input.

PEN = pen number or line density to be used (meaning depends on plotter) - integer - input.

ØPT = $\left\{ \begin{array}{l} -1 \text{ to initiate the line mode} \\ +1 \text{ to terminate a series of plot commands} \\ 0 \text{ to draw an axis} \end{array} \right\}$ -integer - input.

/PLTDAT/

XYMIN = minimum x and y values of the region permitted on plotter i - real - input.

XYMAX = maximum x and y values of the region permitted on plotter i - real - input.

ØRIGIN = location of the lower left corner of the plotter relative to its true physical origin - real - input.

3.4.41.4 Method

Taking into account the true origin of the plotter, the plot command is generated.

UTILITY SUBROUTINE DESCRIPTIONS

3.4.41.5 Design Requirements

Subroutine used: WPLTi.

UTILITY SUBROUTINE DESCRIPTIONS

3.4.42 SKPFRM (Skip a Variable Number of Frames).

3.4.42.1 Entry Point: SKPFRM.

3.4.42.2 Purpose

To skip a variable number of frames, if appropriate to the plotter.

3.4.42.3 Calling Sequence

CALL SKPFRM (BFRAMS)

COMMON/PLTDAT/ - see PLTDAT Miscellaneous Table description, section 2.5.

where:

BFRAMS = number of frames to be skipped - integer - input.

/PLTDAT/

MODEL = plotter model number - integer - input.

PLPLOT = plotter number - integer - input.

REG = plot region parameters - real - input.

AXYMAX = size of the paper (x,y) used, less the borders, in plotter units - real - input.

EDGE = size of the borders (x,y) in plotter units - real - input.

CAMERA = currently active camera - integer - input.

ORIGIN = location (x,y) of the lower left corner of the plotter relative to its true physical origin - real - input.

3.4.42.4 Method

For plotter 3, the specified number of frames (BFRAMS) is skipped. For plotters 4 to 7 the remainder of the current plot is skipped, and another half plot is also skipped leaving the pen at the lower left of the frame.

UTILITY SUBROUTINE DESCRIPTIONS

3.4.42.5 Design Requirements

Subroutines used: LINE, WPLTi.

SUBROUTINE DESCRIPTIONS

3.4.43 SELCAM (To Initiate a New Plot).

3.4.43.1 Entry Point: SELCAM.

3.4.43.2 Purpose

To select a camera and/or to generate a setup record for a new plot.

3.4.43.3 Calling Sequence

CALL SELCAM (CAMERA,PLTNUM,ØPT)

COMMON/PLTDAT/ - see PLTDAT Miscellaneous Table description, section 2.5.

where:

CAMERA = camera number to be selected (if appropriate) - integer - input.

PLTNUM = plot number - integer - input.

ØPT ≠ 0 = if the camera is to be selected when appropriate, and nothing is to be done when not appropriate - integer - input.

/PLTDAT/

MØDEL = plotter model number - integer - input.

PLØTER = plotter number - integer - input.

XYMAX = size of the paper (x,y) used, less the borders, in plotter units - real - input.

EDGE = size of the borders (x,y) in plotter units - real - input.

CAMNUM = last selected camera - integer - output.

ØRIGIN = location (x,y) of the lower left corner of the plotter relative to its true physical origin - real - input and output.

3.4.43.4 Method

If ØPT ≠ 0 and a camera is not appropriate to the plotter, nothing is done by this subroutine.

SUBROUTINE DESCRIPTIONS

Otherwise, what is done is dependent upon the plotter hardware requirements.

For plotters 4 to 7, a block address record with the plot number is generated.

3.4.43.5 Design Requirements

Subroutines used: SPLTi, LINE.

SUBROUTINE DESCRIPTIONS

THIS PAGE HAS BEEN LEFT BLANK INTENTIONALLY.

3.4.44 IDPLØT (Generate an "ID" Plot).

3.4.44.1 Entry Point: IDPLØT.

3.4.44.2 Purpose

To identify the owner of all the plots by printing the information contained on the PLØTID card in the user's Case Control Deck prior to generating the first plot.

3.4.44.3 Calling Sequence

CALL IDPLØT (IDX)

CØMMØN/ØUTPUT/SKIP(32,6),ID(32)

CØMMØN/PLTDAT/ - see PLTDAT Miscellaneous Table description, section 2.5.

where:

$$\text{IDX} = \begin{cases} 0 & \text{if a plot id was not generated} \\ 1 & \text{if a plot id was generated} \end{cases} \quad \text{- integer - output.}$$

/ØUTPUT/

ID = user supplied PLØTID, in the Case Control Deck - BCD - input.

/PLTDAT/

XYMIN

= {plot region parameters - real - input.

XYMAX

AXYMAX = size of the paper (x,y) used, less the borders, in plotter units - real
- input.

EDGE = size of the borders (x,y) in plotter units - real - input.

CNTX,CNTY= number of counts per printed character in the x and y directions respectively
- real - input.

PLTYPE = plotter type - integer - input.

UTILITY SUBROUTINE DESCRIPTIONS

3.4.44.4 Method

If there is no PLØTID (ID = blanks), IDX is set to zero and no identification is generated. Otherwise, IDX is set to one and an identification is generated. The current region parameters are saved (they will be restored at the end of the subroutine) and are set to include the entire paper area. The identification generated varies, depending upon the plotter type.

If the plotter is a microfilm plotter ($|PLTYPE| = 1$), an entire frame is generated as identification. The top and bottom of the frame are a series of closely spaced horizontal lines. The PLØTID is then printed six times in the center of the frame.

If the plotter is a drum or table plotter ($|PLTYPE| \neq 1$), the identification is printed once at the very bottom of the paper within the bottom border.

After the identification is generated, the PLØTID is set to blanks. This insures that the identification will be generated prior to the first plot only.

3.4.44.5 Design Requirements

Subroutines used: AXIS, PRINT.

SUBROUTINE DESCRIPTIONS

3.4.45 INTGPX (Search a List of Integers).

3.4.45.1 Entry Points: INTGPX, INTGPT.

3.4.45.2 Purpose

Given a list of N integers, to find the index of the list item equal to ITEM (primarily used to search a list of external grid point id's).

3.4.45.3 Calling Sequence

```
CALL INTGPX(LIST,N)
```

```
K = INTGPT (ITEM)
```

where:

LIST = list of N integers, in arbitrary order - input.

N = number of entries in LIST - integer - input.

ITEM = integer for which a match is to be found in LIST - input.

3.4.45.4 Method

Search LIST using a linear search until a match for ITEM is found. Then the result (INTGPT) is set equal to the index of LIST where the match occurs. If no match is found, the result is set = 0.

3.4.45.5 Design Requirements

INTGPX must be called before INTGPT is used. As long as LIST does not change location and the value of N does not change, INTGPX need not be called again.

UTILITY SUBROUTINE DESCRIPTIONS

3.4.46 INTLST (Interpret a List of Integers).

3.4.46.1 Entry Point: INTLST.

3.4.46.2 Purpose

To interpret a list of integers and/or pairs of integers separated by the word TØ or THRU.

3.4.46.3 Calling Sequence

CALL INTLST(LIST,N,SIGN,N1,N2)

where:

LIST - the list to be interpreted - integer - input.

N - index location of the next list item(s) to be interpreted - integer - input.

SIGN - sign (± 1) of the interpreted integer or the first of a pair of integers - output.

N1 - absolute value of the interpreted integer or the first of a pair of integers - output.

N2 - absolute value of the second integer of pair of integers (= N1 if not a pair) - output.

3.4.46.4 Method

SIGN = ± 1 if LIST(N) is positive or negative.

N1 = absolute value of LIST(N).

If LIST(N+1) \neq TØ or THRU, then N2 = N1 and N is incremented by 1.

If LIST(N+1) = TØ or THRU, then N2 = absolute value of LIST(N+2) and N is incremented by 3.

3.4.46.5 Design Requirements

Initially, N must be set to the index of the first integer or integer pair to be interpreted in LIST. If the list is consecutive, N need not subsequently be altered until a new list is to be interpreted. It is advisable that the value following the last item in LIST be set = 0 to avoid the chance that it may equal TØ or THRU.

3.4.47 LINE (Draw a Line on a Plotter).

3.4.47.1 Entry Point: LINE.

3.4.47.2 Purpose

To draw a line on a plotter.

3.4.47.3 Calling Sequence

CALL LINE(X1,Y1,X2,Y2,PEN,ØPT)

COMMON/PLTDAT/ - see PLTDAT Miscellaneous Table description, section 2.5.

where:

X1,Y1 = starting point of the line - real - input.

X2,Y2 = terminal point of the line - real - input.

PEN = pen number or line density to be used - integer - input.

$$\text{ØPT} = \left. \begin{array}{l} -1 \text{ to initiate the line mode.} \\ +1 \text{ to terminate a series of plot commands.} \\ 0 \text{ to draw a line.} \end{array} \right\} \text{integer - input.}$$

/PLTDAT/

MØDEL = plotter model number - integer - input.

PLØTER = plotter number (i) - integer - input.

REG = x and y values defining the region in which the line is to be drawn - real - input.

NPENS = maximum number of pens or line density possible for plotter i - integer - input.

3.4.47.4 Method

If the line to be drawn is entirely outside the specified region, the subroutine immediately returns without drawing anything. If only part of the line is outside the region, only that portion of the line within the region is drawn. The actual pen number or line density used will be modulo the maximum number of pens or line density as follows:

$$\text{PENX} = \text{PEN} - \text{NPENS} * ((\text{PEN}-1)/\text{NPENS})$$

Then LINEi is called.

UTILITY SUBROUTINE DESCRIPTIONS

3.4.47.5 Design Requirements

Generally, LINE should be called with $\emptyset PT = -1$ before any lines are drawn, even though it is not necessary to specifically put all plotters in the line mode (e.g., plotter 3). Once this is done, it need not be repeated unless the plotter has been put into some other mode (e.g., the typing mode). If $\emptyset PT \neq 0$, all other arguments are ignored. Subroutine used: LINEi.

3.4.48 LINEi (Draw a Line on Plotter i).

3.4.48.1 Entry Point: LINEi.

3.4.48.2 Purpose

To draw a line on plotter i.

3.4.48.3 Calling Sequence

CALL LINEi(X1,Y1,X2,Y2,PEN,ØPT)

COMMON/PLTDAT/ - see PLTDAT Miscellaneous Table description section 2.5.

where:

X1,Y1 = starting point of the line - real - input.

X2,Y2 = terminal point of the line - real - input.

PEN = pen number or line density to be used - integer - input.

$$\text{ØPT} = \left. \begin{array}{l} -1 \text{ to initiate the line mode.} \\ +1 \text{ to terminate a series of plot commands.} \\ 0 \text{ to draw a line.} \end{array} \right\} \text{-integer - input.}$$

/PLTDAT/

MØDEL = plotter model number - integer - input.

PLØTER = plotter number - integer - input.

MAXLEN = maximum length of a line segment - real - input.

ØRIGIN = x and y values of the current position of the pen (applicable only to incremental plotters) - real - input and output.

3.4.48.4 Method

If ØPT ≠ 0, all other arguments are ignored. If ØPT = -1 and if applicable for plotter i, a flag is set so that when LINEi is subsequently called with ØPT = 0, the plotter will be put into the line mode before drawing the requested line. If ØPT = +1 and if applicable for plotter i, the pen is raised. Then, no matter which plotter is being used the current sequence of plotter commands is terminated. If ØPT = 0, the line is drawn as a series of line segments, each of maximum length MAXLEN.

SUBROUTINE DESCRIPTIONS

3.4.48.6 Design Requirements

Subroutines used: WPLTi.

3.4.49 PRINT (Print a Title on a Plotter).

3.4.49.1 Entry Point: PRINT.

3.4.49.2 Purpose

To type a title on a plotter horizontally or vertically.

3.4.49.3 Calling Sequence

CALL PRINT(X,Y,XYD,CHR,N,ØPT)

CØMMØN/PLTDAT/ - see PLTDAT Miscellaneous Table description, section 2.5.

where:

X,Y - starting or ending point of the title to be typed (always left-to-right or top-to-bottom) - real - input.

XYD - $\begin{cases} +1 & \text{if X = starting or ending point of the title} \\ +2 & \text{if Y = starting or ending point of the title} \end{cases}$ - integer - input.

CHR - title to be typed (four characters/word - left adjusted followed by blanks) - BCD - input.

N - number of words in the title - integer - input.

ØPT - $\begin{cases} -1 & \text{to initiate the typing mode.} \\ +1 & \text{to terminate a series of plot commands.} \\ 0 & \text{to type a title.} \end{cases}$ - integer - input.

/PLTDAT/

CNTCHR = number of plotter counts per character in the x and y directions - real - input.

3.4.49.4 Method

If ØPT ≠ 0, all other arguments are ignored and TIPE is called. Otherwise, each character in the title (CHR) is separated and put into another array (C). This is done for each 20 words of the title (80 characters), and TIPE is then called to type these characters.

3.4.49.5 Design Requirements

Generally, one of the typing subroutines (PRINT, TIPE, TYPFLT, TYPINT, SYMBOL) should be

SUBROUTINE DESCRIPTIONS

called with $\emptyset PT = -1$ before any typing is attempted, even though it is not necessary to specifically put all plotters in the typing mode (e.g., plotter 3). Once this is done, it need not be repeated unless the plotter has been put into some other mode (e.g., the line mode).

Subroutines used: TIPE.

UTILITY SUBROUTINE DESCRIPTIONS

3.4.50 RDMØDX (Read a File Containing XRCARD Translations).

3.4.50.1 Entry Points: RDMØDX, RDMØDY, RDMØDE, RDWØRD

3.4.50.2 Purpose

To read from a file or storage a record containing the subroutine XRCARD interpretation of free field data cards (e.g., the PCDB data block).

3.4.50.3 Calling Sequence

```
CALL RDMØDX(FILE,MØDE,WØRD)
CALL RDMØDY(A,MØDE,WØRD)
CALL RDMØDE($n1,$n2,$n3,FILE,MØDE,WØRD)
CALL RDMØDE($n1,$n2,$n3,A,MØDE,WØRD)
CALL RDWØRD(FILE,MØDE,WØRD)
CALL RDWØRD(A,MØDE,WØRD)
```

where

FILE = GINØ file name which is to be read - integer - input.

MØDE = storage location into which the XRCARD mode value is to be read - integer - output.

WØRD = 2 locations into which XRCARD card items are to be read - integer - output.

A = array which is to be "read" (instead of FILE) - integer - input.

n₁ = the FØRTRAN statement number defining the return at which numeric data are interpreted (MODE < 0).

n₂ = the FØRTRAN statement number defining the return at which alphabetic data are interpreted (0 < MODE < 1,000,000).

n₃ = the FØRTRAN statement number defining the return when the end of a logical card is encountered (MODE ≥ 1,000,000).

3.4.50.4 Method

RDMØDX and RDMØDY are intialization calls for the file and core oriented options respectively.

RDMØDE determines if the following data is BCD, numeric, end-of-physical card, or end-of-logic

SUBROUTINE DESCRIPTIONS

card. An XRCARD mode word (see Section 3.4.19.4) is read into MØDE.

1. End-of-physical card (MØDE=0) - the record is terminated if FILE is being read and the first word of the next record or core location is read into MØDE and appropriate action is continued.
2. End-of-logical card (MØDE > 1,000,000). If the FILE option is being read the current record is terminated and return n_3 is executed.
3. BCD or delimiter word ($0 < \text{MØDE} < 1,000,000$) - MØDE pairs of 4-character BCD words follow. The first two of these words are read into NEXT(1) and NEXT(2).

In addition the next set of words are checked for NEXT(1) being a blank or all-bits-on. If true, MØDE is decremented by one. This is continued until either a MØDE=0 or NEXT(1) is not a blank (end-of-physical card, go to step 1) and not all-bits-on (delimiter). Note that WØRD is not set but NEXT(1) and NEXT(2) are set for a RDWØRD call and that delimiters are ignored prior to returning (n_2).

4. Integer or real word (MØDE < 0) - the next word is read into WØRD(1). If MØDE=-4 (double precision), another word is read into NEXT(2). A RDWØRD call must not follow.

RDWØRD sets WØRD(1) and WØRD(2) from NEXT(1) and NEXT(2). This call is only used when $0 < \text{MØDE} < 1,000,000$. The value of MØDE is decremented by 1. If MØDE is greater than zero, the next two BCD words are read into NEXT(1) and NEXT(2). If NEXT(1) is blank or all-bits-on, this step is repeated until either MØDE=0 or NEXT(1) is not a blank or all-bits-on.

3.4.50.5 Design Requirements

RDMØDX or RDMØDY must be called before RDMØDE and RDWØRD. As long as FILE does not change in value, and MØDE, WØRD, and A do not change locations, RDMØDX and RDMØDY need not be called again. If RDMØDX is called, FILE must be opened and properly positioned by the calling program. In addition, RDMØDE and RDWØRD cannot be called when FILE is closed. If an end-of-file or -record condition is encountered, a fatal error occurs (see subroutine FREAD).

UTILITY SUBROUTINE DESCRIPTIONS

3.4.51 SGINØ (GINØ for Unformatted Tapes).

3.4.51.1 Entry Points: SØPEN, SWRITE, SEØF, SCLØSE.

3.4.51.2 Purpose

To write unformatted BCD and binary tapes to drive NASTRAN plotters.

3.4.51.3 Calling Sequences

CALL SØPEN(\$n,PLTTP,BUFFER,LBUFF)

n -- FØRTRAN statement number defining the return if PLTTP is not available for writing.

PLTTP - GINØ file name of the plot tape. This may have two values: PLT1 - BCD plot tape; PLT2 - binary plot tape - BCD - input.

BUFFER - Array in which the plot data transmitted during SWRITE calls are stored.

LBUFF - Length of the buffer array - integer - input.

CALL SWRITE(PLTTP,DATA,LDATA,IØPT)

PLTTP - GINØ file name of the plot tape - BCD - input.

DATA - Array of plot data (1 character/word, right justified, leading zeros).

LDATA - Length of the DATA array in words - integer - input.

IØPT - $\left\{ \begin{array}{l} 0, \text{potentially more data to be transmitted in this record.} \\ 1, \text{end of record with this data transmission.} \end{array} \right\}$ integer - input.

CALL SEØF(PLTTP)

PLTTP - GINØ file name of the plot tape on which a physical EØF will be written.

CALL SCLØSE(PLTTP)

PLTTP - GINØ file name of the plot tape.

3.4.51.4 Method

SGINØ stores data in BUFFER until IØPT = 1 or BUFFER is filled. It then transmits the data to a physical tape without any control words. The data are transmitted to SGINØ 1 character (right

SUBROUTINE DESCRIPTIONS

justified, leading zeros) per word. SGINØ packs these characters into full words. SGINØ is in FØRTRAN on all machines. On the IBM 7094 it interfaces with GINØ; the Univac 1108 version uses NTRAN; the IBM S/360 uses FØRTRAN I/Ø; and the CDC 6600 use XIØRTNS. See section 5 for details.

3.4.51.5 Design Requirements

Only one of PLT1 or PLT2 may be open at one time.

SØPEN must be called before SWRITE, SEØF, or SCLØSE.

PLT1 or PLT2 must be physical tapes if they are written on.

3.4.52 STPLØT (To Initiate a New Plot or Terminate the Current Plot).

3.4.52.1 Entry Point: STPLØT.

3.4.52.2 Purpose

To initiate a new plot or terminate the current plot.

3.4.52.3 Calling Sequence

CALL STPLØT(PLTNUM)

COMMON/PLTDAT/ - see PLTDAT Miscellaneous Table description, section 2.5.

COMMON/XXPARM/ - see XXPARM Miscellaneous Table description, section 2.5.

where:

$$\text{PLTNUM} = \left\{ \begin{array}{l} \text{if nonnegative, the plot number} \\ \text{if negative, terminate the current plot} \end{array} \right\} - \text{integer} - \text{input.}$$

/PLTDAT/

MØDEL = plotter model index - integer - input.

PLØTER = plotter number - integer - input.

REG = plot region parameters - real - input.

XYMAX = size of the paper (x,y) used, less the borders, in plotter units - real - input.

PLTYPE = plotter type - integer - input.

PLTAPE = plot tape - BCD - input.

$$\text{EØF} = \left\{ \begin{array}{l} 0 \text{ if an end-of-file mark is to be written on the plot} \\ \text{tape after each plot} \\ 1 \text{ if } \underline{\text{no}} \text{ end-of-file mark is to be written on the plot} \\ \text{tape after each plot} \end{array} \right\} - \text{integer} - \text{input.}$$

/XXPARM/

CAMERA = camera number (if applicable) to be used - integer - input.

BFRAMS = number of blank frames between plots - integer - input.

UTILITY SUBROUTINE DESCRIPTIONS

3.4.52.4 Method

A. If PLTNUM is nonnegative:

1. Select the specified camera or create a setup record appropriate to the plotter (CALL SELCAM).
2. Skip to a new frame (if applicable) and create the owner identification frame. If the owner identification frame is generated by subroutine IDPLØT, re-execute step 1 and skip to a new frame.
3. If appropriate to this plotter, insert the desired number of blank frames on film only. If the camera specified is camera 2 (paper only), no blank frames are inserted.
4. If the plot number is nonzero, type this number in the upper left and right corners of the picture and the date between.

B. If PLTNUM is negative:

1. Terminate the current plot tape record.
2. Close the current plot tape file (CALL SCLØSE).
3. If each plot is to be separated by an end-of-file mark (EØF = 0), write an end-of-file on the plot tape (CALL SEØF).

3.4.52.5 Design Requirements

Subroutines used include: IDPLØT, SELCAM, SKPFRM, TYPINT, SCLØSE, SEØF, TIPE.

SUBROUTINE DESCRIPTIONS

3.4.53 SYMBOL (Type a Symbol on a Plotter).

3.4.53.1 Entry Point: SYMBOL.

3.4.53.2 Purpose

To type a symbol on a plotter.

3.4.53.3 Calling Sequence

CALL SYMBOL(X,Y,SYM,OPT)

COMMON/PLTDAT/ - see PLTDAT Miscellaneous Table description, section 2.5.

COMMON/SYMBLS/ - see SYMBLS Miscellaneous Table description, section 2.5.

where:

X,Y - point at which the symbol is to be typed - real - input.

SYM - two consecutive storage locations each containing an index into the SYMBLS table - integer - input.

OPT - $\left. \begin{array}{l} -1 \text{ to initiate the typing mode.} \\ +1 \text{ to terminate a series of plot commands.} \\ 0 \text{ to type the symbol.} \end{array} \right\} \text{ - integer - input.}$

/PLTDAT/

MODEL - plotter model number - integer - input.

PLPLOT - plotter number (i) - integer - input.

/SYMBLS/

NSYM - number of symbols defined in the SYMBLS table - integer - input.

SYMBL(20,i) - character indices defining the symbols of plotter i - integer - input.

3.4.53.4 Method

If OPT \neq 0, all other arguments are ignored and TYPEI or DRWCHR is called. Otherwise, an alternate symbol index (SYM) is calculated module NSYM for each index in SYM and is used as the actual symbol index, as follows:

UTILITY SUBROUTINE DESCRIPTIONS

$$\text{SYM}_j = \text{SYM}_j - \text{NSYM} * ((\text{SYM}_{j-1}) / \text{NSYM}) \quad , \quad j = 1, 2.$$

Then TYPEi or DRWCHR is called for each symbol.

The reason for SYM being two indices is to enable the user to create any additional symbol by combining any two of the valid symbols in the SYMBLS table. Note: any of the indices in SYM may = 0. This would imply that a new symbol is not being created.

3.4.53.5 Design Requirements

Generally, one of the typing subroutines (PRINT, TIPE, TYPFLT, TYPINT, SYMBØL) should be called with ØPT ≠ -1 before any typing is attempted, even though it is not necessary to put all plotters in the typing mode (e.g., plotter 3). Once this is done, it need not be repeated unless the plotter has been put into some other mode (e.g., the line mode).

Subroutines used: TYPEi, DRWCHR.

SUBROUTINE DESCRIPTIONS

3.4.54 TIPE (Type a Line of Characters on a Plotter).

3.4.54.1 Entry Point: TIPE.

3.4.54.2 Purpose

To type a line of characters on a plotter horizontally or vertically.

3.4.54.3 Calling Sequences

CALL TIPE (X,Y,XYD,CHR,N,ØPT)

CØMMØN/PLTDAT/ - see PLTDAT Miscellaneous Table description, section 2.5.

CØMMØN/CHAR94/ - see CHAR94 Miscellaneous Table description, section 2.5.

where:

X,Y - starting or ending point of the line to be typed (always left-to-right or top-to-bottom) - real - input.

$XYD - \left\{ \begin{array}{l} +1 \text{ if } X = \text{starting or ending point of the line.} \\ +2 \text{ if } Y = \text{starting or ending point of the line.} \end{array} \right\}$ - integer - input.

CHR - line of characters to be typed (one character/word - left adjusted followed by blanks) - BCD - input.

N - number of characters to be typed - integer - input.

$\text{ØPT} - \left\{ \begin{array}{l} -1 \text{ to initiate typing mode.} \\ +1 \text{ to terminate a series of plot commands.} \\ 0 \text{ to type a line of characters.} \end{array} \right\}$ - integer - input.

/PLTDAT/

MØDEL - plotter model number - integer - input.

PLØTER - plotter number (i) - integer - input.

CNTCHR - number of plotter counts per character in the x and y directions - real - input.

/CHAR94/

CHAR - Section I of the CHAR94 table - BCD - input.

3.4.54.4 Method

If $\emptyset PT \neq 0$, all other arguments are ignored and TYPEi or DRWCHR is called. Otherwise, for each character to be typed, an index into the CHAR character set is found. This is done 80 characters at a time. If a character cannot be located, it is treated as a blank. For each set of 80 character indices set up, TYPEi or DRWCHR is called.

3.4.54.5 Design Requirements

Generally, one of the typing subroutines (PRINT, TIPE, TYPFLT, TYPINT, SYMBOL) should be called with $\emptyset PT \neq -1$ before any typing is attempted, even though it is not necessary to put all plotters in the typing mode (e.g., plotter 3). Once this is done, it need not be repeated unless the plotter has been put into some other mode (e.g., the line mode).

Subroutines used: TYPEi, DRWCHR.

3.4.55 TYPEi (Type a Line of Characters on Plotter i).

3.4.55.1 Entry Point: TYPEi.

3.4.55.2 Purpose

To type a line of characters on plotter i horizontally or vertically.

3.4.55.3 Calling Sequence

CALL TYPEi(X,Y,XYD,CHR,N,ØPT)

CØMMØN/PLTDAT/ - see PLTDAT Miscellaneous Table description, section 2.5.

CØMMØN/CHAR94/ - see CHAR94 Miscellaneous Table description, section 2.5.

where:

X,Y = starting or ending point of the line to be typed (always right-to-left or top-to-bottom) - real - input.

$XYD = \begin{cases} +1 & \text{if } x = \text{starting or ending point of the line.} \\ +2 & \text{if } y = \text{starting or ending point of the line.} \end{cases}$ - integer - input.

CHR = indices of the line of characters to be typed (see description for TIPE, section 3.4.54) - integer - input.

N = number of characters to be typed - integer - input.

$\text{ØPT} = \begin{cases} -1 & \text{to initiate the typing mode.} \\ +1 & \text{to terminate a series of plot commands} \\ 0 & \text{to type a line of characters.} \end{cases}$ - integer - input.

/PLTDAT/

XYMIN = minimum x and y values of the region in which the line is to be typed - real - input.

XYMAX = maximum x and y values of the region in which the line is to be typed - real - input.

CNTCHR = number of plotter counts per character in the x and y directions - real - input.

UTILITY SUBROUTINE DESCRIPTIONS

/CHAR94/

CHRCOD = Section II, III, or IV of the CHAR94 table - integer - input.

3.4.55.4 Method

If $\emptyset PT \neq 0$, all other arguments are ignored. If $\emptyset PT = -1$ and if applicable for plotter i , a flag is set so that when TYPE i is subsequently called with $\emptyset PT = 0$, the plotter will be put into the typing mode before typing the first character. If $\emptyset PT = +1$, the current sequence of plotter commands is terminated.

Define:

LSTCHR = last legitimate character index for plotter i .

NCHR = number of character indices which must be changed for plotter i .

CHAR = NCHR pairs of character indices. The first index of each pair is the index which must be changed, and the second index is the replacement index.

If $N \leq 0$, it is assumed that one character is to be typed.

Each character index in CHR is checked against LSTCHR. If the index is greater than LSTCHR, a blank is inserted at the corresponding point on the plot. Otherwise, indices are altered if need be from CHAR and the character is typed.

No characters will be typed outside the region as defined by XYMIN and XYMAX.

3.4.55.5 Design Requirements

Subroutines used: WPLTi.

3.4.56 TYPFLT (Type a Floating Point Number on a Plotter).

3.4.56.1 Entry Point: TYPFLT.

3.4.56.2 Purpose

To type a floating point number on a plotter, horizontally or vertically.

3.4.56.3 Calling Sequence

CALL TYPFLT (X,Y,XYD,V,FIELD,ØPT)

COMMON/PLTDAT/ - see PLTDAT Miscellaneous Table description, section 2.5.

where:

X,Y - point at which the number is to be typed (always left-to-right or top-to-bottom)
- real - input.

XYD - $\left\{ \begin{array}{l} +1 \text{ if } X = \text{starting or ending point of the typed number.} \\ +2 \text{ if } Y = \text{starting or ending point of the typed number.} \end{array} \right\}$ - integer - input.

V - number to be typed - real - input.

FIELD - field width of the typed number. If FIELD > 0, the number will be centered at (X,Y). If FIELD < 0, the number will be typed starting or ending at (X,Y). If |XYD| = 1 or 2, the number will be typed horizontally or vertically respectively - integer - input.

ØPT - $\left\{ \begin{array}{l} -1 \text{ to initiate the typing mode.} \\ +1 \text{ to terminate a series of plot commands.} \\ 0 \text{ to type the number.} \end{array} \right\}$ - integer - input.

/PLTDAT/

MØDEL - plotter model number - integer - input.

PLØTER - plotter number (i) - integer - input.

CNTCHR - number of plotter counts per character in the x and y directions - real - input.

3.4.56.4 Method

If ØPT ≠ 0, all other arguments are ignored and TYPE1 or DRWCHR is called. Otherwise, the

number of significant digits (NSIG) to be typed is determined.

If $V \geq 0$, the typed number will be unsigned. If $FIELD > 4$, the number of significant digits typed will be at least $= FIELD - 4$. If $FIELD \leq 4$, $NSIG = FIELD - 1$.

If $V < 0$, the typed number will be signed. If $FIELD > 5$, the number of significant digits typed will be at least $FIELD - 5$. If $FIELD \leq 5$, $NSIG = FIELD - 2$.

The number (V) is multiplied by some power of ten such that the product is between 10^7 and 10^8 . It can then be expressed as an 8-significant digit integer. If the number is such that NSIG digits cannot be typed without an exponent, a standard form is used: $-X.XXXX \dots \pm XX$. Otherwise the decimal point is adjusted and the exponent will not be printed.

3.4.56.5 Design Requirements

Generally, one of the typing subroutines (PRINT, TIPE, TYPFLT, TYPINT, SYMBOL) should be called with $\emptyset PT = -1$ before any typing is attempted, even though it is not necessary to put all the plotters in the typing mode (e.g., plotter 3). Once this is done, it need not be repeated unless the plotter has been put into some other mode (e.g., the line mode).

Subroutines used: TYPEi, DRWCHR.

3.4.56.6 Diagnostic Messages

If NSIG significant digits cannot possibly be typed in the field width (FIELD) specified, the entire field will be filled with asterisks (**...*).

SUBROUTINE DESCRIPTIONS

3.4.57 TYPINT (Type an Integer Number on a Plotter).

3.4.57.1 Entry Point: TYPINT.

3.4.57.2 Purpose

To type an integer number on a plotter, horizontally or vertically.

3.4.57.3 Calling Sequence

CALL TYPINT (X,Y,XYD,NUM,FIELD,ØPT)

CØMMØN/PLTDAT/ - see PLTDAT Miscellaneous Table description, section 2.5.

where:

X,Y - point at which the number is to be typed (always left-to-right or top-to-bottom)
- real - input.

XYD $\left\{ \begin{array}{l} +1 \text{ if } X = \text{starting or ending point of the typed number.} \\ +2 \text{ if } Y = \text{starting or ending point of the typed number.} \end{array} \right\}$ - integer - input.

NUM - number to be typed - integer - input.

FIELD $\left\{ \begin{array}{l} +1 \text{ if the typed number is to be centered at } (X,Y). \text{ If } |XYD| = 1 \text{ or } 2, \text{ the number} \\ \text{will be typed horizontally or vertically, respectively.} \\ -1 \text{ or } 0, \text{ the number will be typed starting or ending at } (X,Y). \text{ If } FIELD = -1, \\ \text{FIELD will be set to the number of digits typed by the subroutine; in this} \\ \text{case, FIELD must be a symbol in the call statement.} \\ \text{- integer - input and} \\ \text{output.} \end{array} \right.$

ØPT $\left\{ \begin{array}{l} -1 \text{ to initiate the typing mode.} \\ +1 \text{ to terminate a series of plot commands.} \\ 0 \text{ to type the number.} \end{array} \right\}$ -integer - input.

/PLTDAT/

MØDEL - plotter model number - integer - input.

PLØTER - plotter number (i) - integer - input.

CNTCHR - number of plotter counts per character in the x and y directions - real - input.

3.4.57.4 Method

If $\emptyset PT \neq 0$, all other arguments are ignored and TYPEi or DRWCHR is called. Otherwise, each digit of the number is separated and used as character indices for the TYPEi or DRWCHR subroutines. In addition, if FIELD < 0, FIELD is set = the number of digits printed.

3.4.57.5 Design Requirements

Generally, one of the typing subroutines (PRINT, TIPE, TYPFLT, TYPINT, SYMBØL) should be called with $\emptyset PT = -1$ before any typing is attempted, even though it is not necessary to specifically put all plotters in the typing mode (e.g., plotter 3). Once this is done, it need not be repeated unless the plotter has been put into some other mode (e.g., the line mode).

Subroutines used: TYPEi, DRWCHR.

SUBROUTINE DESCRIPTIONS

(THIS PAGE AVAILABLE FOR FUTURE USE)

UTILITY SUBROUTINE DESCRIPTIONS

(THIS PAGE AVAILABLE FOR FUTURE USE)

SUBROUTINE DESCRIPTIONS

(THIS PAGE AVAILABLE FOR FUTURE USE)

UTILITY SUBROUTINE DESCRIPTIONS

(THIS PAGE AVAILABLE FOR FUTURE USE)

3.4.60 WPLT3 (Write a Plotter Command for Plotter 3).

3.4.60.1 Entry Point: WPLT3.

3.4.60.2 Purpose

To write a plot command for plotter 3.

3.4.60.3 Calling Sequence

CALL WPLT3 (A,ØPT)

COMMON/PLTDAT/ - see PLTDAT Miscellaneous Table description, section 2.5.

where:

A(1) and A(2) = 36 bit plot command set up by AXIS3, LINE3, or TYPE3, as 2 18-bit words
(right justified, leading zeros) - input.

$$\text{ØPT} = \begin{cases} 0, & \text{if } A = \text{plot command} \\ 1, & \text{if a series of plot commands is to be terminated} \end{cases} - \text{integer} - \text{input.}$$

/PLTDAT/

PLØT - GINØ file name of the plot tape to be written - BCD - input.

MAXCHR - plot tape buffer size (number of characters) - integer - input.

3.4.60.4 Method

Each plotter command is 36 bits long (6 six-bit characters). Six of the 36 bits in A(1) and A(2) are written on the plot tape until all 36 bits have been written. In addition, the number of six-bit characters written in a record is calculated. When WPLT3 is called with ØPT = 1, a check is made to determine if the number of 6 bit characters written in the current record is an integer multiple of the number of characters per word on the computer. If such is not the case, an additional 36 bit command is written as many times as necessary until this condition does exist before terminating the plot tape record. The command used will do nothing to affect the generated plot.

3.4.60.5 Design Requirements

Subroutine used: SWRITE.

UTILITY SUBROUTINE DESCRIPTIONS

THE MATERIAL PREVIOUSLY ON PAGE 3.4-103 HAS BEEN DELETED

SUBROUTINE DESCRIPTIONS

THE MATERIAL PREVIOUSLY ON PAGE 3.4-104 HAS BEEN DELETED

UTILITY SUBROUTINE DESCRIPTIONS

3.4.62 EJECT (Automatic Page Eject)

3.4.62.1 Entry Point: EJECT

3.4.62.2 Purpose

Automatic line counting for printed output and new page initiation when pages are filled.

3.4.62.3 Calling Sequence

K = EJECT (LINES)

COMMON /SYSTEM/ - see SYSTEM table description, section 2.4.1.8.

where:

LINES - Number of lines to be printed.

/SYSTEM/

MAXLIN - Maximum number of lines permitted per page.

LINCNT - Number of lines thus far printed on this page.

3.4.62.4 Method

If the number of lines already printed on this page (LINCNT) added to the number of lines about to be printed (LINES) would be greater than the number of lines permitted per page (MAXLIN), a new page is started (CALL PAGE1), the current line counter is set to the number of lines to be printed (LINCNT = LINES), and the result of this function is set to 1 (EJECT = 1).

If the number of lines about to be printed (LINES) will fit on this page ($LINCNT + LINES \leq MAXLIN$), the result of this function is set to 0 (EJECT = 0).

3.4.62.5 Design Requirements

If it is desired to force a new page to be started, simply set LINCNT = MAXLIN before calling this function.

3.4.63 PLAMAT (Material Property Utility for Two-Dimensional Elements in Piecewise Linear Analysis).

3.4.63.1 Entry Point: PLAMAT.

3.4.63.2 Purpose

To perform the following matrix operation:

$$[C] = [A]^T [B] [A] ,$$

where $[A]$ is equal to $[U]$ (see the subroutine description for PREMAT and MAT, section 3.4.36.3, for a definition of $[U]$ with INFLAG = 2), and $[B]$ is equal to a previously calculated material properties matrix which is in common block /PLAGP/. The result $[C]$, which is symmetric, is stored in common block /MATOUT/.

3.4.63.3 Calling Sequence

CALL PLAMAT

COMMON/PLAGP/GP(9),MIDGP,ELID

COMMON/MATIN/MATID,INFLAG,ELTEMP,PLAARG,SINTH,COSTH

COMMON/MATOUT/G11,G12,G13,G22,G23,G33,DUMMY(14)

where:

/PLAGP/

GP(9) = 3x3 material properties matrix calculated in a PLA element driver - real - input.

MIDGP = the material identification number associated with GP - integer - input.

ELID = the element identification number associated with GP - integer - input.

/MATIN/

MATID = the incoming material identification number - integer - input.

$\left. \begin{array}{l} \text{INFLAG} \\ \text{ELTEMP} \\ \text{PLAARG} \end{array} \right\} = \text{not used by PLAMAT.}$

UTILITY SUBROUTINE DESCRIPTIONS

SINTH = Sine of the material property orientation angle - real - input.

COSTH = Cosine of the material property orientation angle - real - input.

/MATOUT/

Same as /MATOUT/ with INFLAG = 2 as described in section 3.4.36.3 except only the first six cells are used.

3.4.63.4 Method

This routine checks to see if the incoming material identification number (MATID) is equal to the material identification number (MIDGP) which was used to calculate the material properties matrix stored in /PLAGP/. If they are not equal, this routine calls MAT with INFLAG = 2 and returns to the calling program. This will only happen in combination elements (TRIA1, TRIA2, QUAD1, QUAD2) where there is a different material identification number used for the membrane and plate properties. If they are equal, then the matrix operation described above is performed.

SUBROUTINE DESCRIPTIONS

3.4.64 WPLT4 (Write a Plotter Command for Plotters 4 Through 7).

3.4.64.1 Entry Point: WPLT4.

3.4.64.2 Purpose

To write plot commands for plotters 4 through 7.

3.4.64.3 Calling Sequence

CALL WPLT4 (A,ØPT)

COMMON/PLTDAT/ - see PLTDAT Miscellaneous Table description, Section 2.5.

where:

$$\begin{aligned} A(1) &= \text{command type (0 = Search Address record, 1 = line, 2 = position)} \\ A(2-N) &= \text{additional data used to generate the plot commands (contents and} \\ &\quad \text{length, N, vary with command type)} \end{aligned} \left. \vphantom{\begin{aligned} A(1) \\ A(2-N) \end{aligned}} \right\} \begin{array}{l} \text{- integer,} \\ \text{input.} \end{array}$$
$$\begin{aligned} \text{ØPT} &= \left\{ \begin{array}{l} 0, \text{ if a plot command is to be generated} \\ 1, \text{ if the current command buffer is to be terminated} \end{array} \right\} \quad \begin{array}{l} \text{- integer, input.} \end{array}$$

/PLTDAT/

PLTMDL = plotter model number - integer - input.

PLØT = GINØ file name of the plot tape to be written - BCD - input.

PBFSIZ = Plotter buffer size (number of characters).

3.4.64.4 Method

Two types of records are created depending on the value of A(1). If A(1) = 0, a Search Address record necessary to start the plotter or identify each frame is created. If A(1) > 0, a Plot Data record is created which causes pen up, down, right and left commands. If ØPT is non-zero, A(1) is ignored and the current plot buffer is terminated (if any).

The Search Address record (A(1) = 0) input consists of the appropriately coded plot number as follows:

UTILITY SUBROUTINE DESCRIPTIONS

<u>A</u>	<u>Entries</u>
1	0
2	Number of characters to output starting at A(4) (15 maximum)
3	If non zero close the Search Address record also
4	Coded "plot number" characters.

The necessary synchronization and search-address-indicators commands are automatically supplied.
The plot buffer size must be sufficient to hold the entire record.

The Plot Data records (A(1)=1 or 2), will begin with the characters necessary to lower or raise the pen, unless the pen is already in position. The data in A is used to control the movement of the pen as follows:

<u>A</u>	<u>Entries</u>
1	1 = move with pen down, 2 = move with pen up
2	Number of half steps - major
3	Number of half steps - compensation
4	Control pointer - major movement
5	Control pointer - compensation movement
6)	Same as A(4) and A(5) but for last half step (if any)
7)	

Drum movement is caused by the "control pointers" in A(4) to A(7) indexing the following commands:

1 = +Y	9 = +Y/2	17 = +X/2, +Y
2 = +X, +Y	10 = +X/2, +Y/2	18 = +X, +Y/2
3 = +X	11 = +X/2	19 = +X, -Y/2
4 = +X, -Y	12 = +X/2, -Y/2	20 = +X/2, -Y
5 = -Y	13 = -Y/2	21 = -X/2, -Y
6 = -X, -Y	14 = -X/2, -Y/2	22 = -X, -Y/2
7 = -X	15 = -X/2	23 = -X, +Y/2
8 = -X, +Y	16 = -X/2, +Y/2	24 = -X/2, +Y

SUBROUTINE DESCRIPTIONS

The number of, and values of, the control characters for each increment is a function of the internal plotter model number, PLTMDL. If the hundred's digit is 1 (e.g., 150), three characters/command are required, otherwise only one character/command is needed. The one's digit position is the number of spacers desired to account for different tape densities and tape transport switch settings. A "0" is used before each plot command for each spacer unless the hundreds digit is 4 where an octal 10 is used.

As with the Search Address records, each plot record automatically accounts for the synchronization, start-plot, end-plot and the pen-up or pen-down commands as a function of the plot buffer size, PBFSIZ.

The commands that are written on the PLT2 tape by calling SWRITE are a function of the tape density. The value is one (1) or two (2) if a 7 track or a 9 track tape respectively is mounted. The value is set in the preface by subroutine GNFIAT or NASCAR.

3.4.64.5 Design Requirements

The only incremental drum movements available for the CALCOMP drum plotter indicated as PLTMDL = 11i or 21i are the first eight (8) as listed above. Therefore, the values in A(4 - 7) must be less than nine (9). The plotter buffer size must be sufficient to hold the entire Search Address record.

Subroutine used: SWRITE.

SUBROUTINE DESCRIPTIONS

THIS PAGE HAS BEEN LEFT BLANK INTENTIONALLY.

Handwritten signature or initials

SUBROUTINE DESCRIPTIONS

(THIS PAGE AVAILABLE FOR FUTURE USE)

Handwritten marks: a horizontal line and a circular symbol.

3.4.66 WPLT10 (Write a Plotter Command for the NASTRAN General Purpose Plotter).

3.4.66.1 Entry Point: WPLT10.

3.4.66.2 Purpose

To write the plotter commands for the NASTRAN general purpose plotter.

3.4.66.3 Calling Sequence

CALL WPLT10 (A,ØPT)

COMMON/PLTDAT/ - see PLTDAT Miscellaneous Table description, section 2.5.

where:

A(1) = plot mode index	}	- integer - input.
A(2) = control index		
A(3) = x_1 = x-coordinate		
A(4) = y_1 = y-coordinate		
A(5) = x_2 = x-coordinate		
A(6) = y_2 = y-coordinate		

$$\text{ØPT} = \begin{cases} 0 & \text{if A = plot command} \\ 1 & \text{if a series of plot commands is to be terminated} \end{cases} \text{ - integer - input.}$$

/PLTDAT/

EDGE = size of the borders (x,y) in plotter units - real - input.

PLØT = GINØ file name of the plot tape to be written - BCD - input.

MAXCHR = plot tape buffer size (number of characters) - integer - input.

3.4.66.4 Method

Each plot command written is composed of 30 six-bit unsigned integers. The plot mode index, A(1), and the control index, A(2), are the first two integers. The next 20 integers represent the values in A(3-6). Each value is represented by five 6-bit integers, each integer being a decimal digit of the decimal representation of the value as follows:

SUBROUTINE DESCRIPTIONS

$$d_4 d_3 d_2 d_1 d_0$$

where the original integer value is given by

$$d_0 \cdot 10^0 + d_1 \cdot 10^1 + d_2 \cdot 10^2 + d_3 \cdot 10^3 + d_4 \cdot 10^4 .$$

This representation is used so as to make it easy to recover the original integer values on any binary computer. The last 8 characters are always zeros.

The end result is a plot command of the following format:

MCP₄P₃P₂P₁P₀Q₄Q₃Q₂Q₁Q₀R₄R₃R₂R₁R₀S₄S₃S₂S₁S₀00000000

where:

- M = plot mode index
- C = control index
- P_i = decimal digit of the 1st integer value
- Q_i = decimal digit of the 2nd integer value
- R_i = decimal digit of the 3rd integer value
- S_i = decimal digit of the 4th integer value
- 0 = zero

When WPLT10 is called with OPT = 1, the current plot tape record is filled with as many dummy plot commands as is necessary to generate a fixed length record. The dummy plot command is made of 30 zeros. This is done so that the plot tape can be read in FORTRAN without having to worry about variable length records as long as the plot tape buffer size (MAXCHR) is an integer multiple of the number of characters per word on the computer on which the plot tape is being read (see section 6.10.6 for further details).

3.4.66.5 Design Requirements

Subroutine used: SWRITE.

= 71-

3.4.67 PLTSET (Plotting Parameter Initialization).

3.4.67.1 Entry Point: PLTSET.

3.4.67.2 Purpose

Given the internal plotter and model numbers, to initialize the /XXPARAM/ and /PLTDAT/ tables as needed by the NASTRAN plotter software package.

3.4.67.3 Calling Sequence

CALL PLTSET

COMMON/XXPARAM/ - see XXPARAM Miscellaneous Table description, section 2.5.

COMMON/PLTDAT/ - see PLTDAT Miscellaneous Table description, section 2.5.

where:

/XXPARAM/

PBUFSZ = plot tape buffer size (number of words) - integer - output.

PAPSIZ = size of the paper to be used (inches) - real - input.

/PLTDAT/

MØDEL = internal plotter model number - integer - input.

PLØTER = internal plotter number - integer - input.

REG = plotting region parameters - real - output.

AXYMAX = size of the paper (x,y) used, less the borders, in plotter units - real - output.

XYEDGE = size of the borders (x,y) in plotter units - real - output.

XYMAX = maximum useable x and y coordinate values on the plotter - real - output.

CNTSIN = number of plotter counts per inch of paper - real - output.

CNTCHR = number of plotter counts per character in the x and y directions - real - output.

PLTYPE = plotter type - integer - output.

PBFSIZ = plot tape buffer size (number of characters) - integer - output.

SUBROUTINE DESCRIPTIONS

3.4.67.4 Method

Using the internal plotter (PLØTER) and model (MØDEL) numbers, the initialization needed to properly use the NASTRAN plotting software is performed as follows:

1. Section 2 of /PLTDAT/, of which XYMAX, CNTSIN, CNTCHR, PLTYPE and PBFSIZ are a part, is set to a duplicate of section PLØTER+2. If the 20th word of section 1 is nonzero, the CNTCHR entries are multiplied by it so as to scale the character size.
2. PBFSIZ of /XXPARM/ is then set to PBFSIZ/CHRWRD where CHRWRD = number of characters per word on the subject computer.
3. If the plotter is a drum plotter and the paper size is 0.0 X 0.0 the paper size is calculated as;

$$PAPSIZ(I) = XYMAX(I)/CNTSIN$$

For table plotters the paper size is set to 11 X 8.5 if either entry is 0.0.

4. AXYMAX and XYEDGE are calculated based upon the plotter type and/or paper size. If the plotter is a table plotter (PLTYPE = +2 or -2), the borders are set up as 1/2 inch borders. If the plotter is not a table plotter and has no typing capability (PLTYPE = -1 or -3), the borders are set up as half the horizontal and vertical character sizes (CNTCHR/2). Otherwise, the borders are set to zero.
5. The plotting region is then set to (0,0,AXYMAX(1),AXYMAX(2)). This region can be subsequently altered by the module writer.

3.4.68 DRWCHR (To Draw a Line of Characters).

3.4.68.1 Entry Point: DRWCHR.

3.4.68.2 Purpose

To draw a line of characters on a plotter, horizontally or vertically.

3.4.68.3 Calling Sequence

CALL DRWCHR (X,Y,XYD,CHR,NN,ØPT)

COMMON/PLTDAT/ - see PLTDAT Miscellaneous Table description, section 2.5.

COMMON/CHDRW/ - see CHDRW Miscellaneous Table description, section 2.5.

where:

X,Y = starting or ending coordinate of the line of characters to be drawn (always left-to-right or top-to-bottom) - real - input.

$XYD = \begin{cases} +1 & \text{if } X = \text{starting or ending point of the line} \\ +2 & \text{if } Y = \text{starting or ending point of the line} \end{cases}$ - integer - input.

CHR = indices of the line of characters to be drawn (see subroutine TIPE) - integer - input.

NN = number of the characters to be drawn - integer - input.

$\text{ØPT} = \begin{cases} -1 & \text{to initiate the line mode.} \\ +1 & \text{to terminate a series of plot commands.} \\ 0 & \text{to draw a line of characters.} \end{cases}$ - integer - input.

/PLTDAT/

REG = plot region parameters - real - input.

XYMAX = size of the paper (x,y) used, less the borders, in plotter units - real - input.

EDGE = size of the border (x,y) in plotter units - real - input.

CNTCHR = number of plotter counts per character in the x and y directions - real - input.

/CHDRW/

LSTIND = index of the last character which can be drawn - integer - input.

SUBROUTINE DESCRIPTIONS

CHRIND = indices into XYCHR used to locate the data needed to draw characters - integer
- input.

XYCHR = lines which must be drawn to produce alphanumeric characters - integer - input.

3.4.68.4 Method

If $\emptyset PT \neq 0$, all other arguments are ignored and LINE is called. Otherwise, the characters are drawn. The width and height of each character position are assumed to be integer multiples of 8 and 16, respectively. The size of the drawn character will be this integer multiple of 6. The remaining space in each character position is used as the horizontal and vertical spacing. No character will be drawn outside the region specified in REG.

3.4.68.5 Design Requirements

Subroutine used: LINE.

UTILITY SUBROUTINE DESCRIPTIONS

3.4.69 FNDPLT (Determine the Internal Plotter and Model Indices).

3.4.69.1 Entry Point: FNDPLT.

3.4.69.2 Purpose

Given the external name and model of a plotter, to determine the corresponding internal plotter and model numbers used by the NASTRAN plotting software package.

3.4.69.3 Calling Sequence

```
CALL FNDPLT (PLØTER,MØDEL,PLTNAM,PMØDEL)
```

where:

PLØTER = internal plotter number - integer - output.

MØDEL = internal model number - integer - output.

PLTNAM(2) = external plotter name - BCD - input.

PMØDEL(2) = external model name - integer or BCD - input and output.

3.4.69.4 Method

PLTNAM and PMØDEL are compared with an internal table of plotter names and models. When a match is found, PLØTER and MØDEL are set to the corresponding internal plotter and model numbers. If a match is found only for the plotter name (PLTNAM), the model name for the first model appropriate to the matched model name will be used to determine PLØTER and MØDEL, and the model name used will be stored in PMØDEL. If no match is found, PLØTER and MØDEL will be set to zero. See section 3.1 for further details.

SUBROUTINE DESCRIPTIONS

3.4.70 PHDMIA (DMI punch routine)

3.4.70.1 Entry Points: PHDMIA, PHDMIB, PHDMIC, PHDMID

3.4.70.2 Purpose

Writes DMI bulk data card images on a FØRTRAN unit for small real matrices.

3.4.70.3 Calling Sequence

CALL PHDMIA - Initializes matrix

CALL PHDMIB - Initializes each non-null column

CALL PHDMIC - Collect each non-zero term of column

CALL PHDMID - Wrap up column

COMMON /PHDMIX / N(2),C,IFØ,TIN,TØUT,IR,IC,NØ,KPP,NLP,ERNØ,ICØL,IRØ,XX

Communication area as follows:

- N - Alphanumeric name of matrix, 2A4.
- C - Alphanumeric string for continuation chaining, A3.
- IFØ - 1 for a square, non-symmetric matrix;
2 for a rectangular matrix;
6 for a symmetric matrix.
- TIN - 1 (input to IFP will be single precision).
- TØUT - 1 if IFP is to generate single-precision terms.
2 if IFP is to generate double-precision terms.
- IR - Number of rows in matrix, Integer > 0.
- IC - Number of columns in matrix, Integer > 0.
- NØ - FØRTRAN printer output unit number (if $NØ \leq 0$, no printing will be done;
if $NØ > 0$, the card images will be listed on FØRTRAN unit NØ as well as
"punched").
- KPP - 1, single-field DMI card images will be generated;
2, double-field DMI card images will be generated.
- NLP - Number of data lines per page.
- ERNØ - 0, no errors were detected;
(output) 1, more than 9999 card images for a single matrix were requested.
- ICØL - Current column number.
- IRØ - Current row number.
- XX - Current value of matrix term as a single-precision floating point number.

UTILITY SUBROUTINE DESCRIPTIONS

3.4.70.4 Method

To use this routine, carry out the steps below after loading the common block.

1. For each matrix, CALL PHDMIA. All data items in /PHDMIX/ must be loaded except ERNØ (output), ICØL, IRØ and XX.
2. For each non-null column, do the following:
 - a. Load ICØL, IRØ and XX with data corresponding to the first non-zero term in the column.
 - b. CALL PHDMIB
 - c. For any other non-zero terms in the column, load IRØ and XX and
 - d. CALL PHDMIC
 - e. After all non-zero terms have been processed, CALL PHDMID to wrap up the column.

Matrices will be punched on single-field DMI cards using a F8.1 format for the element values if KPP = 1; double-field cards will be punched using a 1PE16.8 field if KPP = 2.

3.4.70.5 Design Requirements

A PUNCH file is assumed to exist on FØRTRAN unit 7 (for the CDC and IBM) and unit 1 (for the UNIVAC).

3.4.70.6 Diagnostic Messages

None.

SUBROUTINE DESCRIPTIONS

3.4.71 HEAD (Plot Heading)

3.4.71.1 Entry Point: HEAD

3.4.71.2 Purpose

Create four line heading blocks for structure plot frames.

3.4.71.3 Calling Sequence

CALL HEAD(DTYP,PLTP,MTYP,IDAT)

DTYP - Analysis 0 = 1 Undeformed shape
 1 = STATIC
 2 = FREQ.
 3 = TRANS.
 4 = MØDAL
 5 = CMØDAL

PLTP - Response 1 = DEFØR.
 2 = VELOCITY
 3 = ACCEL.

MTYPE - Variable 1 = FREQ.
 2 = EIGENV.
 3 = TIME

IDAT - Value of phase lag (Real); array of 17 words

Word	Description
1	Number of words that follow (Integer)
3,4	Analysis type heading (4-character BCD)
7	Subcase number (Integer)
8	Load or Mode number (Integer)
9	BCD of LØAD or MØDE
10	Value of frequency, eigenvalue or time (Real)
14 to 16	BCD of PHASE LAG or MAGNITUDE
17	Value of phase log (Real)

3.4.71.4 Method

The TITLE, SUBTITLE and LABEL from /ØUTPUT/ are typed in the lower left hand corner of the plot frame, followed by the plot identification line.

The plot identification line may have the following forms,

UNDEFORMED SHAPE

$$\left\{ \begin{array}{l} \text{STATIC} \\ \text{FREQ.} \\ \text{TRANS.} \\ \text{MØDAL} \\ \text{CMØDAL} \end{array} \right\} \left\{ \begin{array}{l} \text{DEFØR.} \\ \text{VELOCITY} \\ \text{ACCEL.} \end{array} \right\} \text{SUBCASE } n \left[\left\{ \begin{array}{l} \text{LØAD} \\ \text{MØDE} \end{array} \right\} i \right] \left[\left\{ \begin{array}{l} \text{FREQ.} \\ \text{EIGENV.} \\ \text{TIME} \end{array} \right\} v \right] \left\{ \begin{array}{l} \text{PHASE LAG } w \\ \text{MAGNITUDE} \end{array} \right\}$$

UTILITY SUBROUTINE DESCRIPTIONS

The format is controlled by the calling sequence and IDAT(1). The following combinations and parameters for deformed plots are legal:

	STATIC	FREQ.	TRANS.	MØDAL	CMØDAL
IDAT(1)	8	15,16	12	12	15,16
DEFØR.	X	X	X	X	X
VELØCITY		X	X		
ACCEL		X	X		
LØAD	X	X	X		
MØDE				X	X
FREQ.		X		X	X
EIGENV				X	
TIME			X		
PHASE LAG w		X			X
MAGNITUDE		X			X

3.4.71.5 Design Requirements

The plotter software package must be available to this routine.

3.4.71.6 Diagnostic Messages

None.

7-1-73

UTILITY SUBROUTINE DESCRIPTIONS

3.4.72 DELSET (Dummy Element Setup)

3.4.72.1 Entry Point: DELSET

3.4.72.2 Purpose

Modifies /GPTA1/ to accommodate any dummy elements present.

3.4.72.3 Calling Sequence

CALL DELSET

3.4.72.4 Method

The ADUMi bulk data card information is picked up from the 46th thru 54th words of /SYSTEM/ where it was placed by IFS5P. The desired entries in /GPTA1/ are generated and inserted into the 53rd thru 61st positions as required.

3.4.72.5 Design Requirements

All modules using /GPTA1/ should call this routine to insure that the dummy elements are properly recognized.

3.4.72.6 Diagnostic Messages

None.

UTILITY SUBROUTINE DESCRIPTIONS

3.4.73 HMAT (Set Up and Compute Heat Transfer Material Coefficients)

3.4.73.1 Entry Point: HMAT

3.4.73.2 Purpose

Used in place of subroutine MAT and PREMAT for heat transfer analysis. Sets up material property data and calculates properties.

3.4.73.3 Calling Sequence

CALL HMAT (ID,Z)

ID - Element identification number or 0.

Z - Array of open core.

/MATIN/MATID,INFLAG,ELTEMP,DUM,SIN,CØS - Input parameters.

/MATØUT/ØUTPUT(7) - Depends on INFLAG as follows:

<u>INFLAG</u>	<u>ØUTPUT</u>
1	K_1
2	$K_{11} \ K_{12} \ K_{22}$
3	$K_{11} \ K_{12} \ K_{13} \ K_{22} \ K_{23} \ K_{33}$
4	C_p

3.4.73.4 Method

HMAT has two modes. If the value of ID is zero, the MPT data block is read and the MAT4, MAT5, MATT4, and MATT5 data cards are stored in core. A list of material tables is built and subroutine PRETAB is called to read the DIT table data and store in core.

If the value of ID is greater than zero, the data of /MATIN/ is examined. The material ID (MATID) is found in the MAT4 or MAT5 data in core. If the ID is found in the corresponding MATT4 or MATT5 data, the material is assumed to be temperature dependent, and subroutine TAB is called using the value ELTEMP for each table referenced on the MATT4 or MATT5 card. The operations involved for each combination of data is given in the table below.

SUBROUTINE DESCRIPTIONS

TYPE OF CARD	INFLAG = 1	INFLAG = 2	INFLAG = 3	INFLAG = 4
MAT4	$K = C_1$	$K_{11} = C_1$ $K_{12} = 0$ $K_{22} = C_1$	$K_{11} = C_1$ $K_{12} = 0$ $K_{13} = 0$ $K_{22} = C_1$ $K_{23} = 0$ $K_{33} = C_1$	$C_p = C_2$
MAT5	$K = C_1$	$K_{11} = C_1 C^2$ $+ C_4 S^2 - 2C_2 SC$ $K_{12} = (C_1 - C_4)$ $SC + (C^2 - S^2)C_2$ $K_{22} = C_1 S^2 +$ $C_2 C^2 + 2C_2 SC$	$K_{11} = C_1$ $K_{12} = C_2$ $K_{13} = C_3$ $K_{22} = C_4$ $K_{23} = C_5$ $K_{33} = C_6$	$C_p = C_7$

Note: $C = \cos$, $S = \sin$

UTILITY SUBROUTINE DESCRIPTIONS

3.4.74 BISRCH (Binary Search)

3.4.74.1 Entry Point: BISRCH

3.4.74.2 Purpose

To perform a binary search on a list of sorted data.

3.4.74.3 Calling Sequence

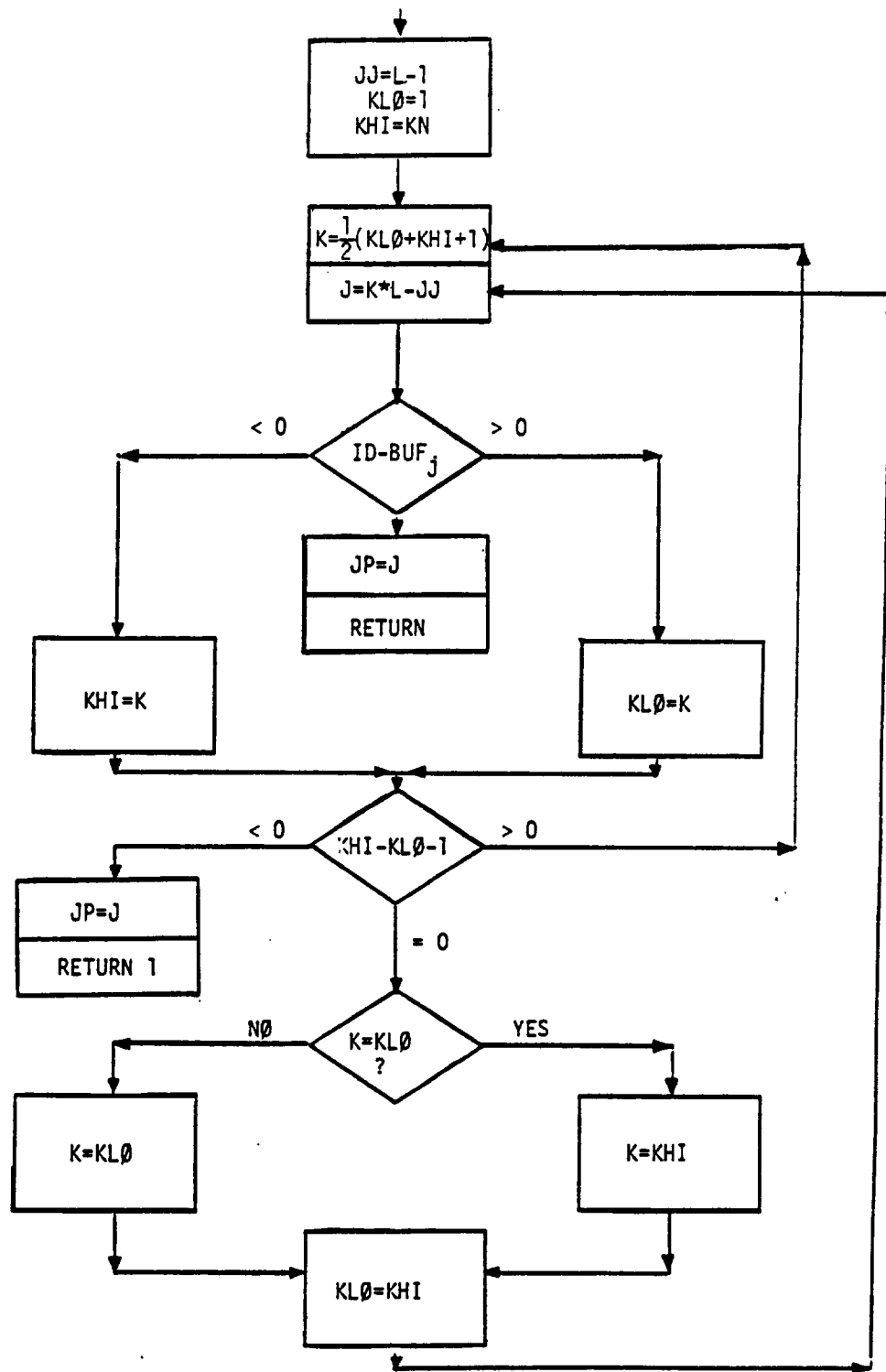
CALL BISRCH (\$n ID,BUF,L,KN,JP)

- n - FORTRAN statement number defining the return to be taken in the event a match is not found.
- ID - Word to match with first word of entry.
- BUF - Array to be searched.
- L - Length of each entry of array.
- KN - Number of entries in BUF.
- JP - Pointer returned to calling program. This pointer gives the first word of the matching entry in the array.

SUBROUTINE DESCRIPTIONS

3.4.74.4 Method

A standard binary search algorithm is used as shown below:



UTILITY SUBROUTINE DESCRIPTIONS

3.4.74.5 Design Requirements

None.

3.4.74.6 Diagnostic Messages

None.

UTILITY SUBROUTINE DESCRIPTIONS

3.4.75 FØRFIL (File Unit)

3.4.75.1 Entry Point: FØRFIL

3.4.75.2 Purpose

To extract the logical unit to which a given GINØ file name belongs.

3.4.75.3 Calling Sequence

INTEGER FØRFIL

NUNIT = FØRFIL(NAME)

NAME - GINØ file name

3.4.75.4 Method

FØRFIL searches the FIST for the GINØ file name. When a match is found, the internal unit number is either /XXFIAT/ or /XFIAT/ and is extracted and returned through the function name as in integer.

3.4.75.5 Design Requirements

None.

3.4.75.6 Diagnostic Messages

Message 2179 may be issued in the event that the requested GINØ file name cannot be found.

SUBROUTINE DESCRIPTIONS

3.4.76 DADØTB (Double Precision Vector Dot Product)

3.4.76.1 Entry Point: DADØTB

3.4.76.2 Purpose

To compute the scalar inner product of two vectors in double precision.

3.4.76.3 Calling Sequence

DØUBLE PRECISION DADØTB, A(3),B(3),C

C = DADØTB(A,B)

3.4.76.4 Method

$$C = \sum_{i=1}^3 A_i B_i$$

3.4.76.5 Design Requirements

None.

3.4.76.6 Diagnostic Messages

None.

UTILITY SUBROUTINE DESCRIPTIONS

3.4.77 DAXB (Double Precision Vector Cross Product)

3.4.77.1 Entry Point: DAXB

3.4.77.2 Purpose

To compute the vector product of two vectors in double precision.

3.4.77.3 Calling Sequence

DOUBLE PRECISION A(3),B(3),C(3)

CALL DAXB (A,B,C)

3.4.77.4 Method

$$\vec{D} = \vec{A} \times \vec{B}$$

$$\vec{C} \Leftarrow \vec{D}$$

3.4.77.5 Design Requirements

\vec{C} may overlay \vec{A} or \vec{B} in core.

3.4.77.6 Diagnostic Messages

None.

SUBROUTINE DESCRIPTIONS

3.4.78 SADØTB (Single Precision Vector Dot Product)

3.4.78.1 Entry Point: SADØTB

3.4.78.2 Purpose

To compute the scalar inner product of two vectors in single precision.

3.4.78.3 Calling Sequence

DIMENSION A(3),B(3)

C = SADØTB (A,B)

3.4.78.4 Method

$$C = \sum_{i=1}^3 A_i B_i$$

3.4.78.5 Design Requirements

None.

3.4.78.6 Diagnostic Messages

None.

UTILITY SUBROUTINE DESCRIPTIONS

3.4.79 SAXB (Single Precision Vector Cross Product)

3.4.79.1 Entry Point: SAXB

3.4.79.2 Purpose

To compute the vector product of two vectors in single precision.

3.4.79.3 Calling Sequence

DIMENSION A(3),B(3),C(3)

CALL SAXB (A,B,C)

3.4.79.4 Method

$$\vec{D} = \vec{A} \times \vec{B}$$

$$\vec{C} \Leftarrow \vec{D}$$

3.4.79.5 Design Requirements

\vec{C} may overlap \vec{A} or \vec{B} in core.

3.4.79.6 Diagnostic Messages

None.

SUBROUTINE DESCRIPTIONS

3.4.80 HBDY (Compute HBDY Element Coefficients)

3.4.80.1 Entry Point: HBDY

3.4.80.2 Purpose

To compute the variables A_1 , $\{A_i\}$, $\{V_1\}$, $\{V_2\}$, and/or $\{G_i\}$ from the HBDY element data.

3.4.80.3 Calling Sequence

CALL HBDY (ECPT,NECPT,IØPT,RDATA,IDATA)

ECPT - ECPT/EST data for one element, real array, input.

NECPT - ECPT/EST data for one element integer array, input.

IØPT - Option flag for output data.

RDATA - Real array of output data.

IDATA - Integer array of output data.

3.4.80.4 Method and Design Requirements

See Section 4.87.17 for details of the algorithm. The input arrays ECPT and NECPT may be the same array. The output arrays RDATA and IDATA are assumed to be the same array.

UTILITY SUBROUTINE DESCRIPTIONS

3.4.81 DECØDE (Decodes "1" bits in a 32-bit computer word)

3.4.81.1 Entry Point: DECØDE

3.4.81.2 Purpose

To decode a 32-bit computer word and return a list of integers corresponding to each bit position which is "on" (i.e. = 1).

3.4.81.3 Calling Sequence

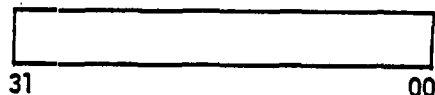
CALL DECØDE (CØDE, LIST, N)

CØDE - the word to be decoded

LIST - address of an array of dimension ≥ 32 where the integers corresponding to bit positions which are "on" are stored.

N - the number of integers stored at LIST.

Note: the numbering convention for the bit positions is right (low order) to left (high order) 00 through 31.



3.4.81.4 Method

DECØDE uses the NASTRAN ANDF and common block /TWØ/.

SUBROUTINE DESCRIPTIONS

3.4.82 ECTLØC (Special Version of Entry Point LØCATE)

3.4.82.1 Entry Point: ECTLØC

3.4.82.2 Purpose: To read either the GEØM2 or ECT data block sequentially and position the data block to read element connection data from a logical record which has a corresponding definition in the element data table in common block /GPTA1/.

3.4.82.3 Calling Sequence

CALL ECTLØG (\$n, ECT, BUF, IELEM)

- n - FØRTRAN Statement number definition return when end-of-file on the data block is encountered. In this case, ECTLØC closes the data block prior to return.
- ECT - GINØ reference name for either the GEØM2 or ECT data block.
- BUF - Address of a 3-word array into which the first 3 words of each logical record are read.
- IELEM - Pointer to the first word of the element entry in /GPTA1/ corresponding to the logical record at which the data block is now positioned.

Note: The file ECT must be open on each entry to ECTLØC.

3.4.82.4 Method

ECTLØC differs from LØCATE in that ECTLØC is designed to read the data block once sequentially returning pointers to the elements in the data block. LØCATE is, of course, more general. ECTLØC will operate successfully only on the GEØM2 or ECT data blocks.

UTILITY SUBROUTINE DESCRIPTIONS

3.4.83 MRGE (Merges two sorted lists to form a new list)

3.4.83.1 Entry Point: MRGE

3.4.83.2 Purpose

To form a new unique list as a result of merging two sorted lists.

3.4.83.3 Calling Sequence

CALL MRGE (LIST, N, STRING, M)

LIST - An array containing a sorted list which is to be merged and potentially extended.

N - Input: the number of entries in LIST;
Output: the number of entries in the extended LIST.

STRING - An array containing a sorted list to be merged with LIST.

M - The number of entries in STRING.

Note: LIST is both an input and output array. Its dimension must be $\geq N+M$.

3.4.83.4 Method

BISRCH is called to locate the position of the first entry in STRING in LIST. A "hole" in LIST is created by moving entries from the point determined to the end of the LIST array (N+M). At this point the two strings (STRING and LIST) are merged to form a new list. A new value of N is returned.

Example:

prior to call

LIST = 5, 8, 9, 13, 17, 20

N = 6

STRING = 10, 12, 13, 15, 17

M = 5

CALL MRGE (LIST, N, STRING, M)

after call

LIST = 5, 8, 9, 10, 12, 13, 15, 17, 20

N = 9

STRING = 10, 12, 13, 15, 17 } unchanged

M = 5

UTILITY SUBROUTINE MODULES

3.4.84 DMPFIL (Print a NASTRAN Scratch Data Block)

3.4.84.1 Entry Point: DMPFIL

3.4.84.2 Purpose: To provide a debugging printout of a scratch data block

3.4.84.3 Calling Sequence: CALL DMPFIL (IFILE,IZ,LZ)

IFILE is the GINØ file name of the data block. IFILE > 0 implies a table,

IFILE < 0 implies a matrix.

IZ is an available array of storage

LZ is the length of IZ

3.4.84.4 Method: If DIAG 30 is not set DMPFIL will return. (See BUG for method of setting DIAG 30.) LZ must be at least SYSBUF + 5 or DMPFIL will terminate. DMPFIL will dump 5 words per line (I10 and E11.3 for each word) if it is a table. The file must be closed upon entry and will be closed upon return. At the conclusion of the dump, the trailer will be displayed. A matrix will be dumped based on its matrix control block (up to 300 columns and 300 rows) in single precision, 10 values per line.

3.4.84.5 Design Requirements: DMPFIL calls BUG.

3.4.84.6 Diagnostic Messages: None.

SUBROUTINE DESCRIPTIONS

3.4.85 SSPLIN (Produce an Interpolation Matrix)

3.4.85.1 Entry Point: SSPLIN.

3.4.85.2 Purpose

The purpose of this subroutine is to produce an interpolation matrix for functions of two variables (x, y). The options include:

1. Nonsymmetric, symmetric, or axisymmetric in either x or y.
2. The output matrix, G, may be for interpolation only, or interpolation plus slope.
3. The matrix G or its transpose may be computed.
4. "Attachment flexibility" is included for smoothing the interpolation.

3.4.85.3 Calling Sequence

CALL SSPLIN(NI,XYI,ND,XYD,KX,KY,KD,KT,DZ,G,NCØRE,ISNG)

where:

- NI - Number of independent points - integer - input
- XYI - Array of independent point locations in pairs (X(I),Y(I), I = 1,NI) - real - input
- ND - Number of dependent points - integer - input
- XYD - Array of dependent points in pairs (X(I),Y(I) I = 1,ND) - real - input
- KX - Symmetry flag for X direction; +1 = Symmetric, -1 = antisymmetric, 0 = general - integer - input
- KY - Symmetry flag for Y direction; same convention as KX.
- KD - Flag for slopes; 0 = interpolation only, 1 = interpolation plus slope - integer - input
- KT - Flag for transposition; 0 = G output, 1 = G^T output - integer - input
- DZ - Attachment flexibility - real ≥ 0.0 - input
- G - Array location for output G matrix - real - output
- NCØRE - Amount of core available for subroutine starting at G(1) - integer - input
- ISNG - Flag for singular matrix; +2 = singular matrix for spline equations - integer - output

UTILITY SUBROUTINE MODULES

3.4.85.4 Method

A check is made to see if enough core is available. If there is sufficient core, the matrices A, B, and C are built in core (see the Theoretical Manual, Section 17.3 for a description of these matrices). Then the output matrix G is returned in core after the following computations using utility subroutines INVERS and GMMATS:

$$G = \begin{cases} B^T (A^{-1}C) & \text{if } KT = 0 \\ C^T (A^{-1}B) & \text{if } KT = 1 \end{cases}$$

If INVERS gets a singular matrix, no G output is produced and ISNG = 2.

3.4.85.5 Design Requirements

Enough core is needed to satisfy the following equation:

$$NCORE > (NI+3)^2 + 3*(NI+3) + NI*ND*(1+KD) + \begin{cases} NI*(NI+3) & \text{if } KT = 0 \\ (NI+3)*ND*(1+KD) & \text{if } KT = 1 \end{cases}$$

3.4.85.6 Diagnostic Messages

If not enough core is available, message 3008 occurs.

SUBROUTINE DESCRIPTIONS

3.4.86 LSPLIN (Produce an Interpolation Matrix)

3.4.86.1 Entry Point: LSPLIN

3.4.86.2 Purpose

The purpose of this subroutine is to produce an interpolation matrix for functions of one variable (y). The options include:

1. Nonsymmetry, symmetry, or antisymmetry about y.
2. The output G matrix may be for interpolation only, or for interpolation plus slopes.
3. The matrix G or its transpose may be computed.
4. "Attachment flexibility" is included for smoothing the interpolation.
5. The interpolation may be based upon slopes at the independent points, or just independent deflections.

3.4.86.3 Calling Sequence

CALL LSPLIN (NI,XYI,ND,XYD,KY,KD,KT,DZ,DTX,DTY,DTØR,G,NCØRE,ISNG)

where:

- NI - Number of independent points - integer - input
- XYI - Array of independent points in pairs (X(I), Y(I) I = 1,NI) - real - input
- ND - Number of dependent points - integer - input
- XYD - Array of dependent points in pairs (X(I), Y(I) I = 1,ND) - real - input
- KY - Symmetry flag for Y direction; 0 = general, 1 = symmetric, -1 = antisymmetric - integer - input
- KD - Flag for slopes; 0 = interpolation, 1 = interpolation plus slopes about X, 2 = interpolation plus slopes about X and Y - integer - input
- KT - Flag for transposition; 0 = G output, 1 = G^T output - integer - input
- DZ - Attachment flexibility - real ≥ 0.0 - input
- DTX - Slope attachment flexibility (X); 0.0 or greater = use in computations, less than 0.0 = omit these slopes - real - input

UTILITY SUBROUTINE MODULES

- DTY - Slope attachment flexibility (Y); 0.0 or greater = use in computations, less than 0.0
= omit these slopes - real - input
- DTØR - Torsion flexibility - real \geq 0.0 - input
- G - Array location for output matrix - real - output
- NCØRE - Amount of core available starting at G(1) - integer - input
- ISNG - Flag for singular matrix; 2 = singular - integer - output

3.4.86.4 Method

A check is made to see if enough core is available. If there is sufficient core, the matrices A, B, and C are built (see Theoretical Manual Section 17.3 for a description of these matrices). Then the output matrix G is produced using the utility subroutines INVERS and GMMATS.

$$G = \begin{cases} B^T (A^{-1}C) & \text{if } KT = 0 \\ C^T (A^{-1}B) & \text{if } KT = 1 \end{cases}$$

If INVERS produces a singular matrix, no G matrix is produced and ISNG = 2.

3.4.86.5 Design Requirements

Enough core is needed to satisfy the following equation:

$$NCØRE > N*N + 3*N + S*NI*ND*(1+KD) + \begin{cases} S*NI*N & \text{if } KT = 0 \\ N*ND*(1+KD) & \text{if } KT = 1 \end{cases}$$

where:

$$S = 3 \text{ if } DTX \text{ and } DTY \geq 0.0$$

$$2 \text{ if } DTX \text{ or } DTY < 0.0$$

$$1 \text{ if } DTX \text{ and } DTY < 0.0$$

$$N = S*NI + (3 \text{ if } KY = 0, 2 \text{ if } KY = 1, 1 \text{ if } KY = -1)$$

3.4.86.6 Diagnostic Messages

If not enough core is available, message 3008 occurs.

SUBROUTINE DESCRIPTIONS

3.4.87 BISLØC (Binary Search Routine)

3.4.87.1 Entry Point: BISLØC

3.4.87.2 Purpose

The purpose of this subroutine is to use a binary search to position a word in a table using the first entry.

3.4.87.3 Calling Sequence

CALL BISLØC (\$n₁, ID, ARR, LEN, KN, JLØC)

where:

n_1 - Nonstandard return if ID is not one of the first entry words in ARR

ID - Integer to locate as first word of entry - input

ARR - Table to search - input

LEN - Number of words in each entry of ARR - integer - input

KN - Number of entries in ARR - integer - input

JLØC - Integer pointer to location of first word in the entry. If the nonstandard return is taken, JLØC is where the entry should be positioned - output

3.4.87.4 Method/Example

A table with two words per entry exists as follows:

<u>Location</u>	<u>Contents</u>	where location is word position in ARR
1	3	
2	19.0	LEN = 2

3	5	KN = 3
4	6.0	

5	6	
6	20.0	

UTILITY SUBROUTINE MODULES

If ID = 3, the standard return with JLØC = 1 occurs. If ID = 4, the nonstandard return with JLØC = 3 occurs. This is the position ID should have been. If ID = 100, the nonstandard return with JLØC = 7 occurs.

SUBROUTINE DESCRIPTIONS

3.4.88 BISHEL (Merge Unique Entries into an Array)

3.4.88.1 Entry Point: BISHEL

3.4.88.2 Purpose

The purpose of this subroutine is to merge an entry based on the first word into a list sorted on the first word. If the entry is a duplicate, only a nonstandard return results.

3.4.88.3 Calling Sequence

CALL BISHEL (n_1 , LIST, NENT, NTERM, ARRAY)

where:

- n_1 - Nonstandard return if the first word of the entry is not unique
- LIST - NTERM words to merge into ARRAY
- NENT - Number of words in ARRAY, integer
- NTERM - Number of words per entry in ARRAY, integer
- ARRAY - Array sorted on first word of each entry

3.4.88.4 Method

The array will be sorted on the first word of each entry, an integer.

For the first call (NENT = 0), the LIST is added to ARRAY and a return is made.

For an existing array (NENT \geq NTERM) the LIST is checked if it is beyond the end of the table. If it is, the LIST is appended and a return is made. If it is not, subroutine BISLØC is called. A nonstandard return is made if the entry is not unique, otherwise the high end of the array is pushed down and the LIST items inserted.

In all cases, NENT is properly set. (NENT = $i \cdot$ NTERM, $i=1,2,\dots$)

3.4.88.5 Subroutines Called

BISLØC (see Section 3.4.87)

UTILITY SUBROUTINE MODULES

3.4.88.6 Example

Let ARRAY be desired to be built as

<u>Location</u>	<u>Contents of X</u>
1	5
2	ABC
3	DEF
- - -	- - -
4	10
5	XYZ
6	WVU
- - -	- - -
7	15
8	PQR
9	STU
- - -	- - -

To get this, the following sequence is one way this could have taken place:

<u>LIST</u>		
1	10	NW=0
2	XYZ	CALL BISHEL (\$,LIST,NW,3,X)
3	WVU	NW will be 3 on return
1	15	
2	PQR	CALL BISHEL (\$,LIST,NW,3,X)
3	STU	NW will be 6 on return
1	5	
2	ABC	CALL BISHEL (\$,LIST,NW,3,X)
3	DEF	NW will be 9 on return

SUBROUTINE DESCRIPTIONS

3.4.89 BUG (Produce debugging printout)

3.4.89.1 Entry Point: BUG

3.4.89.2 Purpose

The purpose of this subroutine is to provide debugging printout on certain runs.

3.4.89.3 Calling Sequence

CALL BUG(LABEL,ISTNØ,BUF,LBUF)

where:

LABEL = a one-word BCD table for the printed data - BCD - input

ISTNØ = an integer label (such as FØRTRAN statement number) - BCD - input

BUF = the array to be printed - input

LBUF = the number of words to be printed - integer - input

3.4.89.4 Example

CALL BUG (4HTEST,500,10,1) will print:

TEST IN AREA OF LABEL 500

10 0.0E0

3.4.89.5 Restrictions

- (A) BUG uses the second word of /SYSTEM/ as its output unit.
- (B) BUG will only print up to 5000 lines (until overlayed).
- (C) BUG will not print a negative number of words or more than 1000 words in one call.
- (D) BUG prints three words per line, printing each word as I10 and 1PE21.3.
- (E) DIAG 30 must be set for BUG to print.

Note that the user cannot set DIAG 30 and get thru the PREFACE. DIAG 30 must be set prior to entry of the "bugged" module via the PARAM DMAP statement, i.e.,

PARAM //C,N,SSST/C,N,30 \$

UTILITY SUBROUTINE DESCRIPTIONS

3.4.90 SKPREC (Skip Records on a File)

3.4.90.1 Entry Point: SKPREC

3.4.90.2 Purpose

The purpose of this subroutine is to skip forward or backward on a given file.

3.4.90.3 Calling Sequence

CALL SKPREC (NAME,NREC)

where:

NAME = GINØ file name of the data block to be positioned - integer - input

NREC = number of records to skip - integer - input; NREC = 0 implies RETURN; NREC > 0, NREC calls to FWDREC will occur; NREC < 0, |NREC| calls to BCKREC will occur.

3.4.90.4 Design Requirements

If SKPREC hits an EOF while going forward, a fatal message (3002) will result.

SUBROUTINE DESCRIPTIONS

3.4.91 RE2AL (Real number to alphanumeric)

3.4.91.1 Entry Point: RE2AL

3.4.91.2 Purpose

To convert a single precision to alphanumeric that may be printed with a 2A4 format.

3.4.91.3 Calling Sequence

CALL RE2AL(RE,ALPH)

where:

RE - single precision real number, input

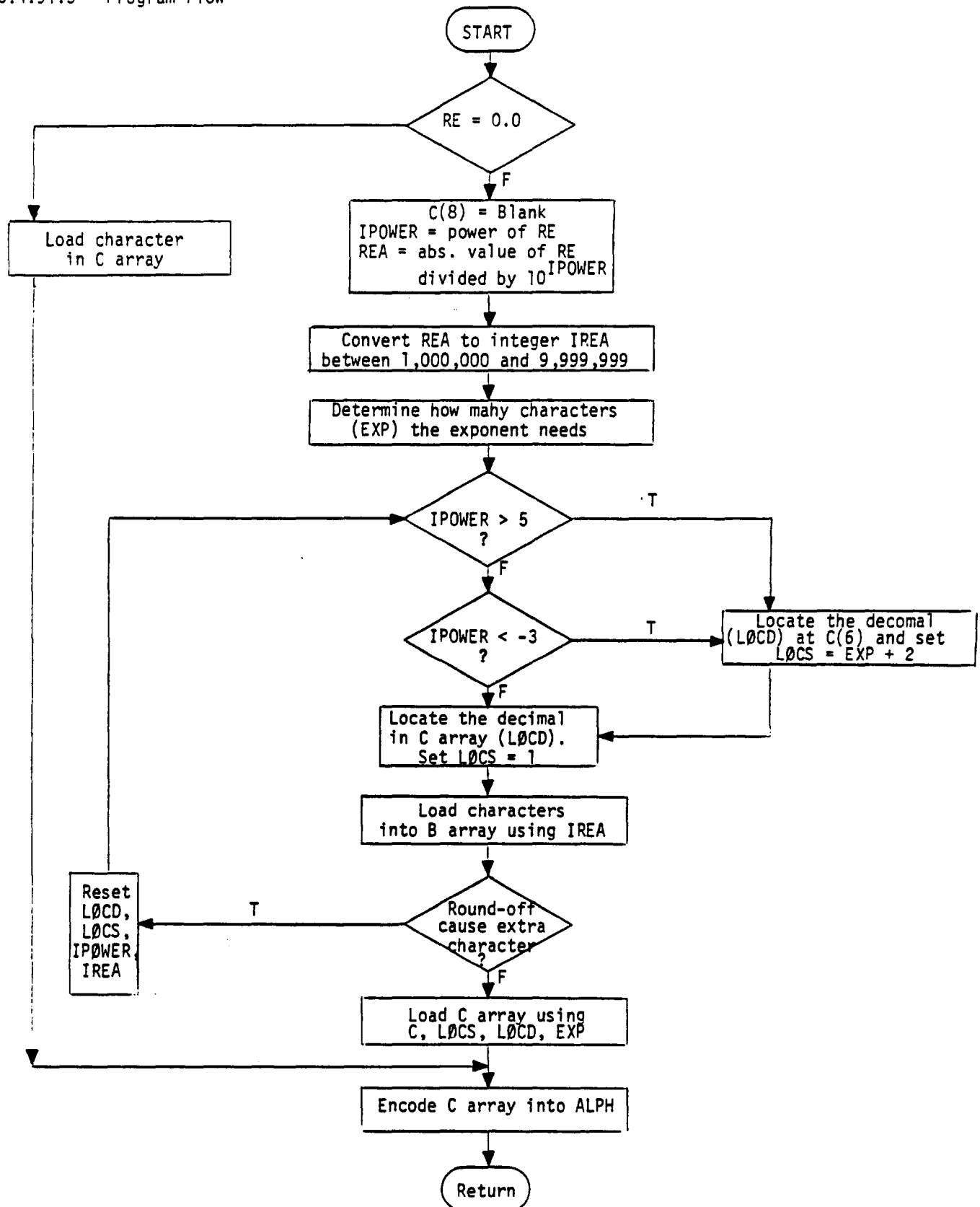
ALPH - BCD output (2 words)

3.4.91.4 Method

The output value is in the floating or exponential form depending on which gives greater significant digits. Round off is based on the number of significant digits being output.

UTILITY SUBROUTINE DESCRIPTIONS

3.4.91.5 Program Flow



SUBROUTINE DESCRIPTIONS

3.4.92 CPYSTR (Copies a string formatted record)

3.4.92.1 Entry Point: CPYSTR

3.4.92.2 Purpose: To copy a logical record written in string format from one data block to another data block.

3.4.92.3 Calling Sequence:

CALL CPYSTR (INBLK, ØUTBLK, FLAG, CØL)

INBLK - 15-word string communication block for input file.

ØUTBLK - 15-word string communication block for output file.

FLAG - $\begin{cases} 0: & 1^{st} \text{ call to GETSTR has not been made} \\ \neq 0: & 1^{st} \text{ call to GETSTR has been made} \end{cases}$

CØL $\begin{cases} 0: & \text{column number is in INBLK(12)} \\ \neq 0: & \text{column number is CØL} \end{cases}$

3.4.92.4 Method

CPYSTR uses GETSTR and PUTSTR and moves the strings directly from one buffer to another. If possible, consecutive strings are concatenated.

UTILITY SUBROUTINE DESCRIPTIONS

3.4.93 CPYFIL (CoPY FIle)

3.4.93.1 Entry Point: CPYFIL

3.4.93.2 Purpose: To copy a logical file from one data block to another data block.

3.4.93.3 Calling Sequence:

CALL CPYFIL (INFILE,ØUFILE,AREA,N,CØUNT)

INFILE = GINØ file name of file from which GINØ file is copied.

ØUFILE = GINØ file name of file to which GINØ file is copied.

AREA = Work area of Length N

N = Length of AREA

CØUNT = Number of words copied

3.4.93.4 Method

CPYFIL calls RECTYP to determine the type of the next record on the input data block. For a normal NASTRAN logical record, READ and WRITE are used to copy the record. For a string formatted logical record, CPYSTR is used to copy the record. The process is continued until an end-of-file is encountered.

SUBROUTINE DESCRIPTIONS

3.4.94 SETFND (ID lookup utility)

3.4.94.1 Entry Point: SETFND

3.4.94.2 Purpose

To find an ID in a sorted set list which may contain the NASTRAN thru notation. An example of such a sorted set as might be found in the CASECC data block is:

4, 5, 10, 15, - 19, 31, 55

The above list includes the following ID's:

4, 5, 10, 15, 16, 17, 18, 19, 31, 55

3.4.94.3 Calling Sequence

CALL SETFND(\$n₁,SET,LSET,ID,NEXT)

n₁ - FORTRAN statement label defining return to be taken in the event a match on 'ID' is not in the list 'SET'

SET - The sorted array list of the type shown in the above example. This list is in sort with respect to absolute values only. Negative ID's are undefined with respect to this routine.

LSET - Length in computer words of the list 'SET'

ID - Positive identification to be searched for in the list 'SET'

NEXT - If calling this routine continuously with ID's themselves which are in sort lowest to highest, NEXT should be a local variable (not used for anything else) set equal to 1 on the first call and not modified on subsequent calls. If ID's being searched for are in nonsorted order, NEXT should be a local variable set equal to 1 on each call.

3.4.94.4 Method

SETFND performs a linear search of the list starting at SET(NEXT) until either the list is exhausted or an ID in the list 'SET' is greater than or equal in absolute value to the ID to be found. If the list is exhausted, the nonstandard return is made. If the ID is matched exactly, a normal return is taken. If an ID of 'SET' is greater or equal to the absolute value and

UTILITY SUBROUTINE DESCRIPTIONS

negative in true value, then the normal return is made. In all cases when the search begins, SETFND assumes that the SET(NEXT) in the list 'SET' is less than or equal to the ID in question. Thus, the routine can be called with ID's to match which they themselves are in sort, and which may also contain repeated ID's, without resetting the value of 'NEXT'. SETFND always leaves NEXT equal to the index of the last ID looked at or one greater than the value of LSET.

SUBROUTINE DESCRIPTIONS

3.4.95 SAMB (Vector Subtract)

3.4.95.1 Entry Point: SAMB

3.4.95.2 Purpose

The purpose of SAMB is to subtract two vectors of length 3.

3.4.95.3 Calling Sequence

CALL SAMB(A,B,C)

$$\vec{A} - \vec{B} = \vec{C}$$

Vectors may overlap.

UTILITY SUBROUTINE DESCRIPTIONS

3.4.96 SAPB (Vector Add)

3.4.96.1 Entry Point: SAPB

3.4.96.2 Purpose

The purpose of SAPB is to add two vectors of length 3.

3.4.96.3 Calling Sequence

CALL SAPB(A,B,C)

$A+B = C$

Vectors may overlap.

SUBROUTINE DESCRIPTIONS

3.4.97 GMMATC (General Matrix Multiply and Transpose - Complex Single Precision)

3.4.97.1 Entry Point: GMMATC

3.4.97.2 Purpose

To perform the same operations as GMMATD, but where [A], [B], [C] and [D] are complex single precision matrices.

3.4.97.3 Calling Sequence

CALL GMMATC(A,IRØWA,ICØLA,MTA,B,IRØWB,ICØLB,MTB,C)

This routine is exactly the same as GMMATD except this routine operates on complex single precision matrices. See Section 3.4.32.

MATRIX SUBROUTINE DESCRIPTIONS

3.5 MATRIX SUBROUTINE DESCRIPTIONS

3.5.1 PAKUNPK (PAcK/UNPaK)

3.5.1.1 Entry Points: BLDPK, BLDPKI, ZBLPKI, BLDPKN, INTPK, INTPKI, ZNTPKI, PACK, UNPACK

3.5.1.2 Purpose: To provide a general capability for input, output and interpretation of matrices.

3.5.1.3 Calling Sequences

If several different matrices are to be packed concurrently, the multi-column version of BLDPK is used:

```
CALL BLDPK(TYPIN,TYPØUT,NAME,BLØCK,FLAG)
```

```
CALL BLDPKI(A,I,NAME,BLØCK)
```

```
CALL BLDPKN(NAME,BLØCK,MCB)
```

where:

BLDPK is an initialization call and is made once for each column to be packed.

BLDPKI is the call made to supply a single element of the column to be packed.

BLDPKN is a call to terminate processing of the column.

TYPIN - Arithmetic type of the elements to be packed (1 = real single precision, 2 = real double precision, 3 = complex single precision, 4 = complex double precision) - integer - input.

TYPØUT - Arithmetic type of the elements in the packed column - integer - input.

NAME - GINØ reference name of data block where packed column will be written.

BLØCK - An array of dimension ≥ 20 for use by BLDPK and BLDPKI.

A - An array of dimension 1, 2 or 4 (depending on TYPIN) where the element to be packed is stored - real - input.

FLAG - { 1 to indicate multi-column version with no string trailers.
-1 to indicate multi-column version with string trailers.

I - Row position of element to be packed - integer - input.

MCB - An array of dimension 7 where the trailer information about the matrix is accumulated.

If only one matrix is being packed, the single column version should be used as it is more efficient.

```
COMMON/ZBLPKX/A(4),I
```

```
CALL BLDPK(TYPIN,TYPØUT,NAME,0,0)
```

SUBROUTINE DESCRIPTIONS

CALL ZBLPKI

CALL BLDPKN(NAME,0,MCB)

where:

BLDPK and its arguments are as defined above.

ZBLPKI is the call made to provide an element of the column to be packed. The element (A), and its row position (I), are stored in /ZBLPKX/ by the user prior to each CALL ZBLPKI.

BLDPKN and its parameters are defined as above.

Note:

BLDPKN accumulates the following three words of MCB:

MCB(2) = column number

MCB(6) = number of words in the densest column

MCB(7) = number of non-zero words in the matrix

In the multi-column version, BLOCK must be different for each matrix being packed.

To pack a column of a matrix:

COMMON/PACKX/TYPIN,TYPØUT,I,N,INCR

CALL PACK(A,NAME,MCB)

A - An array where the elements of the column are stored in unpacked form.

NAME - GINØ name of the data block where the packed column will be written.

MCB - An array of dimension = 7 where the matrix trailer information will be accumulated.

TYPIN - Arithmetic type of the elements of the column stored at A (1 = real single precision, 2 = real double precision, 3 = complex single precision, 4 = complex double precision).

TYPØUT - Arithmetic type in which the elements are to be in packed form. Same convention as TYPIN.

I - Row position of the element stored at A(1).

N - Row position of the last element in the column stored at A.

INCR - Spacing of the elements in column stored at A in units of elements, e.g., if real double precision elements are stored consecutively, INCR = 1.

If several different matrices are to be read and interpreted concurrently, the multi-column version of INTPK is used.

CALL INTPK(\$n,NAME,BLOCK,TYPØUT,1)

CALL INTPKI(A,I,NAME,BLOCK,EØL)

MATRIX SUBROUTINE DESCRIPTIONS

where:

INTPK is the initialization call and is made once for each column to be read and interpreted.

INTPKI is the call made to read successive non-zero elements of the column. Each call to INTPKI returns one non-zero element.

n - FORTRAN statement number defining return to be taken in the event the column is null.

NAME - GINØ file name of data block where the matrix is stored.

BLØCK - An array of dimension ≥ 20 for use by INTPK and INTPKI.

TYPØUT - Arithmetic type into which the elements are to be unpacked (+1 = real single precision, +2 = real double precision, +3 = complex single precision, +4 = complex double precision). If TYPØUT < 0, the sign of each non-zero element is to be changed - integer - input.

A - An array of dimension 1, 2 or 4, depending on TYPØUT, where the non-zero element is to be stored - real - output.

I - Row position of the non-zero element - integer - output.

EØL = 1 indicates last non-zero element in the column was read on the current call to INTPKI, =0 otherwise - integer - output.

If only one matrix is to be read and interpreted, the single-column version should be used as it is more efficient.

```
COMMON/ZNTPKX/A(4),I,EØL,EØR
```

```
CALL INTPK($n,NAME,),TYPØUT,0)
```

```
CALL ZNTPKI
```

where:

INTPK and its arguments are defined as above.

ZNTPKI is the call made to read successive non-zero elements of the column. One element (A), its row position (I), end-of-column indicator (EØL), and end-of-record indicator (EØR) are stored in /ZNTPKX/ for each call to ZNTPKI.

EØL is defined as above.

EØR = 1 indicates the end-of-record has been read by ZNTPKI, = 0 otherwise (EØR may be one before EØL is one. EØR is always one when EØL = 1).

To read and unpack a column of a matrix stored in NASTRAN packed format:

```
CALL UNPACK($n,NAME,A)
```

```
COMMON/UNPAKX/TYPØUT,I,N,INCR
```

where:

n - FORTRAN statement number defining return to be taken if the column is null.

NAME - GINØ name of data block containing the column to be unpacked.

A - An array where the unpacked column will be stored.

SUBROUTINE DESCRIPTIONS

TYPØUT - Arithmetic type in which the elements are to be stored at A (1 = real single precision, 2 = real double precision, 3 = complex single precision, 4 = complex double precision). TYPØUT < 0 means that each of the elements will be stored with a change of sign.

I - Row position of the element to be stored at A(1).

N - Row position of the last element to be stored at A.

INCR - Spacing of the elements to be stored at A in units of elements, i.e., if complex single precision elements are to be stored at A(1), A(5), A(7), etc., INCR = 2.

Notes:

1. Zeros are stored for zero elements.
2. If $I < 0$ or $N < 0$, the column is unpacked from the first non-zero element through the last non-zero element and I and N are set to these row positions.
3. If return to statement n is given, zeros are not stored at A.

In addition, Level 15 calling sequences are also supported by PAKUNPK. As a result, the following forms are also acceptable:

```
CALL BLDPK(TYPIN,TYPØUT,NAME,BLØCK,WRITE,1)
```

```
CALL BLDPKI(A,I,NAME,BLØCK,WRITE)
```

```
CALL BLDPKN(NAME,BLØCK,WRITE,MCB)
```

```
CALL INTPK($n,NAME,BLØCK,READ,TYPØUT,1)
```

```
CALL INTPKI(A,I,NAME,BLØCK,READ,EØL)
```

```
CALL PACK(A,NAME,WRITE,MCB)
```

```
CALL UNPACK($n,NAME,A,READ)
```

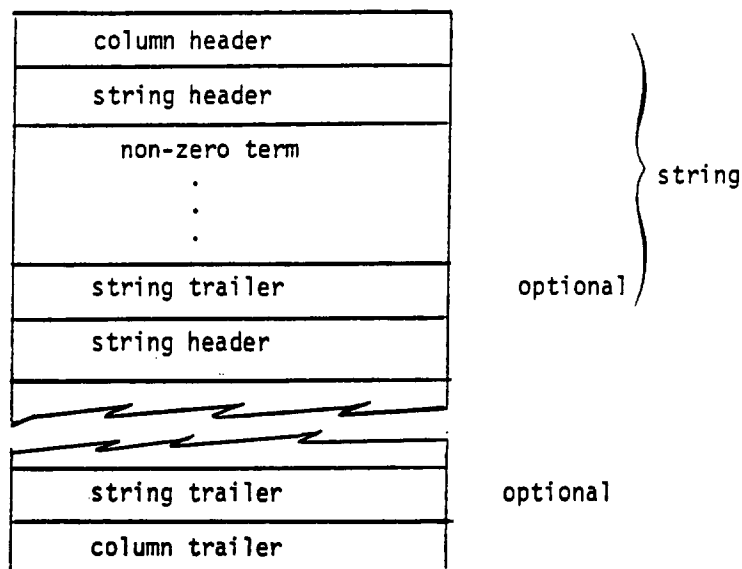
Note: The Level 15 "RDCØR" and "WRTCØR" options are not currently supported.

3.5.1.4 Method

To provide maximum efficiency, PAKUNPK is coded in assembly language on each of the three NASTRAN computers. The logic of each of the versions is very similar. Input and output of the matrix elements is accomplished through the new Level 16 entry points PUTSTR, ENDPUT, GETSTR, ENDGET.

MATRIX SUBROUTINE DESCRIPTIONS

The packed matrix format for Level 16 is as follows:



The format of the column header and trailer and string header and trailer is machine dependent. However, the contents of the control words is machine independent and is as follows:

column header or trailer

type of element in column (1,2,3 or 4)

format of strings (0=no trailers, 1=trailers)

column number

string header or trailer

row position of $\begin{Bmatrix} \text{first} \\ \text{last} \end{Bmatrix}$ element in string

number of terms in string

Use of the string entry points to access terms directly in the GINØ buffer has simplified the logic of PAKUNPK and increased its efficiency. Pointers, addresses and indicators are maintained by each of the PAKUNPK entry points in a communication block that is passed to the string entry points. For the multi-column versions of BLDPK and INTPK this 20-word array is furnished by the caller. For the single column versions of BLDPK and INTPK, and for PACK and UNPACK, the array is local to PAKUNPK.

SUBROUTINE DESCRIPTIONS

The contents of this communication block is as follows:

<u>Word</u>	
1	GINØ reference name
2	type of elements (1=RSP, 2=RDP, etc.)
3	format of strings (0=no trailers)
4	row position of {first} element in string
5	address of element in string
6	number of terms available or actual in string
7	number of terms written in string
8	begin/end flag
9	{ reserved for string routines
10	
11	{ column number
12	
13	count of non-zero words (BLDPK, PACK only)
14	{ branch addresses and indicators
15	
16	(depends on machine)
17	{ not used
18	
19	
20	

3.5.1.5 Design Requirements

1. Let I_j and I_{j+1} be the row positions of two elements supplied in successive calls to BLDPKI or ZBLPKI. Then $I_{j+1} > I_j$ for all j of a column.
2. MCB(2) and MCB(6) must be set to zero prior to the first call to BLDPKN or PACK for a matrix.
3. The exact format of a packed column is machine dependent. See Section 5 for details.

3.5.1.6 Diagnostic Messages

The diagnostic messages are machine dependent. See Section 5 for a detailed listing of possible messages.

MATRIX SUBROUTINE DESCRIPTIONS

THE MATERIAL PREVIOUSLY IN SECTIONS 3.5.2 THRU 3.5.4

HAS BEEN DELETED.

SUBROUTINE DESCRIPTIONS

3.5.5 CALCV (Compute a Partitioning Vector).

3.5.5.1 Entry Point: CALCV.

3.5.5.2 Purpose

To build a partitioning vector of zeros, ones and twos to be used by subroutines MERGE and PARTN.

3.5.5.3 Calling Sequence

CALL CALCV(FILEP,SET1,SUB0,SUB1,CØRE)

FILEP - GINØ file number of partitioning vector - integer - input.

SET1 - Bit position of major set - integer - input.

SUB0 - Bit position of zero subset - integer - input.

SUB1 - Bit position of one subset - integer - input.

CØRE - Open core.

COMMON/PATX/LCØRE,NSUB0,NSUB1,NSUB2,FUSET

LCØRE - Length of open core - integer - input.

NSUB0 - Number of rows in zero subset - integer - output.

NSUB1 - Number of rows in one subset - integer - output.

NSUB2 - Number of rows in two subset (not in one or zero subset) - integer - output.

FUSET - File name of USET - integer - input.

3.5.5.4 Method

Each element of USET is examined and classified. If it belongs to SET1 it is further classified into SUB0, SUB1, and SUB2.

A vector is constructed which has zeros, ones and twos in order as elements of USET are so classified.

3.5.5.5 Design Requirements

LCØRE must be \geq twice length of GINØ buffer.

3.5.5.6 Diagnostic Messages

System messages if USET or FILEP are not correct GINØ files.

MATRIX SUBROUTINE DESCRIPTIONS

3.5.6 PARTN - MERGE (Partition a Matrix - Merge Matrices Together).

3.5.6.1 Entry Point: PARTN, MERGE. PARTN and MERGE are two distinct routines but are so closely related that they are described together here.

3.5.6.2 Purpose

PARTN will break up a matrix into four submatrices.

$$[A] = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

MERGE is the inverse of PARTN in that given the four building blocks A_{11}, \dots, A_{22} MERGE will reconstruct $[A]$.

3.5.6.3 Calling Sequence

CALL PARTN (RP,CP,Z)

CALL MERGE (RP,CP,Z)

RP - Matrix control block of the row partitioning vector - integer - input.

CP - Matrix control block of the column partitioning vector - integer - input.

Z - Array of open core.

If $RP(1) \leq 0$, or $CP(1) \leq 0$, the core locations from $RP(8)$ or $CP(8)$ will contain this vector in packed form.

If $RP(1) = CP(1) < 0$ the arrays RP and CP coincide in core.

COMMON/PARMEG/MCBA(7),MCBA11(7),MCBA21(7),MCBA12(7),MCBA22(7),LCORE,RULE

MCBA - Matrix control block for $[A]$ - input.

MCBA11 - Matrix control block for $[A_{11}]$ - input.

MCBA21 - Matrix control block for $[A_{21}]$ - input.

MCBA12 - Matrix control block for $[A_{12}]$ - input.

MCBA22 - Matrix control block for $[A_{22}]$ - input.

If any submatrix is not desired or does not exist set $MCBA_{ij}(1) = 0$.

LCORE - Length of Z array - integer - input.

RULE - Rule to be applied to the row and column partitioning vectors - integer - input.

SUBROUTINE DESCRIPTIONS

3.5.6.4 Method

Each element of [A] is assigned to the appropriate submatrix by the following schemes.

```

RULE ≥ 0          N = |RULE|
    aij ∈ [A11] if RP(I) = CP(J) = N
    aij ∈ [A21] if RP(I) = N, CP(J) ≠ N
    aij ∈ [A12] if RP(I) ≠ N, CP(J) = N
    aij ∈ [A22] if RP(I) ≠ N, CP(J) ≠ N

RULE < 0          N = |RULE|
    aij ∈ [A11] if RP(I) ≥ N, CP(J) ≥ N
    aij ∈ [A21] if RP(I) ≥ N, CP(J) < N
    aij ∈ [A12] if RP(I) < N, CP(J) ≥ N
    aij ∈ [A22] if RP(I) < N, CP(J) < N
  
```

Subroutine RULER (RULE,IP,ZCØNT,ØNCNT,LIST,NRØWP,BUFF,IØPT) is called twice to accomplish this assignment where

```

RULE - Rule to be applied

IP    - Either RP or CP

ZCØNT - Number of elements (row or column) assigned to the one class.

ØNCNT - Number of elements (row or column) assigned to the two class.
  
```

For example, if RULER is analyzing RP and RP(I) = N this element of RP is said to belong to the 1 class in that it will go either to [A11] or [A21].

```

LIST - A list of zeros and ones. Zero, if the element belongs to the one class.
      One, if the element belongs to the two class.

NRØWP - Number of rows in IP

BUFF - One GINØ buffer space

IØPT - If IØPT = 1, LIST will be stored 1 number per word. If IØPT = 0, LIST will be
      packed 32 bits/word.
  
```

Non-zero elements are read, classified and output.

MATRIX SUBROUTINE DESCRIPTIONS

3.5.6.5 Design Requirements

Open core must contain n GINØ buffers + 1 column (single precision) of $[A]$ and 1 row/32 of $[A]$, where n = the number of submatrices present plus one.

3.5.6.6 Diagnostic Messages

If insufficient core is available as described above, fatal message 3008 is given.

If illegal input is detected, fatal message 3007 is given.

SUBROUTINE DESCRIPTIONS

3.5.7 SSG2A (Driver for PARTN).

3.5.7.1 Entry Point: SSG2A.

3.5.7.2 Purpose

To partition a vector into two subsets (i.e., to be a driver for PARTN).

3.5.7.3 Calling Sequence

CALL SSG2A(VECTOR,PART1,PART2,PVECT)

VECTOR - GINØ file number of vector to be partitioned - integer - input.

PART1 - GINØ file number of major partition - integer - input.

PART2 - GINØ file number of minor partition - integer - input.

PVECT - GINØ file number of partitioning vector - integer - input.

COMMON/PATX/XXX,NRØW1,NRØW2

NRØW1 - Number of rows in PART1 - integer - input.

NRØW2 - Number of rows in PART2 - integer - input.

3.5.7.4 Method

The PARTN common block is filled.

Based on the trailer of VECTOR and NRØW1, NRØW2:

$$\{\text{VECTOR}\} \Rightarrow \begin{Bmatrix} \text{PART1} \\ \text{PART2} \end{Bmatrix}$$

3.5.7.5 Design Requirements

Open core is needed at /SSGA2/.

MATRIX SUBROUTINE DESCRIPTIONS

3.5.8 SDR1B (Driver for MERGE).

3.5.8.1 Entry Point: SDR1B

3.5.8.2 Purpose

To drive MERGE forming VECTOR

$$\{\text{VECTOR}\} = \left\{ \frac{\text{PART1}}{\text{PART2}} \right\}$$

3.5.8.3 Calling Sequence

CALL SDR1B(PVECT,PART1,PART2,VECTOR,MAJOR,SUB0,SUB1,USET,IØPT,IYS)

PVECT - GINØ name of partition vector - integer - input.

PART1 - GINØ name of vector which corresponds to SUB0 set - integer - input.

PART2 - GINØ name of vector which corresponds to SUB1 set - integer - input

VECTOR - GINØ name of merged vector - integer - input.

MAJOR - Bit position of set of VECTOR - integer - input.

SUB0 - Bit position of set of PART1 - integer - input.

SUB1 - Bit position of set of PART2 - integer - input.

USET - GINØ name of USET - integer - input.

IØPT - '0' $\left\{ \begin{array}{l} \text{These are used in a module specific call to} \\ \text{handle the YS data block in a special manner.} \end{array} \right.$

IYS - '0'

3.5.8.4 Method

CALCV is called to obtain partitioning vector.

PARMEG common block is filled.

MERGE is called.

3.5.8.5 Design Requirements

Open core at /SDRB1/.

SUBROUTINE DESCRIPTIONS

3.5.9 UPART (Symmetric Partition Driver).

3.5.9.1 Entry Points: UPART, MPART

3.5.9.2 Purpose

To compute a partitioning vector and then perform a series of symmetric partitions. A symmetric partition is such that the row partitioning vector equals the column partitioning vector.

For example:

$$[K_{nn}] = \begin{bmatrix} K_{ff} & | & K_{fs} \\ \hline K_{sf} & | & K_{ss} \end{bmatrix}$$

3.5.9.3 Calling Sequence

CALL UPART(USET,SCR1,MAJØR,SUB0,SUB1)

USET - GINØ file number of USET - integer - input.

SCR1 - Scratch file on which the partitioning vector will be written - integer - input.

MAJØR - Bit position within a USET word of the super set (e.g., n set in the above example) - integer - input.

SUB0 - Bit position of the first subset (e.g., f set in the above example) - integer - input.

SUB1 - Bit position of the second subset (e.g., s set in the above example) - integer - input.

CALL MPART(KNN,KFF,KSF,KFS,KSS)

KNN - GINØ name of the matrix to partitioned - integer - input.

KFF,KSF,KFS,KSS - GINØ names of the partition outputs. A zero will cause the respective matrix not to be written.

3.5.9.4 Method

A call to UPART causes CALCV to compute a partitioning vector.

MPART drives PARTN and is called repeatedly to partition several matrices (i.e. KNN, MNN, BNN, K4:IN) in a similar symmetric manner using the same partitioning vector.

3.5.9.5 Design Requirements

Open core at /UPARTX/.

MATRIX SUBROUTINE DESCRIPTIONS

3.5.10 ADD (Driver for SADD)

3.5.10.1 Entry Point: ADD.

3.5.10.2 Purpose

To drive SADD to compute $[X] = \alpha[A] + \beta[B]$ or on option $[X] = \alpha[A]$.

3.5.10.3 Calling Sequence

CALL ADD (Z)

Z -- Array of core

COMMON/ADDX/MCBA(7),MCBB(7),MCBC(7),TYPALPHA(4),LCORE,TYPB,BETA(4)

MCBA - Matrix Control Block for [A] - input.

MCBB - Matrix Control Block for [B] - input.

MCBC - Matrix Control Block for [X] - input.

TYPALPHA - Type of Alpha - integer - input.

1 - real single precision

2 - real double precision

3 - complex single precision

4 - complex double precision

ALPHA - α - input - type depends on TYPALPHA.

LCORE - Length of Z array.

TYPB - Type of BETA - integer - input.

BETA - β - input - type depends on TYPB.

3.5.10.4 Method

ADD rearranges and moves /ADDX/ to /SADDX/, and calls SADD

to compute [X] in above equation.

3.5.10.5 Design Requirements

Matrix add routine ADD is replaced by SADD. The revised ADD routine is kept in the system to accommodate the existing calls to the ADD routine.

However, all future calls to matrix addition should be made directly to SADD.

3.5.11 SSG2C (Driver for ADD).

3.5.11.1 Entry Point: SSG2C.

3.5.11.2 Purpose

To drive ADD to compute $[C] = \alpha[A] + \beta[B]$.

3.5.11.3 Calling Sequence

CALL SSG2C (FILEA,FILEB,FILEC,IØP,BLØCK)

FILEA - GINØ file number of [A] - integer - input.

FILEB - GINØ file number of [B] - integer - input.

FILEC - GINØ file number of [C] - integer - input.

IØP - Option flag - integer - input.

IF IØP < 0 the first column of [A] will be added to each column of [B] to give [C].

BLØCK - 11-word array containing coefficients - input.

<u>Word</u>	<u>Type</u>	<u>Meaning</u>
1	Integer	Type of α
2	Real	α
3		α
4		α
5		α
6		Not used
7	Integer	Type of β
8	Real	β
9		β
10		β
11		β

MATRIX SUBROUTINE DESCRIPTIONS

3.5.11.4 Method

The trailers of FILEA and FILEB and BLØCK are used to fill the ADDX common block.

The type of FILEC is the minimum type compatible with $\alpha[A]$ and $\beta[B]$.

3.5.11.5 Design Requirements

Open core at /SSGC2/.

3.5.12 MPYAD (Matrix Multiplication Routine)

3.5.12.1 Entry Point: MPYAD

3.5.12.2 Purpose

To evaluate the matrix equation

$$D = \pm [A][B] \pm [C] \text{ or } D = \pm [A]^T[B] \pm [C].$$

3.5.12.3 Calling Sequence

CALL MPYAD(Z,Z,Z)

COMMON/MPYADX/A(7),B(7),C(7),D(7),NZ,T,SIGNAB,SIGNC,PREC,SCR

Z - An area of working storage.

NZ - The number of computer words at Z.

A,B,C - Matrix control blocks for the matrices A, B, C.

If C(1) = 0, C is not used, i.e., $[D] = \pm [A][B]$ or $\pm [A]^T[B]$.

D - Matrix control block for the product matrix. The trailer is not written by MPYAD.

D(1) must contain the GINØ file name prior to entry.

D(5) must contain the arithmetic type of the elements of D.

D(2), D(6) and D(7) are initialized and accumulated in MPYAD.

$$T \begin{cases} = 0, \pm [A][B] \pm [C] \text{ is computed.} \\ \neq 0, \pm [A]^T[B] \pm [C] \text{ is computed.} \end{cases}$$

$$\text{SIGNAB} = \begin{cases} +1, \text{ compute } +[A][B] \text{ or } +[A]^T[B] \\ 0, \text{ the product } [A][B] \text{ is null} \\ -1, \text{ compute } -[A][B] \text{ or } -[A]^T[B] \end{cases}$$

$$\text{SIGNC} = \begin{cases} +1, \text{ use } +C \\ -1, \text{ use } -C \end{cases}$$

Note: If C(1) = 0, SIGNC is ignored.

$$\text{PREC} = \begin{cases} 0, \text{ perform arithmetic in double precision if A, B or C is double precision} \\ 1, \text{ perform arithmetic in single precision} \\ 2, \text{ perform arithmetic in double precision} \end{cases}$$

SCR = GINØ file name of a scratch file for use by MPYAD.

MATRIX SUBROUTINE DESCRIPTIONS

3.5.12.4 Method

1. General Comments. Three alternative methods of performing the matrix multiplication are available in MPYAD. Method One holds as many unpacked columns of the B and D matrices as core storage will allow. The A matrix is read interpretively by INTPK. For each non-zero element in A, all combinatorial terms for columns of B currently in core are computed and accumulated in the storage for D. At the completion of one pass of the A matrix, the matrix product is complete to the extent of the number of columns of B currently in core. (If the C matrix is present, columns are initially unpacked into the storage for D.) The process is repeated until the B matrix is exhausted. One GINØ buffer only is required for Method One. The number of passes of the A matrix for Method One equals the number of columns of B divided by the number of columns of B and D which can be held in core at one time. In Method Two either one element of B ($T = 0$) or one column of B in unpacked form ($T = 1$) is held in core at one time, and either one column of D in unpacked form ($T = 0$) or one element of D ($T = 1$). The remaining storage is allocated to storage of columns of A in packed form (i.e., non-zero terms and row positions only). For all the columns of A in storage at one time the B and E matrices are passed, column by column, forming partial answers on each pass. The E matrix is initially the C matrix (if present) and thereafter is the partial product matrix from the previous pass. Three GINØ buffers are required for Method Two. It may be seen that the A matrix is passed once and the number of passes of the B and E matrices equals the number of columns of A divided by number of columns of A that may be held in core at one time. Method Three is provided for the transpose case only. A pass is initiated by reading and storing rows of the A matrix in unpacked form. For all rows of A in storage at one time the B and E matrices are passed, column by column, forming partial answers on each pass. The E matrix is initially zero and thereafter is the partial product matrix from the previous pass. The C matrix (if present) is added on the last pass. In this way, the E matrix is only unpacked to the number of the last row of A in core. The B matrix is processed directly from the buffer through use of the string routines. Each non-zero term in B is multiplied by terms in the corresponding rows of A currently in core and the products are accumulated in the storage for E. Four GINØ buffers are required.

2. Initialization Phase. The arithmetic type of the elements of D is determined as a function of the types of A, B and C and the precision requested by the user. Various parameters for Methods One, Two and Three are computed and used to estimate the core storage required and execution times. These parameters include:

SUBROUTINE DESCRIPTIONS

N_A - number of words/row in [A].

N_B - number of words/row in [B].

N_D - number of words/column (non-transpose case) or number of words/row (transpose case) for [D] or [A].

$\left. \begin{matrix} M_A \\ M_B \\ M_C \end{matrix} \right\}$ - order (row * column) of [A], [B] and [C] respectively.

$J = \frac{C}{N_B + N_D}$ - number of columns and rows per pass. Fatal exit if less than one. C is open core minus one system buffer.

P - number of words/element in [D].

$\left. \begin{matrix} \rho_A \\ \rho_B \\ \rho_C \end{matrix} \right\}$ - density of [A], [B] and [C] respectively.

ρ_D - density of [D] estimated as maximum of ρ_A and ρ_B

$\left. \begin{matrix} C_A, R_A \\ C_B, R_B \\ C_C, R_C \end{matrix} \right\}$ - number of columns (C) and rows (R) for [A], [B] and [C] respectively.

$T_m, T_{ml}, T_i, T_u, T_p, T_{bp}$ - System timing constants for Multiply-tight, Multiply-loose, INTPK, UNPACK, PACK and Buffer-pack respectively.

$T_{pC} = M_C (0.05 + 0.95 \rho_C) T_u$ - unpack time for [C]. 0.0 if no [C].

$T_{mA} = M_A (T_m + (1 - \rho_A) T_{ml})$ - multiply time for [A].

$T_{iA} = (R_A \rho_A + 5) C_A T_i$ - unpack time for [A].

$T_{uB} = C_A C_B (0.05 + 0.95 \rho_B) T_u$ - unpack time for [B].

$T_{pD} = R_A C_B (0.1 + 0.9 \rho_D) T_p$ - pack time for [D].

For Method One the time is estimated by:

$N_1 = \frac{C_B - 1}{J}$ - integerized number of passes.

$T_1 = M_A C_B \rho_A T_m + N_1 T_{iA} + T_{uB} + T_{pD} + T_{uC}$

MATRIX SUBROUTINE DESCRIPTIONS

For Method Two the core and time are calculated by:

$$C = \text{open core} - 3 \text{ buffers} - N_D \text{ or } N_B \text{ (} T \neq 0 \text{)}$$

$$N_2 = \frac{(2 - \rho_A) M_A \rho_A}{C/P} - \text{integerized number of passes}$$

$$K = \text{maximum number of nonzero words/column for [B]}$$

$$P_2 = \text{minimum} \left(\frac{M_A \rho_B}{K} + C_B, M_A \rho_B, C_B \right)$$

$$C_2 = C_B (R_A \rho_C + 5) - \text{if [C] is present, 0.0 otherwise}$$

$$T_2 = P_2 \rho_A T_{mA} + C_B (R_A \rho_A + 5) T_i + N_2 T_{uB} + \frac{N_2 + 1}{2} (R_A \rho_D + 5) C_B T_{bp} + \frac{N_2 - 1}{2} (R_A \rho_D + 5) T_i + C_2$$

where the length of the strings are assumed proportional to ρ_A .

Method Three is only calculated for the $[A]^T$ case as follows:

$$C = \begin{cases} \text{open core} - 3 \text{ buffers (no [C])} \\ \text{open core} - 4 \text{ buffers (with [C])} \end{cases}$$

$$P_3 = \text{minimum} \left(\frac{C - M_A}{R_A P}, C_A \right)$$

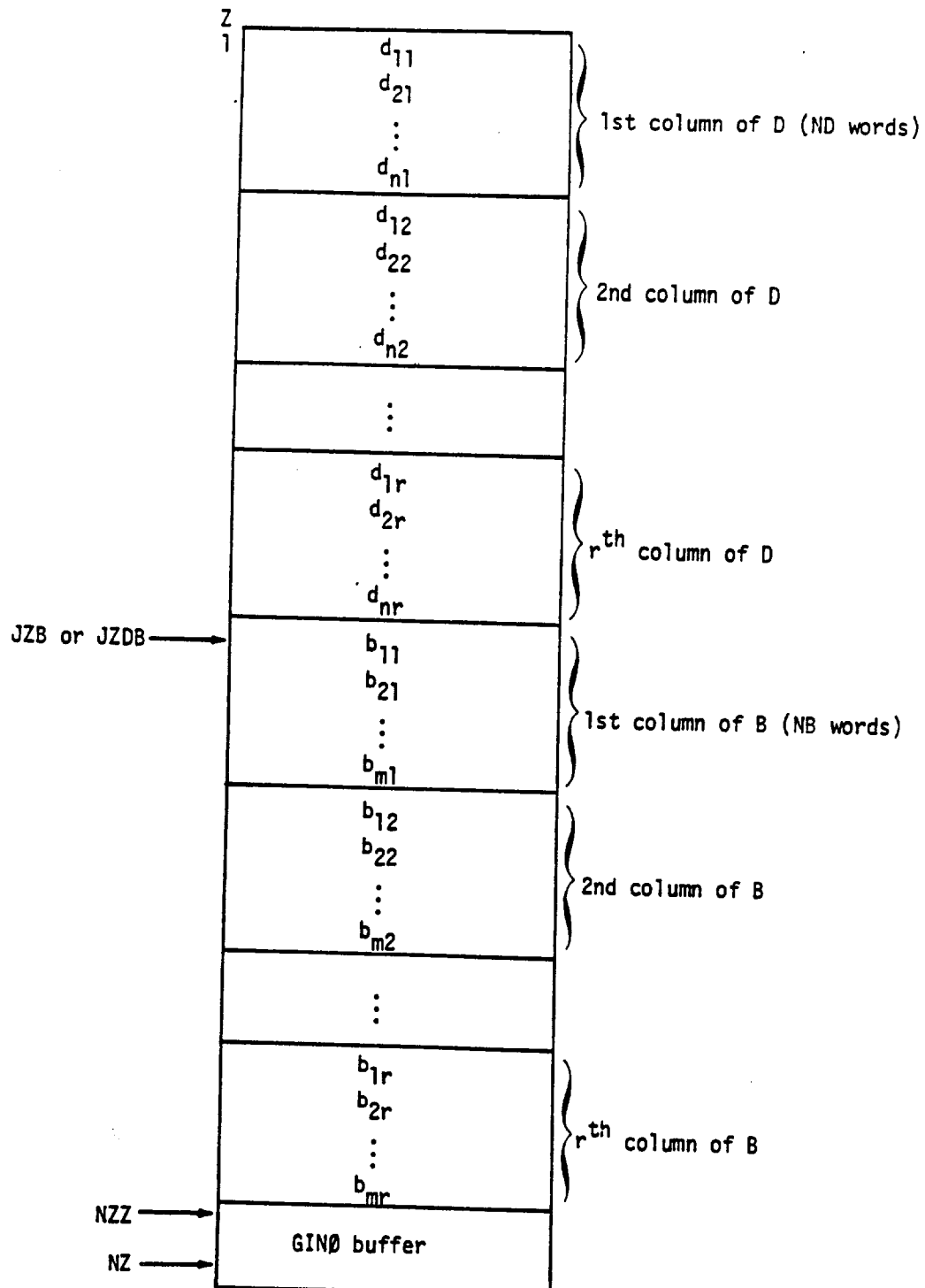
$$N_3 = \frac{C_A - 1}{P_3} + 1 - \text{integerized number of passes}$$

$$T_3 = M_A C_B \rho_B T_m + M_A (0.05 + 0.95 \rho_A) T_u + N_3 (C_A \rho_B + 5) T_i + \frac{N_3 + 1}{2} T_{pD} + \frac{N_3 - 1}{2} (R_A \rho_D + 10) C_B T_i + T_{uC}$$

These time estimates, T_i , and the number of passes may be printed for each of the methods by setting DIAG 19. Note that CPU and not I/O time is used to arrive at these estimates. The method with the minimum time estimate or the method selected by number with SYSTEM(58) is used. By inspection of the equations for T , the running time is proportional to ρ_A for Method One (where ρ_B considered full), is proportional to the product $\rho_A * \rho_B$ for Method Two, and is proportional to ρ_B for Method Three where sparsity of ρ_A is not considered. These three methods are described below.

SUBROUTINE DESCRIPTIONS

3. Method One. The allocation of core storage for Method One is shown below:



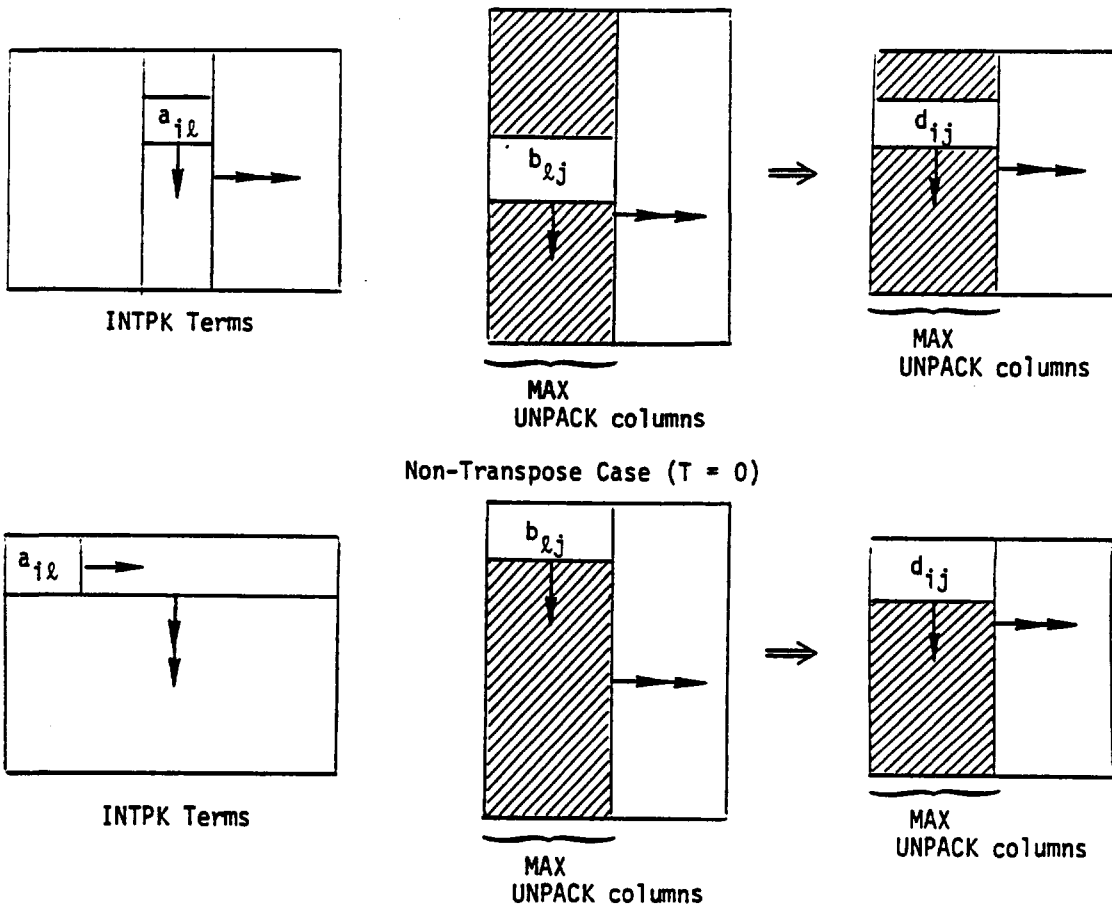
MATRIX SUBROUTINE DESCRIPTIONS

Columns of B and C are read and unpacked by UNPACK. INTPK is called to initiate reading and interpreting the ℓ th column of A ($\ell = 1$ initially). For each non-zero in A, $a_{i\ell}$ or $a_{\ell i}$ depending on T, the following arithmetic computations are made:

$$T = 0: \quad d_{ij} = a_{i\ell} b_{\ell j} + d_{ij}$$

$$T \neq 0: \quad d_{\ell j} = a_{\ell i} b_{ij} + d_{\ell j}$$

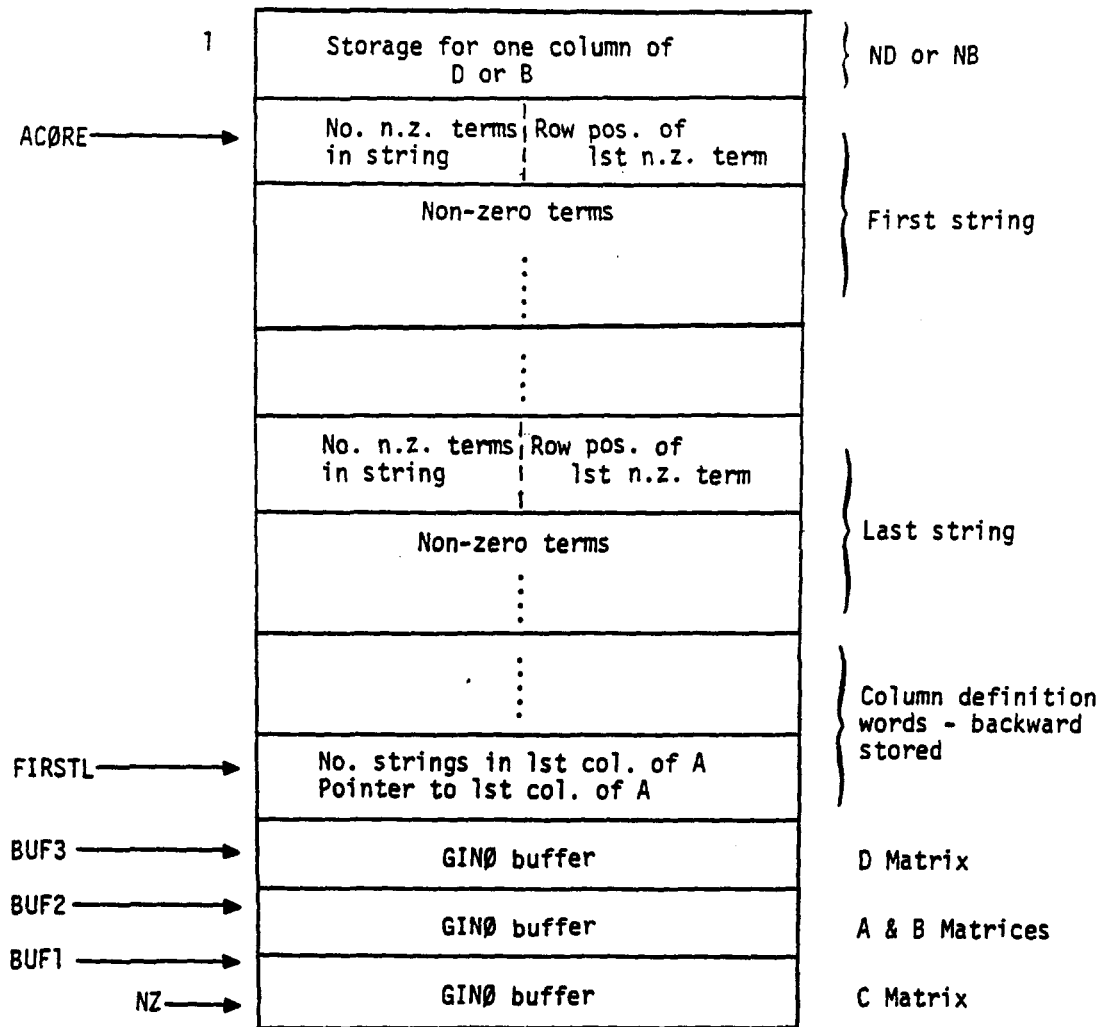
where j runs across the columns of B and D currently in core. At the conclusion of a pass of the A matrix, the columns of D in core are packed and written by PACK. The process is repeated until the multiplication is complete. The scheme for each is diagrammed below.



4. Method Two. The allocation of core storage for Method Two is shown on the next page.

SUBROUTINE DESCRIPTIONS

Z



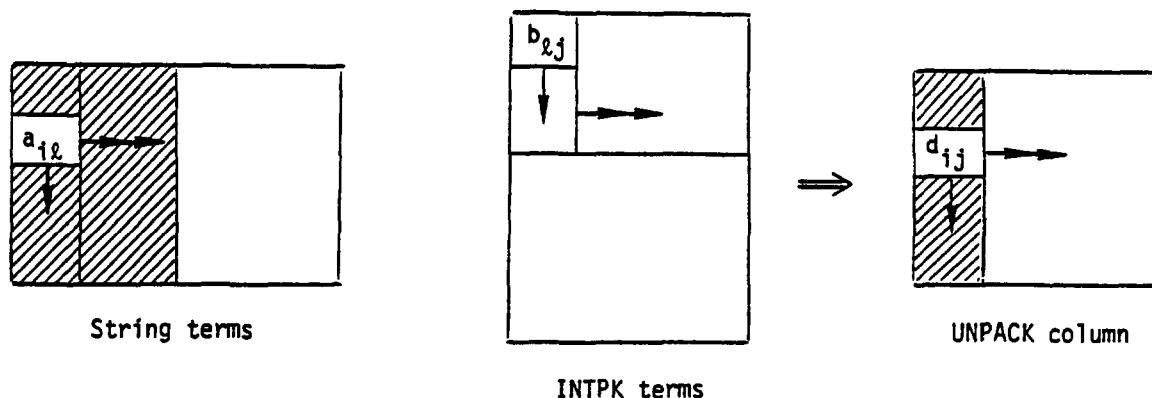
MATRIX SUBROUTINE DESCRIPTIONS

To begin each pass of Method Two, as many columns of A that can be held in core are read using INTPK and stored in core in strings. Each string consists of a string definition word followed by consecutive terms of a column such that no two consecutive terms are zero. For each column there exists a pair of column definition words -- one points to the first string in the column and the other defines the number of strings in the column. The number of passes is determined by the size and density of the A matrix.

The following operations are performed for the nontranspose case:

1. UNPACK is called to unpack the next column of C into the D matrix area.
2. INTPK is called to read the non-zero terms of the corresponding column of B.
3. MPY2NT is called to perform the operation $d_{ik} = a_{ij} b_{jk} + c_{ik}$. Each non-zero element of B (b_{jk}) will combine with all non-zero elements in the j^{th} column of A and add to the corresponding elements in the k^{th} column of D in core.
4. When all columns of A in core are complete, the column of D is packed and written by PACK.
5. When all columns of B and C are complete, a test for completion of the multiplication is made.
6. If incomplete, the C and D files are switched and the process described above is repeated.

This scheme is diagramed below.

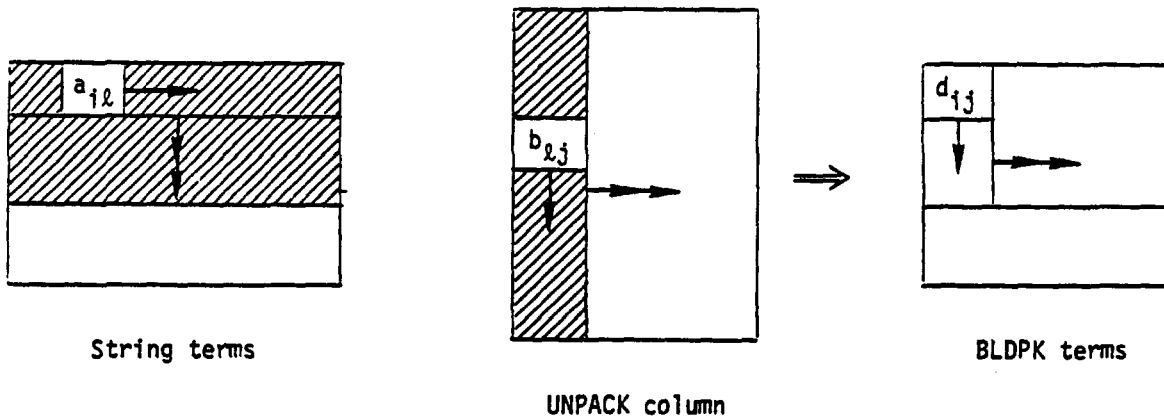


SUBROUTINE DESCRIPTIONS

The following operations are performed for the transpose case:

1. UNPACK is called to unpack the next column of B into core.
2. INTPK is called to read the non-zero terms for the corresponding column of C.
3. BLDPK is called to initiate the packing of a column of D.
4. MPY2T is called to perform the operation $d_{ik} = \sum a_{ij} b_{jk} + c_{ik}$ for each row of A in core.
5. The elements of D are packed using ZBLPKI.
6. When all columns of B and C are complete, a test for completion of the multiplication is made.
7. If incomplete, the C and D files are switched and the process described above is repeated.

This scheme is diagramed below.



If, after completion of Method Two, there has been an even number of passes of the B matrix, FILSWI is called to switch the D matrix from a scratch file to its assigned unit.

5. Method Three. To begin each pass of Method Three, as many rows of A that can be held in core are read using UNPACK. The number of passes is determined by the total number of terms in the A matrix and the number of terms of available core.

The following operations are performed:

1. UNPACK is called to unpack the next column of E in the D matrix area (except on first pass in which case the D area is set to zero).
2. If last pass, the next column of C is read interpretively (if present) and added to the D terms currently in core.

MATRIX SUBROUTINE DESCRIPTIONS

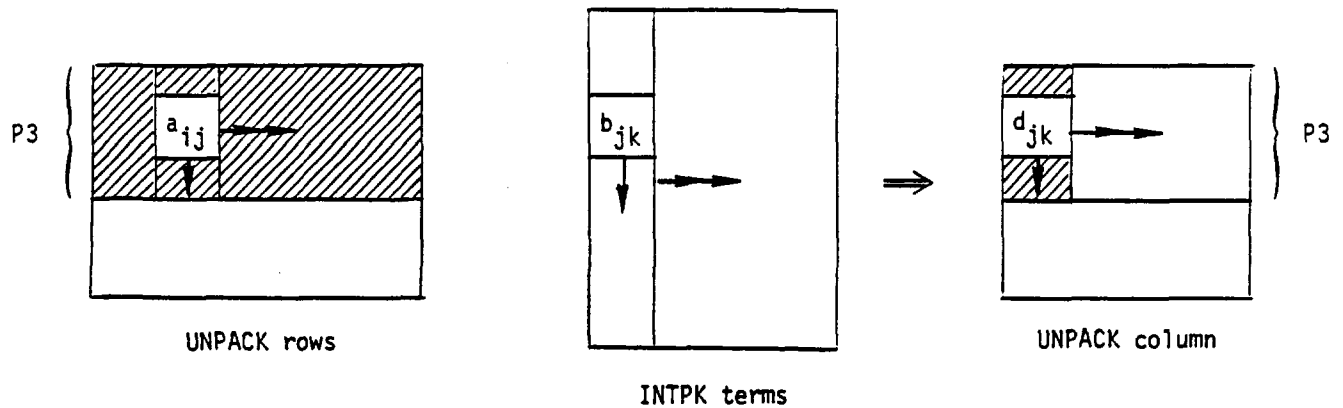
3. MPY3T is called to perform the operation

$$d_i = d_i + a_{ij}b_j$$

for all non-zero elements in the current column of B and all rows of A currently in core.

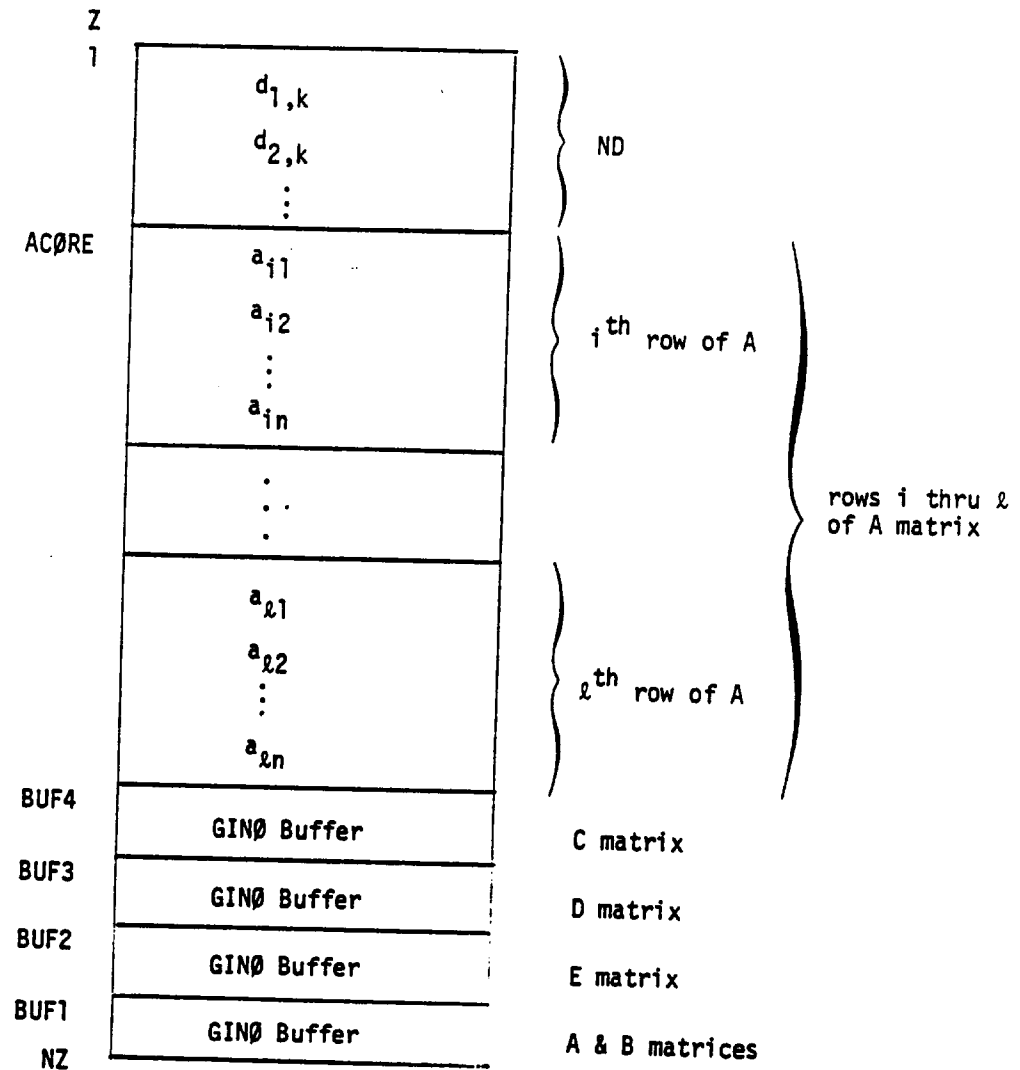
4. The current column of D is packed in the D file.
5. When all columns of B and E are complete, a test for end of the multiplication is made.
6. If incomplete, the D and E files are switched and steps 1 through 5 are repeated.

The scheme is diagramed below.



SUBROUTINE DESCRIPTIONS

The allocation of core storage used above is shown in the following figure:



MATRIX SUBROUTINE DESCRIPTIONS

3.5.12.5 Auxiliary Subroutine MPYQ

1. Entry Points: MPY1, MPY2NT, MPY2T, MPY3T
2. Purpose: MPYQ is called once per execution of MPYAD. It performs general initialization for each of the other entry points.

MPY1, MPY2NT, MPY2T and MPY3T perform the inner loops for Method One, Method Two (non-transpose), Method Two (transpose) and Method Three (transpose) respectively. For efficiency these routines are written in assembly language for each of the machines. A FORTRAN version of MPY3T is also available.

3.5.12.6 Auxiliary Subroutine FILSWI

1. Entry Point: FILSWI
2. Purpose: Auxiliary subroutine to switch unit reference numbers in /XFIAT/ in the event that the product matrix in Method Two ends up on a scratch file (even number of passes of B matrix).

3.5.12.7 Design Requirements

Core storage must be sufficient to hold one unpacked column of B plus one unpacked column of D plus one GINØ buffer.

The matrices to be multiplied (and added) must have compatible dimensions. MPYAD checks for this condition.

3.5.12.8 Information Messages

CØNMSG is called at entry and at exit from MPYAD. Consequently, the line xxxxx MPYAD will appear twice for each call to MPYAD (where xxxxx = time in seconds). The difference is the execution time for MPYAD.

MPYAD method selection data is printed under control of DIAG 19.

2102 LEFT-HAND MATRIX RØW PØSITION _____ ØUT ØF RANGE - IGNØRED.

A term in the A matrix whose row position is larger than the stated dimension was detected and ignored.

SUBROUTINE DESCRIPTIONS

3.5.12.9 Diagnostic Messages

The following messages may be issued by MPYAD:

3001 - undefined file when opening

3002 - end of file encountered

3008 - insufficient core

3050 - insufficient time

3055 - incompatible matrices

3.5.13 SSG2B (Driver for MPYAD).

3.5.13.1 Entry Point: SSG2B.

3.5.13.2 Purpose

To drive MPYAD to compute

$$[D] = \pm [A] [B] \pm [C]$$

or

$$[D] = \pm [A]^T [B] \pm [C]$$

3.5.13.3 Calling Sequence

CALL SSG2B(FILEA,FILEB,FILEC,FILED,T,PREC,ISIGN,SCR1)

FILEA - GINØ name of [A] - integer - input.

FILEB - GINØ name of [B] - integer - input.

FILEC - GINØ name of [C] - integer - input.

FILED - GINØ name of [D] - integer - output.

T - Transpose flag - integer - input.

T = 0 implies use [A]

T = 1 implies use $[A]^T$

PREC - Precision of computation - integer - input. 1 = real single precision,
2 = real double precision, 3 = complex single precision, 4 = complex double
precision.

ISIGN - Sign of products - integer - input.

$$ISIGN = \pm 1 \Rightarrow \begin{cases} \text{sign } [AB] = \text{sign } (ISIGN) \\ \text{sign } [C] = \text{sign } (ISIGN) \end{cases}$$

$$|ISIGN| > 1 \Rightarrow + [AB] - [C]$$

$$|ISIGN| < 1 \Rightarrow - [AB] + [C]$$

SCR1 - GINØ scratch file - integer - input.

3.5.13.4 Method

SSG2B fills /MPYADX/ and calls MPYAD to compute [D] in above equation.

3.5.13.5 Design Requirements

Open core at /SSGB2/.

MATRIX SUBROUTINE DESCRIPTIONS

3.5.13.6 Diagnostic Messages

The following message may be issued by SSG2B: 2363.

MATRIX SUBROUTINE DESCRIPTIONS

THIS PAGE HAS BEEN LEFT BLANK INTENTIONALLY.

SUBROUTINE DESCRIPTIONS

3.5.14 SDCOMP (Symmetric Decomposition)

3.5.14.1 Entry Point: SDCOMP

3.5.14.2 Purpose

To decompose a symmetric matrix $[A]$ into the form $[A] = [L][D][L]^T$ where $[L]$ is a unit lower triangular matrix and $[D]$ a diagonal matrix stored in place of the unit elements on the diagonal of $[L]$. On option, the Cholesky decomposition $[A] = [C][C]^T$ is done for a real, positive definite matrix, with only the lower triangle $[C]$ being output. SDCOMP will also compute the determinant of $[A]$ and minimum diagonal element of $[L]$.

3.5.14.3 Calling Sequence

CALL SDCOMP (\$n₁,Z,Z,Z)

COMMON/FACT/A(7),L(7),Q(7),SCR1,SCR2,NZ,DETR,DETI,PPOWER,SCR3,MINDIA,CHLSKY

- A(7) - Matrix control block for $[A]$
- L(7) - Matrix control block for $[L]$ or $[C]$
- Q(7) - Q(1) = SCR4, Q(2)...Q(7) not used
- SCR1, SCR2
SCR3, SCR4 - Four scratch files.
- NZ - The number of computer words at Z.
- DETR, DETI - Double precision cells where the scaled value of the determinant of $[A]$ will be stored
- PPOWER - Scale factor to be applied to DET (Determinant = DET*10**PPOWER).
- MINDIA - Double precision cell where the absolute value of the minimum diagonal element of $[A]$ will be stored
- CHLSKY - When CHLSKY = 1, form $[C]$
- Z - An area of working storage.
- n₁ - Statement number to which control is transferred if the decomposition fails.

3.5.14.4 Method

The following discussion will be based on the equation

$$[A] = [L][U] \tag{1}$$

MATRIX SUBROUTINE DESCRIPTIONS

where [L] is a unit lower triangle. The elements of the upper triangle may be computed by the following recursion formula:

$$u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{jk} u_{ki} \quad (2)$$

For symmetric matrices without pivoting, the upper and lower triangular elements are related as follows:

$$l_{jk} = \frac{u_{kj}}{u_{kk}} \quad (3)$$

Substituting the relation in Equation 3 into Equation 2 gives:

$$u_{ij} = a_{ij} - \sum_{k=1}^{i-1} \frac{u_{kj}}{u_{kk}} u_{ki} \quad (4)$$

Now, $k < i \leq j$, so that only previously computed results occur on the right-hand side of Equation 4, if the elements u_{ij} are computed in order, starting with the first row.

For the first row, $u_{ij} = a_{ij}$, as the second term in Equation 4 is zero. Next, the second term within Equation 4 is determined for $k = 1$ with i and j taking on all values greater than 1, for which there are non-zero terms, u_{ij} , in the first row of matrix [A]. The results are stored in the ij positions. This results in a triangular array having a side dimension equal to one less than the number of active columns, which is the same as the number of non-zero terms in the first, or more generally the pivotal, row. The locations of the non-zero terms for the preceding operations are shaded in Figure 1 for a particular matrix. The X's indicate the locations of non-zero terms in the original matrix and the O's indicate the locations of terms that are initially zero, but become non-zero as the decomposition proceeds. The shaded locations in the first row form the pivotal row and the remaining shaded locations are the contributions from the second term of Equation 4. A compact storage array is shown in Figure 2 with the shaded area. The integers in the first row are the numbers of the active columns.

The second row of [U] is obtained by adding the second row of the matrix [A] to the previously computed results in the first row of the triangular array, which is the second shaded row in Figure 2. The next step is to calculate the contributions from the second half of Equation 4 for $k = 2$, and with i and j taking on all values greater than 2 associated with non-zero columns of

SUBROUTINE DESCRIPTIONS

[U] for the second row. The results are combined with previously computed results in the ij positions. The locations of the non-zero operations when the second row is pivotal are shown as shaded areas in Figure 3. The equivalent compact storage array is shown in Figure 4. The integers in the first shaded row are the numbers of the current active columns.

The third row of [U] is determined by adding the third row of the matrix [A] to the associated terms in the current first row of the previously computed triangular array. These terms are located in the second shaded row of Figure 4. This process continues with k taking on all values from 1 to $N - 1$, where N is the order of the matrix.

It is clear from the previously described procedure that if a triangular array having a side dimension equal to the number of active columns is available in main storage, the decomposition can proceed in a single pass without using secondary storage for any of the intermediate matrix operations. The number of active columns is always equal to the number of non-zero terms in the pivotal row, which is identified with the index of k in Equation 4. It is clear that in general the number of active columns will vary as the decomposition proceeds, and k takes on all values from 1 to $N - 1$. This means that the actual positions within the triangular array for particular values of i and j will be different for each pivotal row, or for each value of k .

It should be noted that the number of active columns can increase by any number, up to a total number of $N - k$ at any point of the decomposition. However, if it is assumed that the number of active columns will never decrease by more than one for each new pivotal row, it is possible to store the current calculations dynamically in the same array with previous calculations without interference. This is equivalent to assuming that once a column becomes active that it remains active until the column intersects the diagonal (column number equals pivotal row number). This assumption will not cause errors in the calculations, but will result in the performance of a number of zero operations and require additional working space in main storage.

If at some point in the decomposition the diagonal term of the pivotal row initiates a new active column, all active columns will terminate in the previous row (change status from active to passive). The shaded part of Figure 7 indicates the non-zero terms in the upper triangular factor, where the original non-zero terms are indicated with X's. Columns 7, 9, and 13 are terminated at row 3, and columns 11 and 14 are terminated at row 6. New active columns will be activated only if the original matrix contains non-zero terms in the current pivotal row (columns 5 and 11 of row 4, or column 8 of row 7 in Figure 7). Columns will be reactivated (change status from passive to

MATRIX SUBROUTINE DESCRIPTIONS

active) if the second term in Equation 4 has previously made a non-zero contribution to the current pivotal row (columns 9 and 13 of row 7 in Figure 7).

An allowance is made for the termination of active columns at the expense of writing the contents of the working area (passive elements) on a secondary storage file in order to save these intermediate results until a later stage of the decomposition. Figure 8 indicates the positions of non-zero terms when row 3 is the pivotal row. The shaded terms in rows 7, 9, and 13 are written on a secondary file prior to operations with row 4 as a pivotal row. These terms are not needed until each of the rows 7, 9, and 13 become pivotal.

Figure 9 indicates the positions of non-zero terms when row 6 is the pivotal row. The shaded terms in rows 11 and 14 are merged with those of rows 7, 9, and 13 already existing on secondary storage. When row 7 becomes pivotal, the contents of the secondary file for row 7 is added to those of row 7 for matrix [A]. The contents of this secondary file will be referred to as passive elements, whereas the contents of the working area in main storage are referred to as active elements.

The actual decomposition is preceded by the generation of an active column vector for each pivotal row. The active column vector contains the column number for each active column and is generated as follows:

1. The current active column vector (AC) for a particular pivotal row is a union of all column numbers from the previous active column vector (except the first entry), the column numbers of the non-zero terms in the current row of the matrix [A], and the column numbers for non-zero terms associated with the previous termination of active columns (passive elements become active).
2. If a zero diagonal term is encountered, the row number is saved, and the rest of the matrix scanned for zero diagonal terms. The total number of zero diagonals and the row numbers of the first twenty are printed as a user information message. SDCOMP then returns to the calling program via the non-standard return.
3. New active columns are identified by setting the column numbers negative when they appear in the active column vector.

SUBROUTINE DESCRIPTIONS

4. Secondary storage required for intermediate matrix operations is determined. No spill is required if

$$2C + \frac{(C)(C-1)}{2} \leq W \quad (5)$$

where C equals the number of entries in AC and W equals working space for calculation of the triangular factor. The value of W assumes 5 buffers.

5. The active column vector is written on a scratch file along with the pivot row. The maximum number of active columns, as well as a running sum is saved in order to calculate the average number of active columns. Also, the maximum number of passive columns, as well as a running sum of the number at each active column termination point, is saved in order to calculate the average number of passive columns. The maximum and average number of active and passive columns is output as a user information message.
6. If spill is required, the number of rows, S, that can be held in W for the current number of active columns, is calculated according to the following equation:

$$2C + S(C-1) - \frac{(S)(S-1)}{2} = W \quad (6)$$

The end of the current spill group is the column number that is S-positions beyond the beginning of AC for the first row of the spill group.

7. If the number of active columns in any row of the spill group is greater than the number of active columns in the first row of the spill group, a check is made for each such row for a possible reduced value of the number of rows that can be held in W for the spill group. S_c is calculated from the following expression:

$$2C_c + S_c(C_c - 1) - \frac{S_c(S_c - 1)}{2} = W \quad (7)$$

where C_c equals the number of active columns in the current row of the spill group. If

$$S_c \geq C_s \quad (8)$$

MATRIX SUBROUTINE DESCRIPTIONS

where C_s is the number of active columns in the current row in the range of the spill group, no reduction in the number of rows in the current spill group is necessary. Otherwise, the range of the spill group is reduced to satisfy Equation 8.

8. If the following conditions are all satisfied, all active columns are terminated in the previous row and they become passive columns.

$$AC(2) - AC(1) \geq CL\emptyset SE \quad (9)$$

$$C_c \geq \frac{1}{2} CL\emptyset SE \quad (10)$$

$$R_c \geq R_p + CL\emptyset SE \quad (11)$$

$$CL\emptyset SE = \log_e(N) + 5 \quad (12)$$

where AC = Active column vector

C_c = Current number of active columns

R_c = Current row number

R_p = Row number of last active column termination.

If a spill condition exists, the spill group is terminated with the previous row.

The estimated time for the decomposition is calculated from the following expression:

$$T = \frac{1}{2} M_t \sum_{i=1}^N C_i^2 + I(1+n) \sum_{i=1}^N C_i R_i + \frac{1}{2} I \sum C_s^2 \quad (13)$$

$$+ \frac{1}{2} (P_p + P_g) \sum C_t^2 + P_p \sum_{i=1}^N C_i$$

where

M_t = time for tight multiply-add loop

I = time to read and write one term on spill file

P_p = time to put one term in write buffer

P_g = time to get one term from read buffer

SUBROUTINE DESCRIPTIONS

- N = order of matrix
 C_i = number of active columns in the i^{th} row
 S = number of core-held rows in the current spill group
 R_i = number of I/O transfers on the i^{th} row. R_i may be approximated by the integral part of $\frac{C_i}{S}$.
 C_s = number of active columns at beginning of spill operations that are out of range of first spill group (column numbers greater than last row in spill group) for each time that spill operations begin.
 C_t = sum of number of passive columns on secondary storage and number of active columns in working space for each time that active column termination occurs.

and

- n = number of words per term.

For all pivotal rows without spill, the second term of Equation 13 is zero. The estimated time is printed out as a user information message, along with the maximum number of active columns, the average number of active columns, the maximum number of passive columns, the average number of passive columns, the total number of spill groups, and the average number of rows in a spill group.

The decomposition of the matrix proceeds as follows:

1. The active column vector is read from the scratch file into array AC. The use of working storage without spill is shown in Figure 5. The active column vector and the pivotal row are located at the top of working storage. The triangular array for the decomposition products is located at the bottom of working storage.
2. If the next pivotal row is on the spill file, it is read into array P.
3. If the next pivotal row is not on the spill file, it is created as follows:

$$P(i) = W_a(l) + A(j) + B(j) + F(j) \quad (14)$$

where i = all integers from 1 to C

C = number of entries in AC

j = column number in the i -position of AC

$A(j)$ = zero if term is zero in original matrix [A]

MATRIX SUBROUTINE DESCRIPTIONS

$W_a(i) =$ zero if column number in i -position of AC is negative,
or if first entry in AC is negative

$W_a =$ working array associated with previous pivotal row

$F(j) =$ zero if first record on spill file is empty

$B(j) =$ zero unless next row on passive element storage file
equals pivotal row number.

4. The origin of W_a is located at a distance from the end of the working space, W , equal to

$$S_c(C-1) - \frac{1}{2} [S_c(S_c - 1)] \quad (15)$$

where $S_c =$ number of rows currently in working space W

$C =$ current number of active columns

when the pivotal row is within the current spill group. The compact storage for a spill group is indicated in Figure 6. A possible reduction in the number of rows that can be held in W is indicated when the third from the last row of the spill group is the pivotal row.

In the case of no spill, $S_c = C$.

If the pivotal row is on the spill file (pivotal row precedes current spill group),

$$S_c = C - C_p \quad (16)$$

where $C_p =$ number of previously processed columns, providing

$$C - C_p < C_x \quad (17)$$

where C_x satisfies the equation

$$2C + \frac{(C_x)(C_x-1)}{2} = W \quad (18)$$

Otherwise, S_c extends over the full range of the current spill group.

5. If active column termination occurs in the previous row, the contents of W are merged with the contents of the passive column file. The file is rewound and opened to READ.

SUBROUTINE DESCRIPTIONS

6. Processing for the next column proceeds as follows:

$$P(i) = - \frac{P(i)}{P} \quad (19)$$

where P = diagonal term in pivotal row

$$i = C_p + 1, C_p + S_c \quad (C_p + S_c \leq C)$$

C_p = i-position in P of last column processed

S_c = current number of rows in W

If $i_{\max} < C$, the column numbers in the range of i are less than or equal to the number of the last row in the current spill group.

$$W_b(j) = W_a(l) + P(i) * P(k) \quad (20)$$

where $k = i, C$

$$j = 1, W_b$$

$$l = C_a, W_a \quad (C_a = 1 \text{ if pivotal row is on the spill file.})$$

C_a = number of active columns in previous pivotal row

W_b = working array associated with current pivotal row

$W_a(l)$ equals zero if the column number in either the i -position or the k -position of AC is negative. The indexes j and k are incremented together; the index l is incremented only when $W_a(l)$ is not equal to zero. The operations are performed in groups. If i -position is negative, k is indexed from i to C , and j is indexed for the next $(C - i + 1)$ locations. If i -position is positive, k is indexed in groups of positive entries in AC . The indexing of k is interrupted only when new active columns appear.

7. Consider the inner loop of Step 6 for each pivotal row. If the loop terminates with $i = C$, the pivotal row is complete and is written out as the next column of L with the associated diagonal term.
8. If the inner loop terminates with $i < C$ and the current row is the first row of a new spill group (not read from spill file), the contents of W that are out of range of the current spill group (active column numbers greater than last row number in current spill group) are merged with the contents (if any) of the first record on

MATRIX SUBROUTINE DESCRIPTIONS

spill file. Remaining terms are moved to end of W. This situation is shown in Figure 10, with the second row pivotal, and 4 rows in the spill group. The non-zero terms in rows 8, 9, and 13 are beyond the current spill group. The non-zero term in row 9, as shown in Figure 1, is written on the first record of the spill file.

9. If the inner loop terminates with $i < C$, the pivotal row is not complete, and it is written on the spill file.
10. After completing the last row of the spill group, the spill file is rewound and opened to READ. All rows in first record that are beyond the range of the next spill group are copied as the first record on the second spill file. The READ file is rewound when next pivotal row is not on spill file in order to have access to the terms on the first record of the spill file.
11. If the next-to-last pivotal row is written on L, Step 12 is executed next; otherwise, Step 1 is next.
12. The last diagonal term is written on L.
13. On option, the diagonal elements from the original matrix a_{ii} and the upper triangular factor u_{ii} are read and the following operations are performed:
 - a. The number of terms for which $u_{ii} < 0$ is determined.
 - b. $\epsilon_s = \frac{a_{ii}}{u_{ii}}$ is calculated and sorted by absolute magnitude.
 - c. The maximum value (ϵ_{sm}) of ϵ_s is determined.
 - d. The number of ϵ_s values (N_i) in the following ranges are determined:

$$N_1 > 10^{-1} \epsilon_{sm}$$

$$10^{-1} \epsilon_{sm} > N_2 > 10^{-2} \epsilon_{sm}$$

$$10^{-2} \epsilon_{sm} > N_3 > 10^{-3} \epsilon_{sm}$$

$$10^{-3} \epsilon_{sm} > N_4 > 10^{-4} \epsilon_{sm}$$

SUBROUTINE DESCRIPTIONS

$$10^{-4} \epsilon_{sm} > N_5 > 10^{-5} \epsilon_{sm}$$

$$10^{-5} \epsilon_{sm} > N_6 > 0$$

e. A user information message with the following items is printed:

- (1) Number $u_{ij} < 0$
- (2) ϵ_{sm}
- (3) N_1 through N_6
- (4) Row numbers for five largest ϵ_s

MATRIX SUBROUTINE DESCRIPTIONS

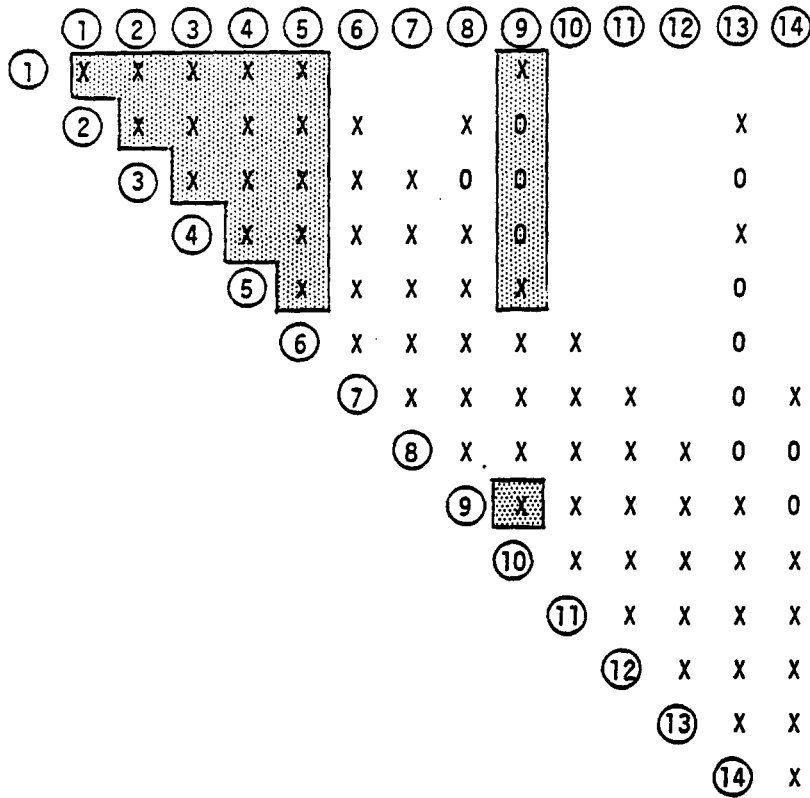


Figure 1. Decomposition with First Row as Pivotal Row

SUBROUTINE DESCRIPTIONS

[illegible]

Figure 2. Compact Storage with First Row as Pivotal Row

MATRIX SUBROUTINE DESCRIPTIONS

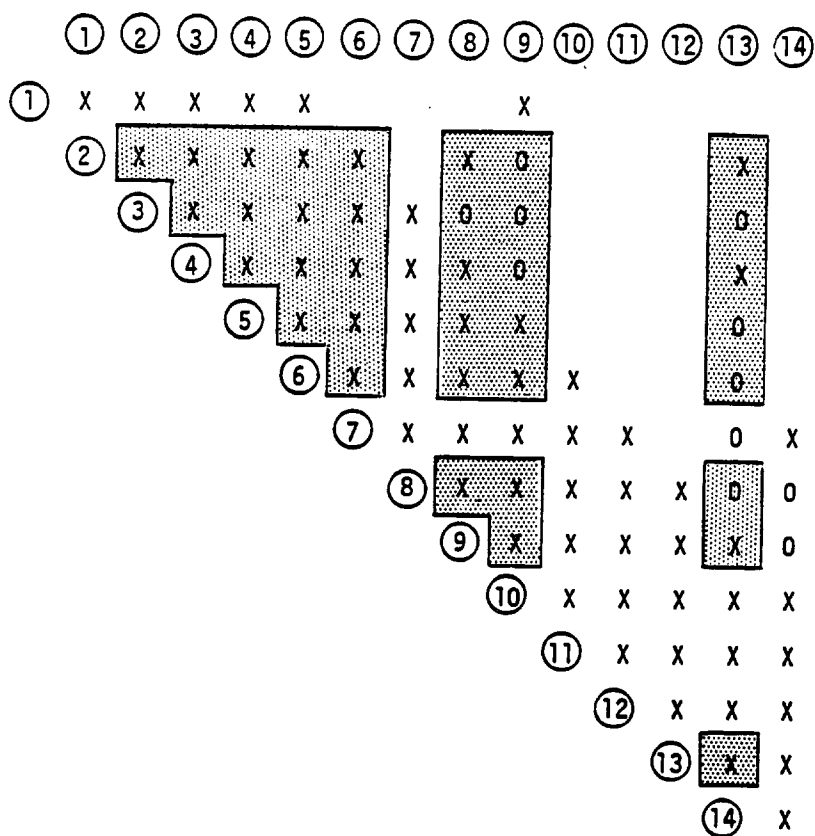


Figure 3. Decomposition with Second Row as Pivotal Row

SUBROUTINE DESCRIPTIONS

1	2	3	4	5	9														
	2	3	4	5	6	8	9	13											
		3	4	5	6	7	8	9	13										
			4	5	6	7	8	9	13										
				5	6	7	8	9	13										
					6	7	8	9	10	13									
						X	X	X	X	X									
							X	X	X	X									
								X	X	X									
									X	X									
										X	X								
												X	X						
														X					

Figure 4. Compact Storage with Second Row as Pivotal Row

MATRIX SUBROUTINE DESCRIPTIONS

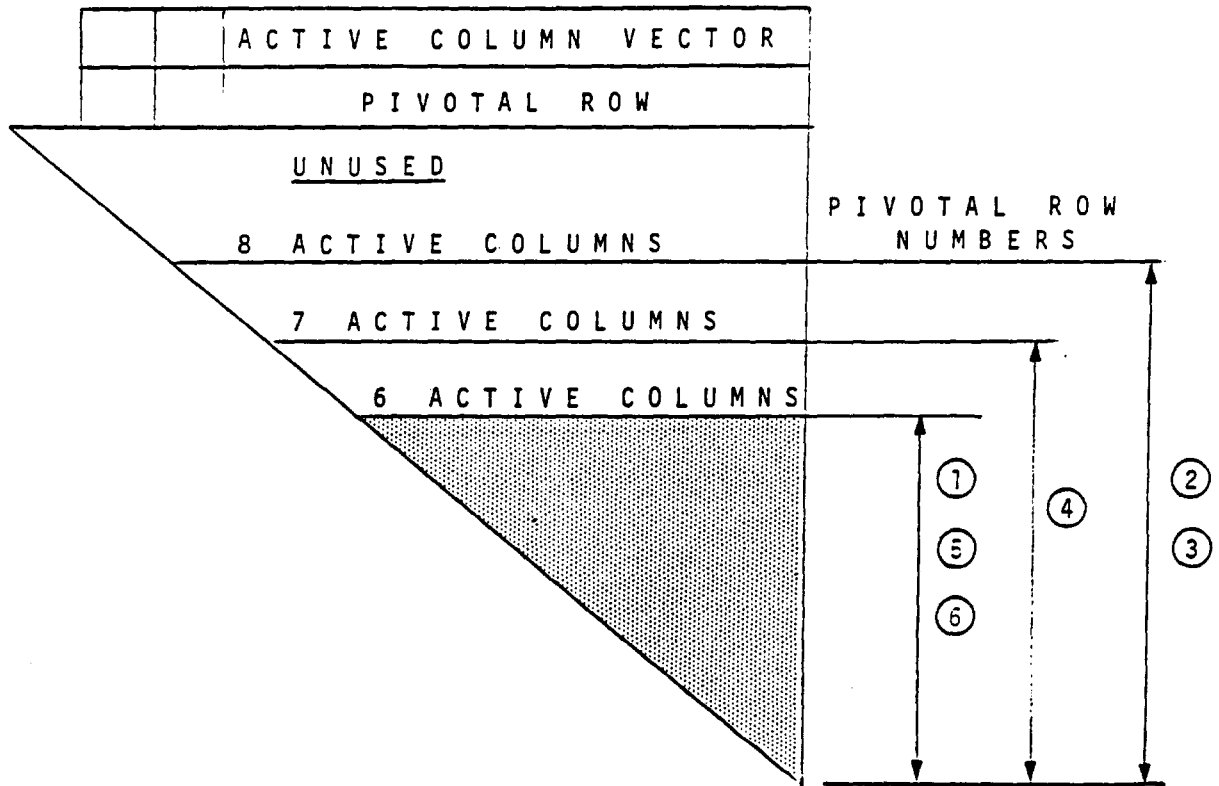


Figure 5. Compact Working Storage

SUBROUTINE DESCRIPTIONS

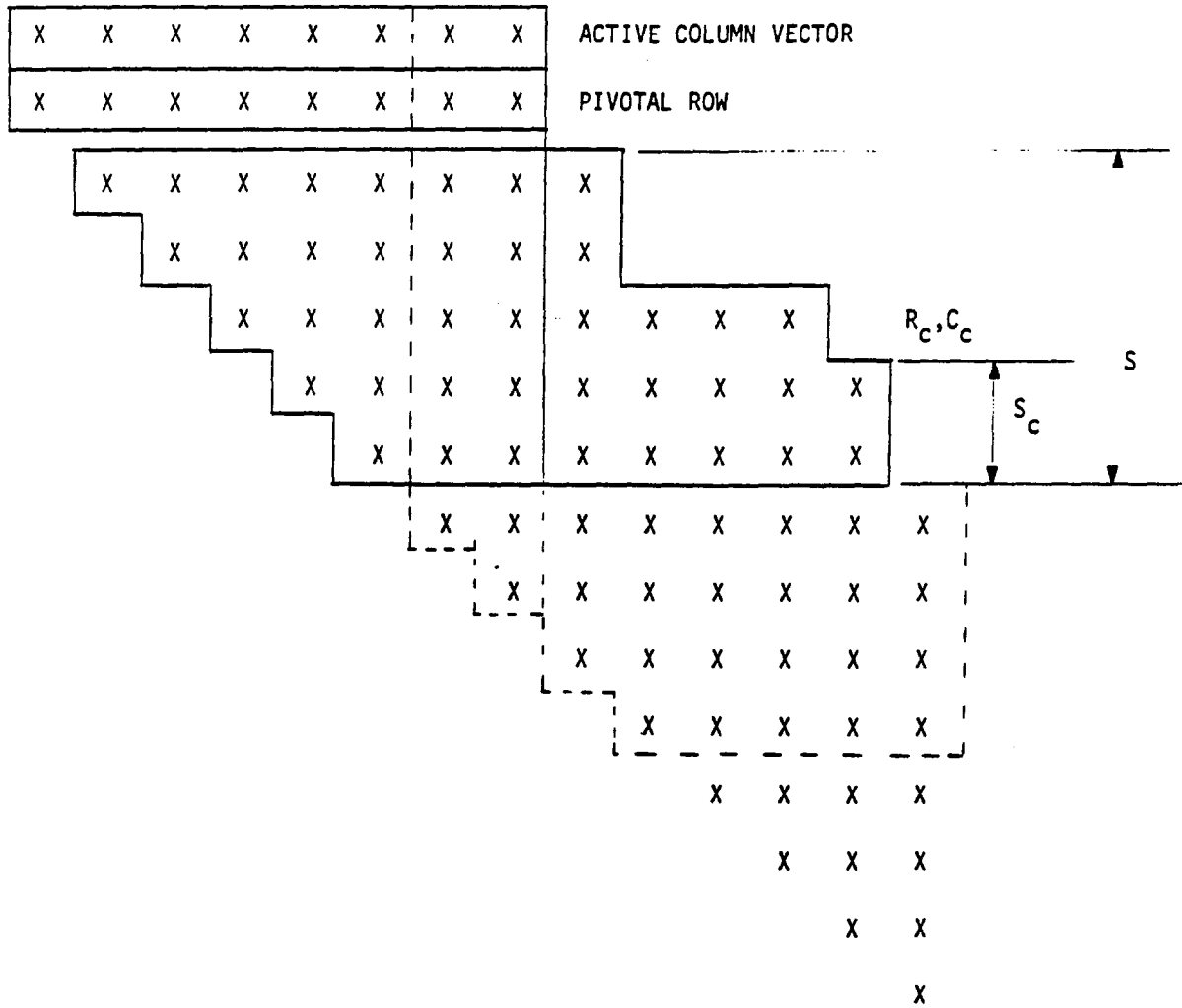


Figure 6. Compact Storage with Spill

MATRIX SUBROUTINE DESCRIPTIONS

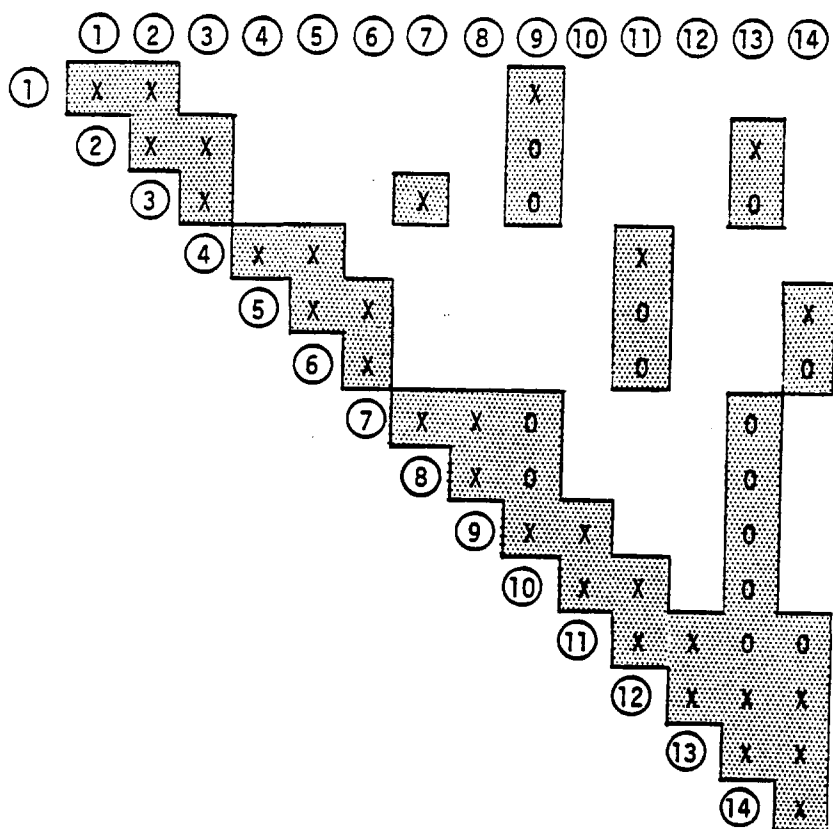


Figure 7. Decomposition with Termination of Active Columns

SUBROUTINE DESCRIPTIONS

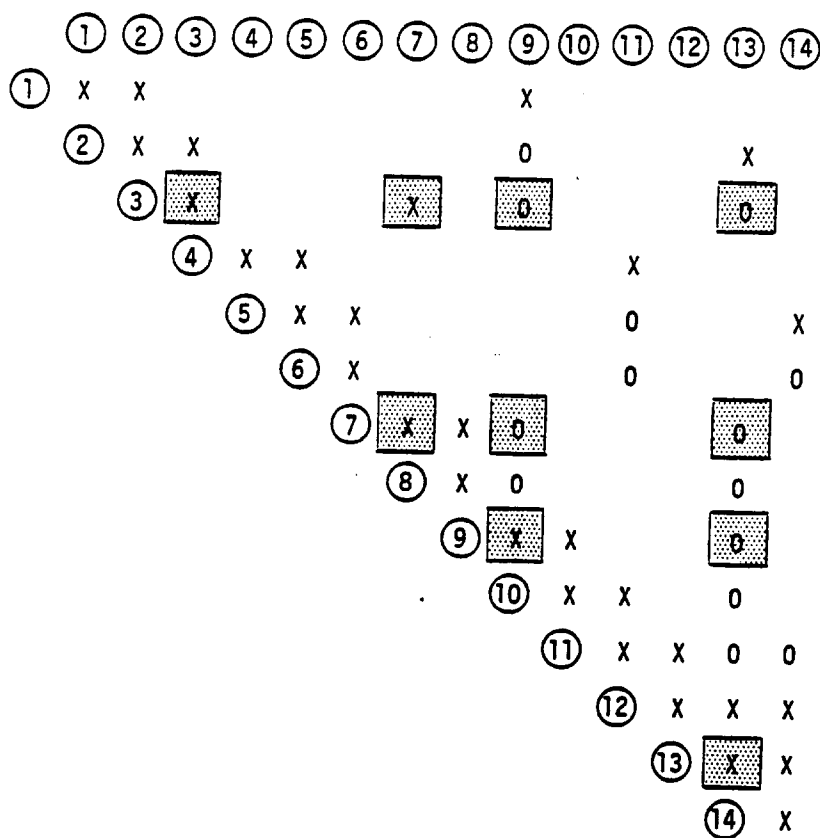


Figure 8. Decomposition with Third Row as Pivotal Row

MATRIX SUBROUTINE DESCRIPTIONS

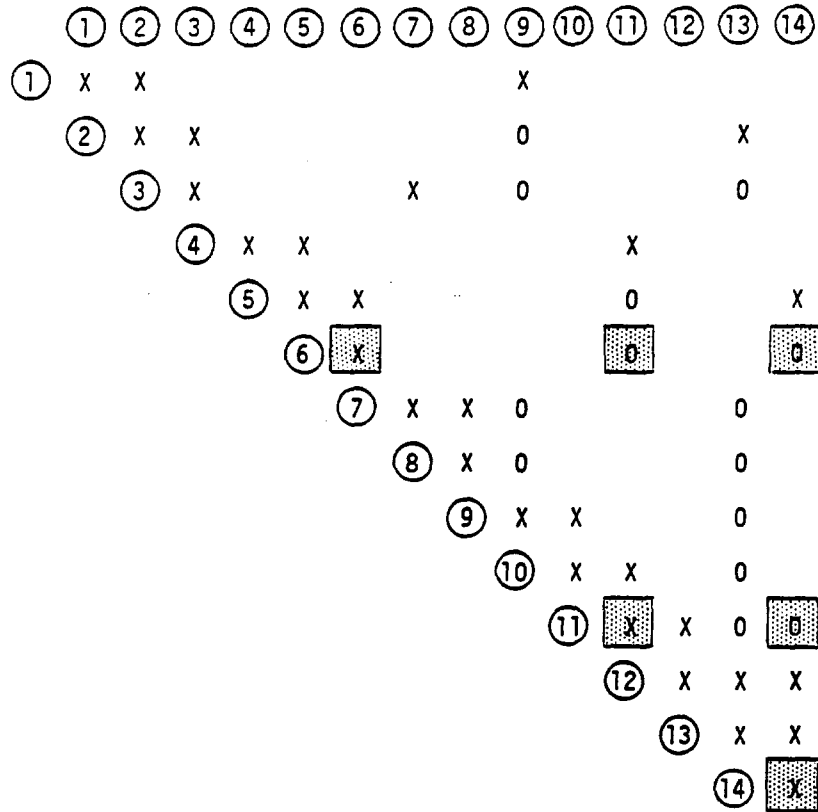


Figure 9. Decomposition with Sixth Row as Pivotal Row

SUBROUTINE DESCRIPTIONS

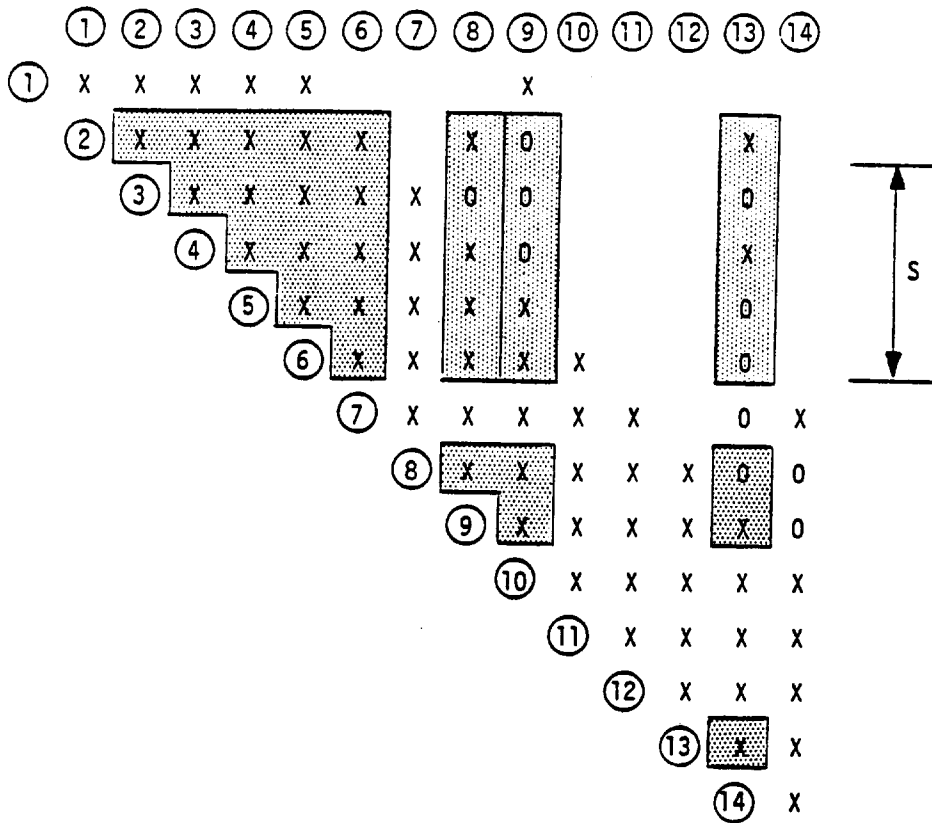


Figure 10. Spill with Second Row as Pivotal Row

MATRIX SUBROUTINE DESCRIPTIONS

3.5.14.5 Auxiliary Subroutine SDCØUT

1. Name: SDCØUT
2. Purpose: To write a row of a matrix in string format using PUTSTR/ENDPUT
3. Calling Sequence:
CALL SDCØUT (BLØCK,RØW,AC,N,VEC,VEC)

where

BLØCK - 15-word array in which the first three words have been completed with GINØ file name, type and format.

RØW - $\begin{cases} 0: \text{row number} = AC(1) \\ \neq 0: \text{row number} = RØW \end{cases}$

AC - a vector of N column positions (column numbers may be < 0)

N - the number of words in AC and the number of terms in VEC

VEC - a vector of N terms. The position of each term is defined by the integer stored in the corresponding position of AC.

3.5.14.6 Auxiliary Subroutine SDCIN

1. Name: SDCIN
2. Purpose: To read a row of a matrix using GETSTR/ENDGET and add the terms of the row into a vector
3. Calling Sequence:
CALL SDCIN (BLØCK,AC,N,VEC,VEC)

where the arguments are defined as in SDCØUT.

Note: The initial call to GETSTR for the row is assumed to have been made prior to CALL SDCIN.

3.5.14.7 Auxiliary Subroutine SDCØM

1. Name: SDCØM
2. Entry Points: SDCØM1, SDCØM2, SDCØM3, SDCØM4
3. Purpose: To compute the contributions of the current pivot row to subsequent rows of [L].

SUBROUTINE DESCRIPTIONS

4. Calling Sequence:

CALL SDCØM_i (P,AC,WA,WB)

where

i - 1,2,3 or 4 for the RSP, RDP, CSP or CDP version respectively.

P - current pivot row

AC - active column vector for P

WA - First address of previous contributions to be accumulated

WB - First address where current contributions are to be stored.

Note: Implementation of the various entry points is computer dependent. Independent FØRTRAN versions are available for each entry point. Assembly language versions are provided for the most commonly used entry points by computer type (e.g., SDCØM1 for CDC, SDCØM2 for IBM).

3.5.14.8 Design Requirements

Core storage requirements include 5 GINØ buffers and a minimum of 3*CMAX terms where CMAX = maximum number of active columns in a pivot row.

3.5.14.9 Information Messages

1. CØNMSG is called following the preface of SDCØMP (prior to the actual decomposition) and after the actual decomposition is completed (prior to the optional statistics computation).
2. Message 3023 prints the following parameters regarding the decomposition:
 - data block name
 - N - number of rows in the data block
 - TIME ESTIMATE - the estimate in seconds to perform the actual decomposition
 - C AVG - the average number of active columns per pivot row
 - PC AVG - the average number of passive columns at each active column termination point
 - SPILL GRØUPS - the number of spill groups
 - S AVG - the average number of rows in each spill group
 - ADDITIONAL CØRE - if positive, the number of additional computer words required to avoid spill. If negative, the number of unused computer words in this decomposition.

MATRIX SUBROUTINE DESCRIPTIONS

C MAX - the maximum number of active columns in any one pivot row

PC MAX - the maximum number of passive columns at any one active column termination point

PC GRØUPS - the number of active column termination points

PREFACE LØØPS - the number of times which the preface to the actual decomposition is executed.

3. If the 57th word of /SYSTEM/ is non-zero, statistics regarding the decomposition are prepared and message 2314 is printed. Section 3.5.14.4 contains a definition of these statistics.

3.5.14.10 Diagnostic Messages

1. Message 3007 is given if the matrix is not square.
2. Message 3008 is given if the preface cannot be executed.
3. The message

SDCØMP FATAL MESSAGE nnnn -- LØGIC ERRØR

may occur followed by the contents of /SDCØMX/. This message indicates a program design error or unexpected condition. nnnn refers to the nearest FØRTRAN statement label at which the error is detected.

4. Message 3097 is printed in the case of a singular matrix prior to a non-standard return.

SUBROUTINE DESCRIPTIONS

3.5.15 DECOMP (Unsymmetric Matrix Decomposition)

3.5.15.1 Entry Point: DECOMP

3.5.15.2 Purpose

To decompose a real square matrix [A] into the form

$$[A] = [L][U] \quad (1)$$

(where [L] is a unit lower triangular matrix, and [U] is an upper triangular matrix), using partial pivoting within the lower band.

3.5.15.3 Calling Sequence

CALL DECOMP (\$n,X,X,X)

COMMON /DCOMPX/ A(7),L(7),U(7),SCR(3),DET,POWER,NX,MINDIA,B,BBAR,C,CBAR,R

- n - Statement number to which control is transferred if [A] is singular.
 - X - An area of core available to DECOMP.
 - A - Matrix control block for the input matrix [A] (if A(1) < 0, avoid re-writing [U] in reverse order).
 - L,U - Matrix control blocks for the output matrices [L] and [U].
 - SCR(3) - GINØ file names for three scratch files - integer.
 - DET - Double precision cell where the scaled value of the determinant of [A] will be stored.
 - POWER - Scale factor to be applied to DET ($\det([A]) = \text{DET} * 10^{**\text{POWER}}$).
 - NX - Number of computer words available at X.
 - MINDIA - Double precision word where the value of the minimum diagonal of [U] is stored.
- B,BAR, } Integer values describing the upper and lower semi-bandwidths, number of active
 C,CBAR, } rows and columns and number of columns of [L] held in core, used to decompose
 R } [A]. (If B, BBAR = 0, GENVEC is called to compute the parameters before de-
 composing [A]. If B, BBAR ≠ 0, the given parameters are used for decomposition).

MATRIX SUBROUTINE DESCRIPTIONS

3.5.15.4 Method

1. Mathematical Considerations: By expanding Equation 1, introducing element notation, and forming the multiplication, we can solve for the elements of [L] and [U]. These equations are given by:

$$l_{ij} = [a_{ij} - \sum_{k=1}^{j-1} l_{ik} u_{kj}] / u_{jj}, \quad i > j \quad (2)$$

$$u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj}, \quad i \leq j. \quad (3)$$

2. General Comments: The implementation of the above equations is accomplished with several constraints in mind. The decomposition procedure is optimized such that the minimum number of operations is performed, with the minimum amount of core used. To accomplish this, the elements of the input matrix [A] are separated into two groups: terms inside the band (band elements), and terms outside the band (active elements). Also, pivoting is used only within the lower band to avoid unnecessarily filling the matrix with non-zero terms.

Since, in practice, structural matrices tend to be semi-banded with scattered terms existing outside the band, this division of the matrix should optimize the decomposition process. Several parameters are generated to describe this division. B is defined as the upper bandwidth, \bar{B} as the lower bandwidth, C is defined as the number of active columns, and \bar{C} as the number of active rows, where an active column is defined as a column containing one or more active elements above the diagonal, and an active row contains one or more active elements below the diagonal. Corresponding to these parameters, several storage areas are defined to hold the various parts of the matrix. The description and location of these areas are given in Figures 1, 2 and 3. A flow chart for DECOMP is given in Figure 4.

The storage areas in Figures 1, 2, and 3 are defined as follows:

- I Storage for the completed columns of [L] still required for computation.
- II Storage for the current column being computed.
- III Storage for active column elements.
- IV Storage for active row elements.
- V Storage for elements created by interactions between active row and column elements.
- VI Storage for indexes identifying active columns.
- VII Storage for indexes identifying active rows.

SUBROUTINE DESCRIPTIONS

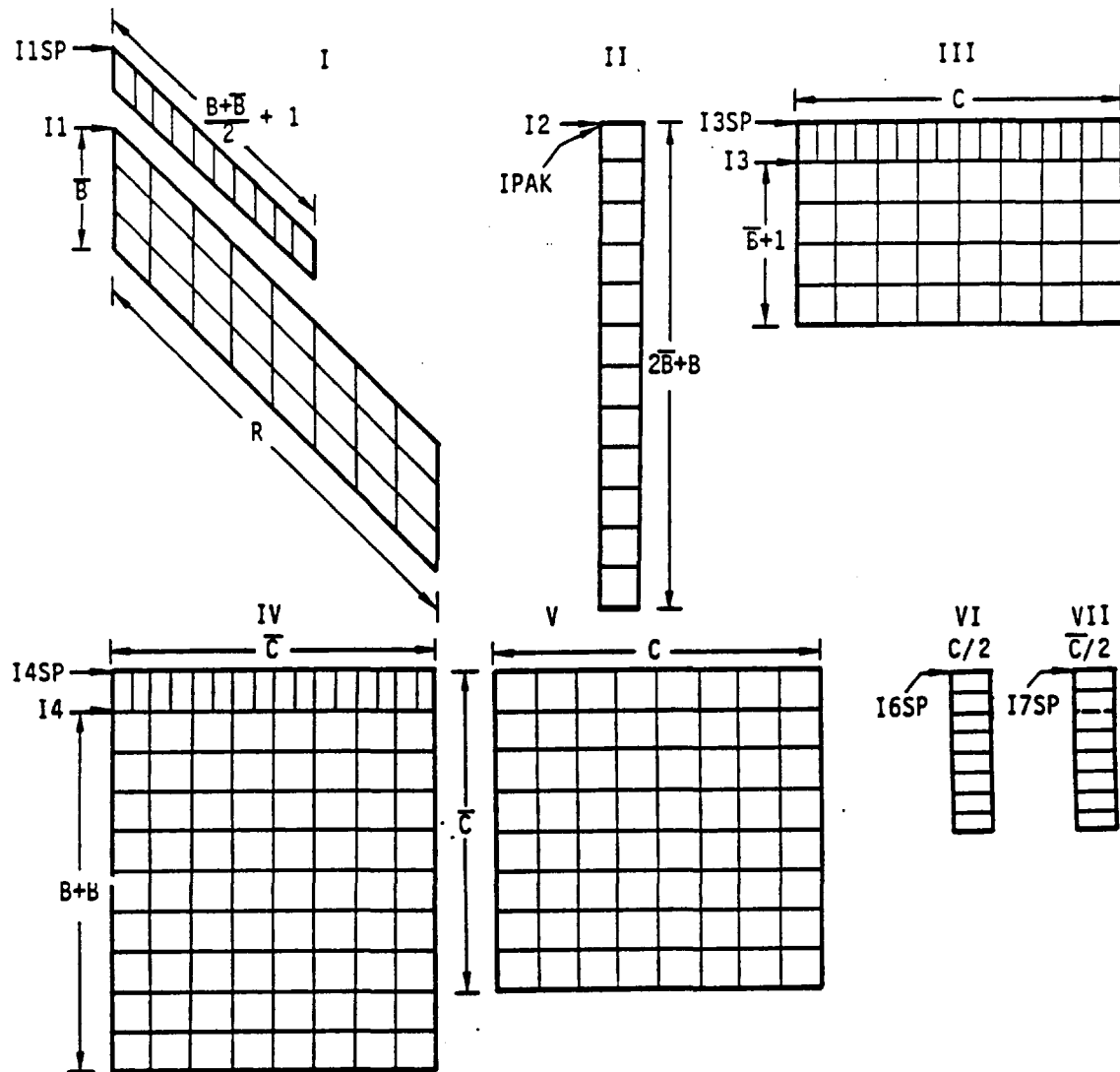


Figure 1. Definition of storage areas for DEC0:IP.

[illegible]

3.5-59 (7/4/76)

SUBROUTINE DESCRIPTIONS

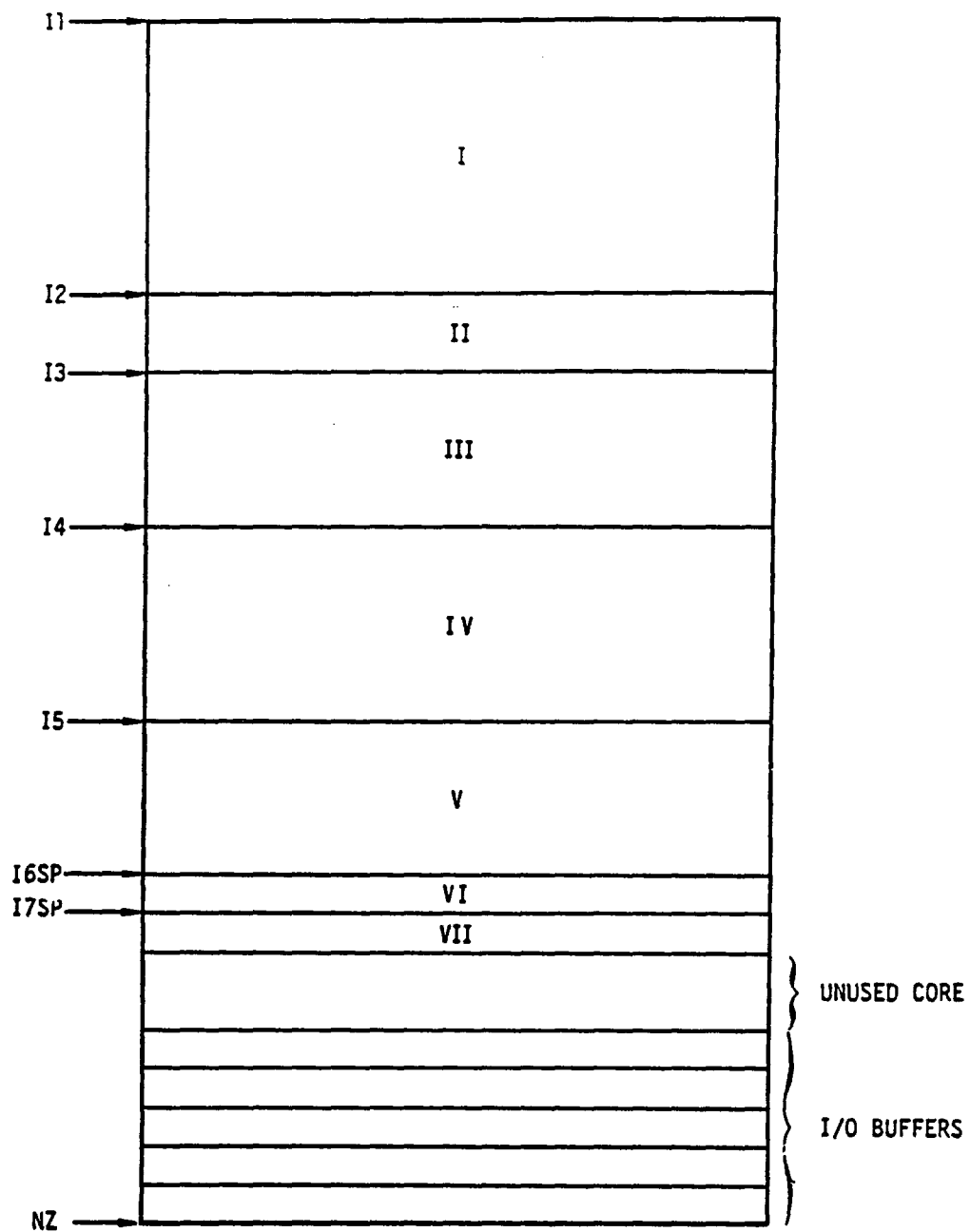


Figure 3. Allocation of core for DECØMP

MATRIX SUBROUTINE DESCRIPTIONS

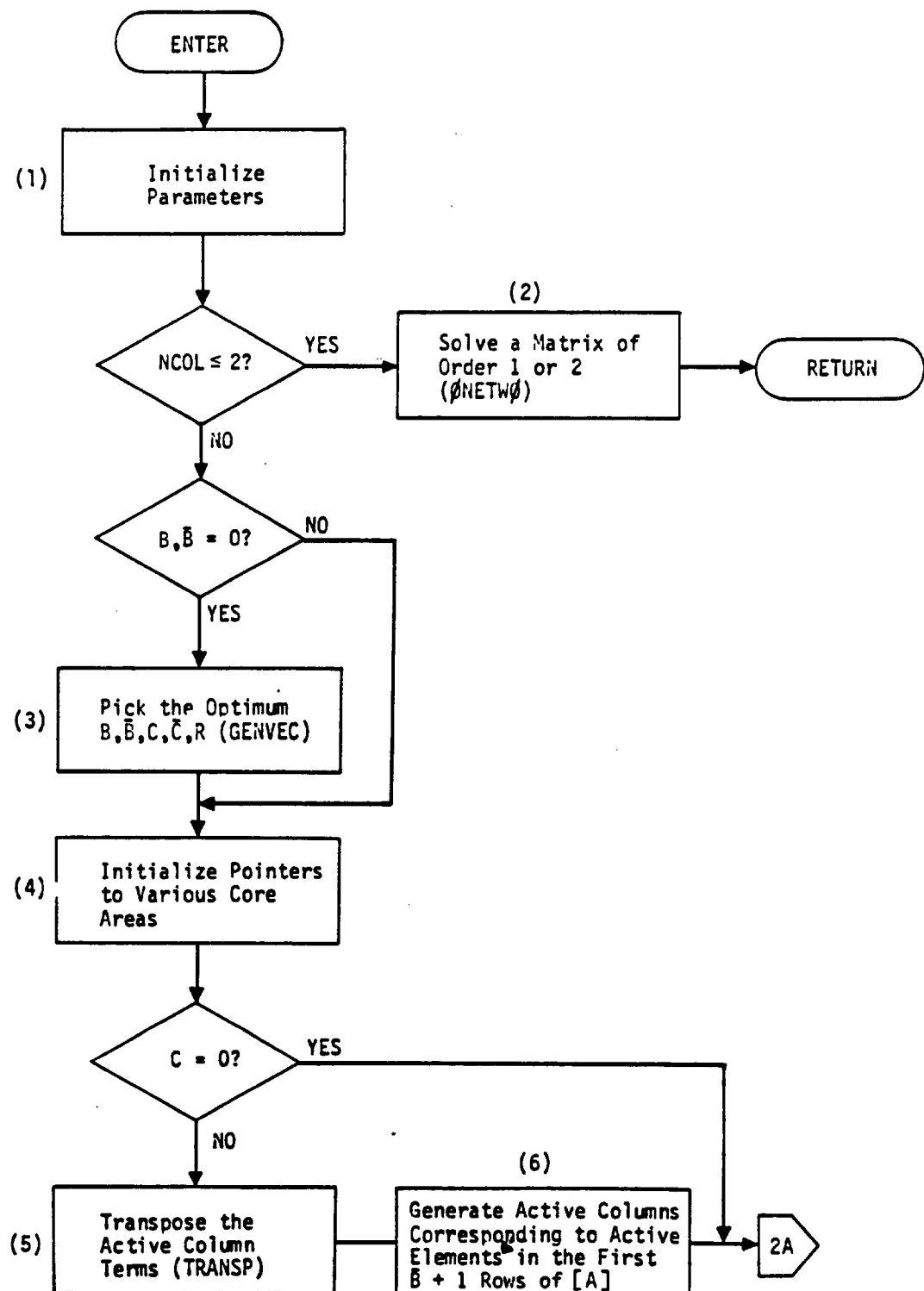


Figure 4.(a) DECOMP program flow

SUBROUTINE DESCRIPTIONS

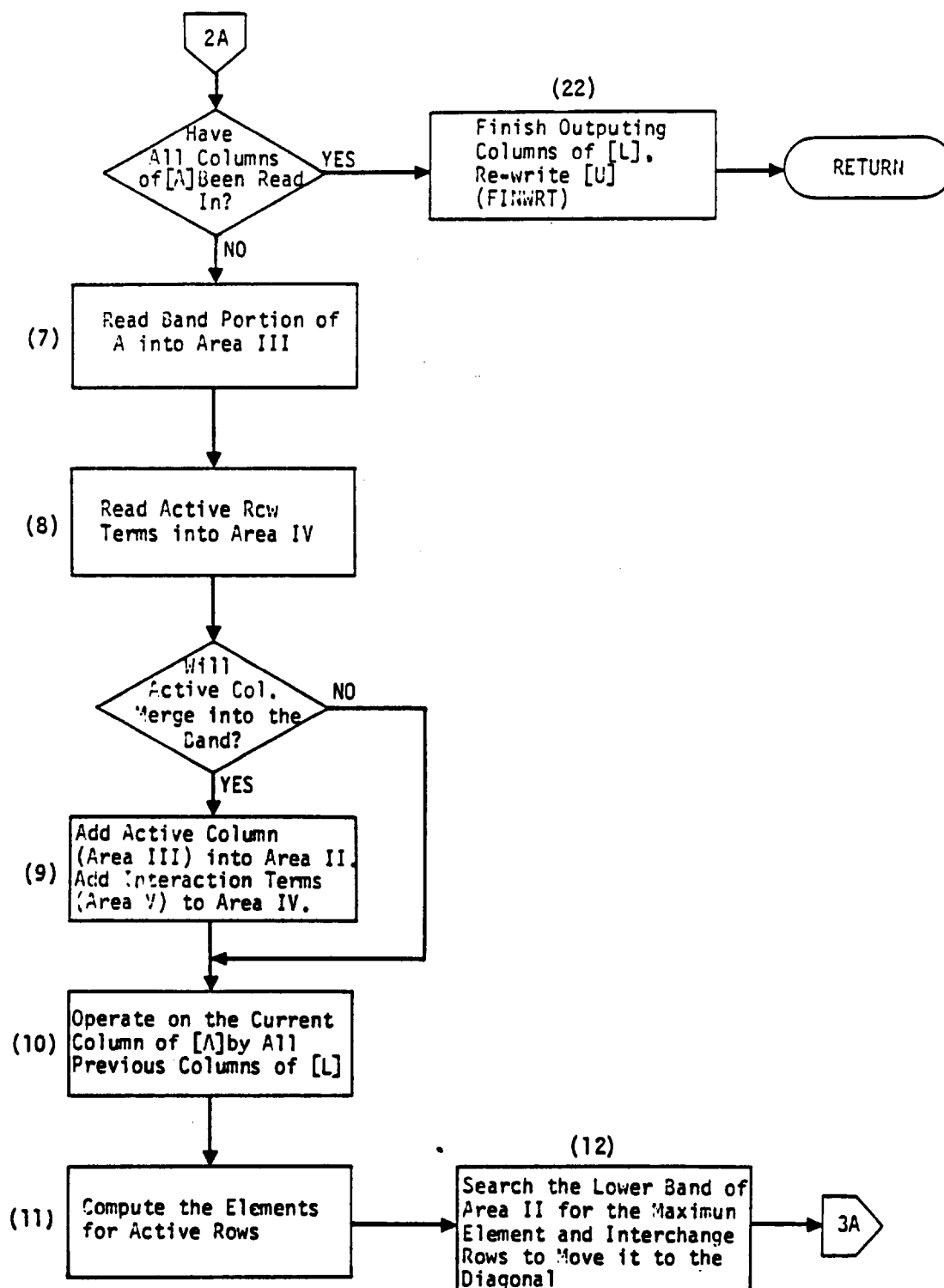


Figure4.(b) DEC0MP program flow

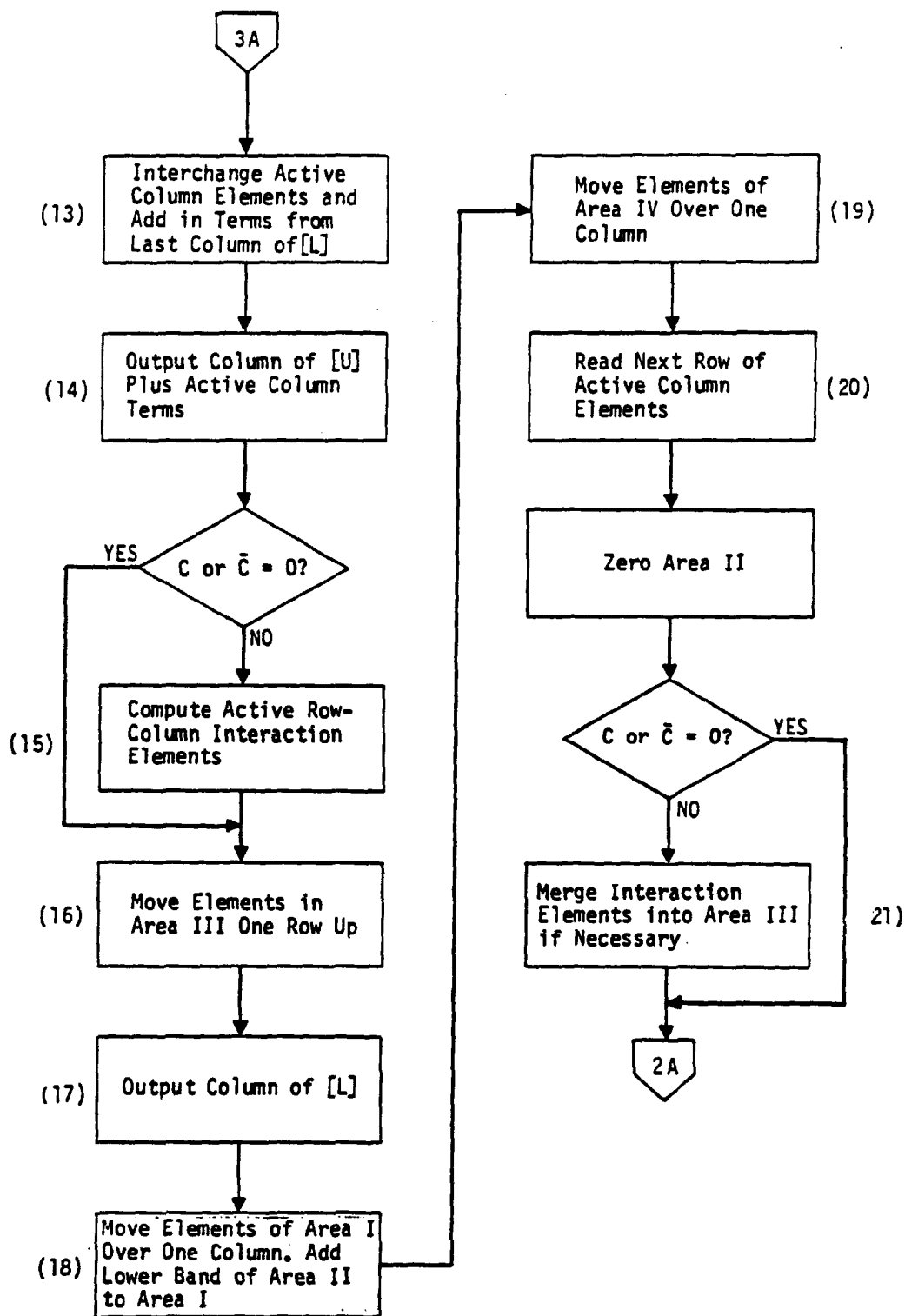


Figure 4.(c) DECØMP program flow

SUBROUTINE DESCRIPTIONS

3. Program Flow: The flow chart in Figure 4 gives the logical program flow of DECØMP. The following comments expand on certain portions of the flow chart:

- (1) Allocate buffers, initialize the determinant, and write header records.
- (2) If the order of [A] is 1 or 2, subroutine ØNETWØ is called to handle the decomposition.
- (3) If B and \bar{B} are input as zero, GENVEC is called to pick the optimum parameters for decomposition.
- (4) The pointers into the various areas of core shown in Figure 1 are computed.
- (5) If there exist active elements in the upper triangle, TRANSP is called to transpose these elements.
- (6) Active columns are initialized for all columns having an active element within the first $\bar{B} + 1$ rows of [A].
- (7) The band portion of the next column of [A] is read into Area II.
- (8) Any active elements occurring below the diagonal in the current column are added into existing active rows, or new active rows are created.
- (9) When an active column merges into the band, a column from Area III is added into corresponding positions in Area II, and a column of interaction elements in Area V is added to the active row terms in Area IV.
- (10) The current column of [A] in Area II is operated on by the columns of [L] stored in Area I.
- (11) Terms corresponding to active rows not yet merged into the band are added into Area II.
- (12) The lower band portion of Area II is searched for the maximum element. Rows are interchanged to bring it to the diagonal position, and the interchange index is stored in Area I.
- (13) Active columns elements are interchanged corresponding to the interchange within the band. If a column of [L] is about to be output (i.e., $B + \bar{B}$ columns of [L] are stored in Area I), terms arising from that column are added into the active columns.

MATRIX SUBROUTINE DESCRIPTIONS

- (14) The column of [U] from Area II plus a row of active column terms in Area III are output.
- (15) If active rows and columns exist, terms arising from their interaction are computed and added into Area V.
- (16) Elements in the active columns are moved up one row position in Area III to replace the output elements, and make room for a new row.
- (17) The first column of [L] in Area I is written out if Area I is full. The active row elements belonging to that column are also output.
- (18) The columns of [L] in Area I are moved over one column and the lower band portion of Area II is stored in Area I.
- (19) Active rows in Area IV are moved over one column.
- (20) The next row of active elements in the upper triangle of [A] is read. Elements are added to existing active columns, or new columns are created.
- (21) When possible, a row of interaction elements in Area V is merged into the bottom row of active column elements (Area III).
- (22) After processing all columns of [A], FINWRT is called to complete the outputting of [L] and to re-write [U].

3.5.15.5 Auxiliary Subroutine TRANSP

- 1. Entry Point: TRANSP
- 2. Purpose: To do an in-core transpose of the active elements occurring outside the band and in the upper half of the matrix (i.e., transpose all elements a_{ij} such that $j - i \geq B$).
- 3. Calling Sequence: CALL TRANSP (X,X,NX,A,B,SCRFIL)
 - X - An area of core available to TRANSP.
 - NX - Number of words available at X.
 - A - GINØ file containing the input matrix [A] - integer.
 - B - Upper bandwidth of matrix [A].
 - SCRFIL - GINØ file where transposed elements are stored.

SUBROUTINE DESCRIPTIONS

4. Method: The input matrix [A] is read, and all elements occurring outside the band in the upper triangle are stored in core, along with their row and column position. This list is then searched and elements output in transposed order.

3.5.15.6 Auxiliary Routine ØNETWØ

1. Entry Points: ØNETWØ, FINWRT
2. Purpose: ØNETWØ is a separate routine whose sole responsibility is to solve matrices of order one or two.

FINWRT is a section of code separated from DECØMP due to compiler overflow. Its function is to finish outputting the remaining columns of [L] and to re-write the columns of [U].

3.5.15.7 Auxiliary Subroutine GENVEC

1. Entry Point: GENVEC
2. Purpose: To pick the optimum B, \bar{B} , C, \bar{C} , and R for a given matrix [A].
3. Calling Sequence: CALL GENVEC (\$n₁, BUF, A, NX, X, N, B, BBAR, C, CBAR, R, IENTRY)
 - n₁ - Statement to which control is transferred if a null column is discovered in [A].
 - BUF - Location of a GINØ buffer.
 - A - GINØ file containing the input matrix [A].
 - X - An area of core available to GENVEC.
 - NX - Number of words of core available at X.
 - N - The order of the matrix [A].
 - B, BBAR, C, CBAR, R - $\left\{ \begin{array}{l} \text{Integer output parameters giving the optimum values for the upper and lower} \\ \text{bandwidths, the number of active rows and columns, and the number of columns} \\ \text{of [L] held in core.} \end{array} \right.$
 - IENTRY - $\left\{ \begin{array}{l} = 1 \text{ implies DECØMP called GENVEC.} \\ = 2 \text{ implies CDCØMP called GENVEC.} \end{array} \right.$

4. Method: The following logic flow gives the means by which the optimum bandwidths are chosen.

MATRIX SUBROUTINE DESCRIPTIONS

A. Locate extreme non-zero terms

1. Initialize active column list to zero. The length of the list is equal to the maximum value of the upper bandwidth (B) that is of interest.
($B_{\max} = \frac{10^5}{NM_B}$, or the order of the problem, whichever is less, where N is the order of the matrix and M_B is the arithmetic time in μ seconds for performing one multiply and one add.
2. Initialize the column list to zero. The length of the list is the order of the problem.
3. Locate the non-zero elements in the next column of the matrix.
4. Insert the column number of all non-zero elements in the correct row position of the column number list with the following constraints:
 - a. Consider only elements in the lower triangle.
 - b. Do not insert column numbers in row positions already occupied.
5. Insert the row number of the non-zero element located in the lowest numbered row into the column position of the row number list for the current column under consideration.
6. Return to step 4 until all columns of the matrix have been processed.

B. Determine active columns

1. Zero counter E

Set Counter F to N-1

Set Counter G to N-1

Set Counter H to 2

2. Beginning with the last entry in the row number list, subtract the current value of counter H, and test for a negative sign. If negative, increment counter E by one.
3. The current value of counter E is the number of active columns when the upper bandwidth is equal to the current value of counter F. Compare counter E with the existing entry in the active column list for the bandwidth indicated by counter F. Update the active column list if counter E is greater than the existing entry.
4. Decrement counter F by one. Return to step 2 unless counter F is zero.

SUBROUTINE DESCRIPTIONS

5. Decrement counter G by one. Increment counter H by one. Set counter F to the value of counter G. Zero counter E. Return to step 2 unless counter G is zero.
6. The final active column list contains the maximum number of active columns for bandwidths of unity through the maximum upper bandwidth investigated.
7. Prepare reduced active column list by extracting pairs - minimum B and associated C for unique values of C.

C. Determine active rows for given B and \bar{B}

1. Extract pairs (row number, L, and column number, K) from the column number list for which

$$L - K > \bar{B}$$

2. Consider a new list of pairs consisting of K and $L + B$. For each pair $(K, L + B)$ determine the number of remaining pairs $(K_i, (L + B)_i)$ for which

$$K_i < L + B \text{ and } (L + B)_i \geq K$$

3. The maximum number of pairs satisfying the relation in step 2 for any single pair is \bar{C} for the given B and \bar{B} .

D. Select B, \bar{B} , C and \bar{C} for Minimum Decomposition Time

1. Select the next value of B and associated C from the reduced active column list (begin with maximum B of interest).
2. Assume $B = \bar{B}$ and $C = \bar{C}$.
3. Calculate R and T with preliminary timing equations.
4. Save the minimum T and the associated B and C.
5. Return to step 1 unless all entries in reduced active column list have been used.
6. If the matrix is unsymmetric use B and C from step 4 and set $\bar{B} = B$.
7. Determine C for a given \bar{B} and B.
8. Calculate R and T with the preliminary timing equations.
9. Compare with the previous minimum T.

MATRIX SUBROUTINE DESCRIPTIONS

10. If the new T exceeds the previous minimum by more than 20% or $\bar{B} = \text{maximum } B$ of interest go to 12.
11. Save the minimum T along with associated B , \bar{B} , C and \bar{C} . Increment \bar{B} by 2% of the B associated with minimum T and go to 7.
12. Return to $B = \bar{B}$.
13. Decrement \bar{B} by 2% of the B associated with minimum T and determine the associated C .
14. Calculate R and T with the preliminary timing equations.
15. Compare with the previous minimum T while decrementing \bar{B} .
16. If the new T exceeds the previous minimum by more than 20% or $\bar{B} = 2$ go to 18.
17. Save the minimum T along with associated B , \bar{B} , C and \bar{C} and go to 13.
18. Save values of B , \bar{B} , C and \bar{C} associated with the minimum value of T from upsearch and downsearch on \bar{B} , for use in decomposition.

3.5.15.9 Auxiliary Function FINDC

1. Entry Point: FINDC
2. Purpose: To find the number of active rows (\bar{C}) for a given B and \bar{B} .
3. Calling Sequence: CALL FINDC (B,BBAR,N,X,Y,CBAR)

B - Upper bandwidth - integer.
BBAR - Lower bandwidth - integer.
N - Order of the problem.
X - Column number list.
Y - Scratch vector.
CBAR - The number of active rows, \bar{C} .
4. Method: See step C in the GENVEC method.

SUBROUTINE DESCRIPTIONS

3.5.15.10 Auxiliary Routine TIMEEQ

1. Entry Points: T,TFIN,RCØRE
2. Purpose: To compute the preliminary and final timing equations for decomposition, and to compute the core allocation function.
3. Calling Sequences: CALL T (B,BBAR,C,CBAR,R,IENTRY,N,TIM)
CALL TFIN (B,BBAR,C,CBAR,R,IENTRY,N,TIMEX)
CALL RCØRE (B,BBAR,C,CBAR,N,IENTRY,NX,R)

B,BBAR
C,CBAR, - Upper and lower bandwidths, number of active rows and columns.

IENTRY - {=1 implies entry was from DECØMP.
 {=2 implies entry was from CDCØMP.

N - The order of the problem.

TIM - Floating point value for the preliminary time.

TIMEX - Floating point value for the final timing equation.

NX - Number of core words available to DECØMP or CDCØMP.

R - The number of columns of [L] that can be held in core (output by RCØRE, input to T and TFIN).

4. Method: The following equations are evaluated to give the desired output.

A. Core Function

$$R = (NX - ((B + \bar{B} + 1) + 2*IENTRY*MINO(N, B + \bar{B} + \bar{B}) + 2*IENTRY*C*(\bar{B} + 2) + 2*\bar{C}*IENTRY*(MINO(B + \bar{B}, N) + 1) + 2*IENTRY*C*\bar{C} + C + \bar{C}*IENTRY + \bar{C}) - 6*SYSEBUF)/(2*IENTRY*\bar{B}) \quad (4)$$

B. Preliminary Timing Equation

$$TIM = \frac{N}{10^6} [M_B \bar{B}R + M_C(\bar{B}C + \bar{B}\bar{C} + B\bar{C} + 2C\bar{C}) + I\bar{B}(B + \bar{B} - R - 1)] \quad (5)$$

where

M_B = Arithmetic time in μ seconds for one term inside the band.

M_C = Arithmetic time in μ seconds for one term in the active row or active column.

MATRIX SUBROUTINE DESCRIPTIONS

$I = I/\emptyset$ time in μ seconds for one term.

C. Final Timing Equations

TIMEX is a function of T_1 , T_2 , T_3 and T_4 as defined below (P = matrix packing time in μ seconds for one term).

T_1 is given by:

$$T_1 = K_1 [M_B \bar{B} R + I \bar{B} (B + \bar{B} - R) + P(B + 2\bar{B})] , \quad (6)$$

where

$$K_1 = \begin{cases} N - B - 2\bar{B} , & \text{if } N - B - 2\bar{B} > 0 \\ 0 & , \text{if } N - B - 2\bar{B} \leq 0 . \end{cases} \quad (7)$$

T_2 is given by:

$$T_2 = \frac{K_2}{2} [\bar{B} K_2 M_B + (K_3 - R)(I - M_B) \bar{B} + 2P\bar{B} + PK_2] , \quad (8)$$

where

$$K_2 = K_3 = B + \bar{B} \quad (9)$$

if $N \geq B + 2\bar{B}$; otherwise,

$$K_2 = N - B , \quad (10)$$

and

$$K_3 = \begin{cases} B + \bar{B} & \text{if } N \geq B + \bar{B} \\ N & \text{if } N < B + \bar{B} . \end{cases} \quad (11)$$

T_3 is given by:

$$T_3 = \frac{\bar{B}^3}{3} M_B + \frac{K_4^3}{2} I + P\bar{B} K_5 . \quad (12)$$

where

$$K_4 = \begin{cases} B + \bar{B} - R , & \text{if } B - R \leq 0 \\ \bar{B} & , \text{if } B - R > 0 . \end{cases} \quad (13)$$

SUBROUTINE DESCRIPTIONS

and

$$K_5 = B + \frac{3}{2} B, \quad (14)$$

if $N \geq B + 2\bar{B}$. Otherwise,

$$K_4 = \begin{cases} N - R, & \text{if } N - R \leq \bar{B} \\ \bar{B}, & \text{if } N - R > \bar{B} \end{cases} \quad (15)$$

and

$$K_5 = N \quad (16)$$

T_4 is given by:

$$T_4 = (N - \bar{B})[M_C(\bar{B}C + B\bar{C} + \bar{B}\bar{C} + C\bar{C}) + P(C + \bar{C})] \quad (17)$$

Finally,

$$\text{TIMEX} = (T_1 + T_2 + T_3 + T_4)10^{-6}, \quad (18)$$

where TIMEX is the total estimated time for decomposition.

3.5.15.11 Auxiliary Subroutine DL00P

1. Entry Points: DL00P,DDL00P,XL00P
2. Purpose: To improve efficiency of F0RTRAN generated code for several loops in DEC0MP.

3.5.15.12 Design Requirements

Core storage requirements depend on the parameters B , \bar{B} , C and \bar{C} . Areas II, III, IV, and V must reside in core at all times, along with a minimum of two columns of Area I, and five GIN0 buffers. GENVEC is designed to pick the combination of B , \bar{B} , C , and \bar{C} such that the problem will allocate if at all possible.

The file containing $[U]$ is not in standard format, as the active column terms are stored out of place. For this reason $[L]$ and $[U]$ should be used as input only to GFBS or an associated routine.

MATRIX SUBROUTINE DESCRIPTIONS

3.5.15.13 Information Messages

1. CØNMSG is called at entry and exit from DECØMP. The line

XXXX DECØMP

will appear twice per decomposition. The actual execution time of DECØMP will be the difference in the times (where XXXX = time in seconds).

2. Message 3028 gives the values of the parameter B , \bar{B} , C , \bar{C} , and R chosen for the decomposition.

3. Message 3027 gives the estimated time in seconds for the decomposition.

3.5.15.14 Diagnostic Messages

Fatal error messages 3008, 3025, and 3097 may occur.

SUBROUTINE DESCRIPTIONS

3.5.16 CDCOMP (Complex Matrix Decomposition)

3.5.16.1 Entry Point: CDCOMP

3.5.16.2 Purpose

To decompose a complex square matrix $[A]$ into the form $[A] = [L][U]$ where $[L]$ is a unit lower triangular matrix and $[U]$ is an upper triangular matrix.

3.5.16.3 Calling Sequence

CALL CDCOMP (n_1, X, X, X)

COMMON /CDCMPX/ A(7),L(7),U(7),SCR(3),DET(2),POWER,NX,MINDIA,B,BBAR,C,CBAR,R

- n_1 - Location in calling program where control is transferred if $[A]$ is singular.
- X - An area of working storage.
- A - Input matrix control block for $[A]$.
- L, U - Output matrix control blocks for $[L]$ and $[U]$.
- SCR(3) - Three scratch files available for use.
- DET(2) - Two double precision words where the real and imaginary values of the determinant are stored.
- POWER - Scale factor to be applied to the determinant ($\det([A]) = \text{DET} * 10^{**\text{POWER}}$).
- NX - Number of computer words at X .
- MINDIA - Double precision word where the modulus of the minimum diagonal element of $[U]$ is stored.
- $B, \text{BBAR}, C, \text{CBAR}, R$ - $\left\{ \begin{array}{l} \text{If } B = \text{BBAR} = 0, \text{ compute and store } B, \text{BBAR}, C, \text{CBAR}, R \text{ before decomposing } [A]. \\ \text{If } B \text{ or } \text{BBAR} \neq 0, \text{ use previously stored values of } B, \text{BBAR}, C, \text{CBAR} \text{ and } R \text{ for decomposing.} \end{array} \right.$

MATRIX SUBROUTINE DESCRIPTIONS

3.5.16.4 Method

CDCOMP is simply a copy of DECOMP with the arithmetic statements replaced by complex arithmetic. Pointers to storage areas were modified to accommodate the extra words needed.

3.5.16.5 Auxiliary Subroutine CTRNSP

Purpose: Complex version of TRANSP (see Section 3.5.15.5).

3.5.16.6 Auxiliary Subroutine COM12

Purpose: Complex version of ONETW0 (see Section 3.5.15.6)

3.5.16.7 Auxiliary Routine CL00P

Inner loop routine.

3.5.16.8 Auxiliary Routine CXL00P

Inner loop routine.

3.5.16.9 Design Requirements

See subroutine descriptions for DECOMP, Section 3.5.15.

3.5.16.10 Diagnostic Messages

See subroutine descriptions for DECOMP, Section 3.5.15.

SUBROUTINE DESCRIPTIONS

3.5.17 FBS (Forward - Backward Substitution)

3.5.17.1 Entry Point: FBS

3.5.17.2 Purpose

Given the decomposition of a real symmetric matrix $[A] = [L][D][L]^T$ FBS will perform the forward-backward pass necessary to solve the system of linear equations $[A][X] = [B]$.

3.5.17.3 Calling Sequence

CALL FBS(Z,Z)

COMMON/FBSX/L(7),U(7),B(7),X(7),NZ,PREC,SIGN

L - Matrix control block for the lower triangular factor output from SDCOMP.

U - Not used.

B,X - Matrix control blocks for the matrices [B] and [X].

NZ - Number of computer words at Z.

PREC - $\begin{cases} 1, & \text{perform arithmetic in single precision} \\ 2, & \text{perform arithmetic in double precision} \end{cases}$

SIGN - $\begin{cases} +1, & \text{solve } [A][X] = [B]. \\ -1, & \text{solve } [A][X] = -[B]. \end{cases}$

Z - An area of working storage.

3.5.17.4 Method

Mathematical Considerations. Given the lower triangular matrix [L] with the diagonal matrix [D] stored over its diagonal, FBS solves the two systems of equations given by

$$[L][Y] = \pm[B]$$

and

$$[L]^T[X] = [D]^{-1}[Y]$$

Elements [Y] and [X] are given by

$$y_{ij} = b_{ij} + \sum_{k=1}^{i-1} l_{ik} y_{kj}$$

$$x_{ij} = y_{ij}/d_i + \sum_{k=i+1}^n l_{ki} x_{kj}$$

MATRIX SUBROUTINE DESCRIPTIONS

3.5.17.5 Auxiliary Subroutine FBSI

1. Name: FBSI
2. Entry Points: FBS1, FBS2, FBS3, FBS4
3. Purpose: To perform the mathematical computations for the forward and backward passes.
4. Calling Sequence:

CALL FBSi (BLØCK,Z,Z(LAST),NWDS)

where

i = 1,2,3 or 4 to specify the RSP, RDP, CSP or CDP version respectively.

BLØCK - 15-word string communication block for [L].

Z,Z(LAST) - Addresses of the first and last load vectors currently in core.

NWDS - Number of computer words in a load vector.

Note: Implementation of the various entry points is computer dependent. Independent FORTRAN versions are available for each entry point. Assembly language versions are provided for the most commonly used entry points (e.g., FBS1 for CDC, FBS2 for IBM).

3.5.17.6 Design Requirements

One column of [B] and one GINØ buffer must fit in core.

3.5.17.7 Information Messages

CØNMSG is called at entry and exit from FBS.

3.5.17.8 Diagnostic Messages

- 3008
- FBSI FATAL MESSAGE 10 is issued when the first element of a column (or row) of [L] is not the diagonal element.

SUBROUTINE DESCRIPTIONS

3.5.18 SSG3A (Driver for FBS).

3.5.18.1 Entry Point: SSG3A.

3.5.18.2 Purpose

To solve $[A] [X] = [B]$ using $[A] = [L] [U]$ where U is an upper triangular matrix obtained from $[L]$. The residual matrix $[RES] = [A] [X] - [B]$ will be computed when requested by the IRES option.

3.5.18.3 Calling Sequence

CALL SSG3A(FILEA,FILEL,FILEB,FILEX,SCR1,SCR2,IRES,FILER)

COMMON / / N,IRESS, NSKIP,IEPSI

FILEA - GINØ file name of $[A]$ - integer - input.

FILEL - GINØ file name of $[L]$ - integer - input.

FILEB - GINØ file name of $[B]$ - integer - input.

FILEX - GINØ file name of $[X]$ - integer - input.

SCR1 - GINØ name of scratch file - integer - input.

SCR2 - GINØ name of scratch file - integer - input.

IRES - Option for residual vector - integer - input. IRES < 0 suppresses calculation of residual vector.

FILER - GINØ file name of residual vector - integer - input.

NSKIP - Record number -1 of CASECC - integer - input.

IEPSI - Residual error - integer - output.

Notes: 1. Files FILEL, FILEB and FILEX may not be purged.

2. Files FILEA, SCR1 and SCR2 may be purged if IRES \leq 0 or FILER is purged.

3.5.18.4 Method

/FBSX/ is set to compute $[A] [X] = [B]$.

If IRES > 0 and FILER is not purged, SSG2B is called to compute $[RES] = [A] [X] - [B]$ (see section 3.5.17.3). Then for each column $\{X_i\}$, ϵ_i is computed and printed, where

$$\epsilon_i = \{X_i\}^T \{RES_i\} / \{B_i\}^T \{X_i\}$$

If ϵ_i is less than .001, message 3058 is printed and IEPSI is set to -1.

3.5.18.5 Design Requirements

Open core at /SSGA3/.

MATRIX SUBROUTINE DESCRIPTIONS

3.5.19 GFBS (General Forward - Backward Substitution).

3.5.19.1 Entry Point: GFBS.

3.5.19.2 Purpose

Given the decomposition of a general square matrix $[A] = [L] [U]$, GFBS will solve the system of equations $[A] [X] = [L] [U] [X] = \pm [B]$

3.5.19.3 Calling Sequence

CALL GFBS (Z,Z)

COMMON/GFBSX/L(7),U(7),B(7),X(7),NZ,PREC,ISIGN

L,U,B,X - Matrix control blocks for the matrices [L], [U], [B], and [X].

X(5) - Desired output type for [X].

NZ - Number of computer words available at Z.

PREC - $\begin{cases} 1, \text{ use single precision arithmetic} \\ 2, \text{ use double precision arithmetic} \end{cases}$

ISIGN - $\begin{cases} 1, \text{ solve } [A] [X] = \pm [B]. \\ -1, \text{ solve } [A] [X] = -[B]. \end{cases}$

Z - An area of working storage.

3.5.19.4 Method

1. Mathematical Considerations. Given [L], [U] (n by n lower and upper triangular matrices) and [B] (a n by m matrix) GFBS solves the two systems of equations

$$[L] [Y] = \pm [B]$$

and

$$[U] [X] = [Y]$$

SUBROUTINE DESCRIPTIONS

Elements of [Y] and [X] are given by

$$y_{ij} = b_{ij} - \sum_{k=1}^{i-1} a_{ik} y_{kj} \quad (1)$$

$$x_{ij} = [y_{ij} - \sum_{k=i+1}^n u_{ik} x_{kj}] / u_{ii} \quad (2)$$

2. Initialization Phase. The type of arithmetic to be performed is computed as a function of the type of [L], [B], and the precision requested by the calling routine. Corresponding transfer vectors are set up to transfer control to the proper arithmetic computation. Core storage is filled with as many columns of [B] as possible.

3. Forward Pass. The intermediate values of [Y] are computed directly over the columns of [B] currently in core. This is a forward pass on [L] since it is read sequentially forward. Interpretation of Equation 1 shows that y_{1j} is complete after the first column of [L] has been read, y_{2j} after column 2, etc. Elements of [L] are read one at a time via INTPK and the appropriate term subtracted off.

4. Backward Pass. Elements of [X] are computed by processing [U] in the reverse order. Reading the last column of [U] completes $x_{n,j}$, the $n-1$ column of [U] gives $x_{n-1,j}$, etc. In order to facilitate the reading of [U], it was written in the reverse order by DECØMP, allowing a forward pass to be made in actuality.

5. Output Phase. The finished columns of [X] are packed and output via PACK. If more columns of [B] still exist, core is once again filled with vectors of [B] and the process repeated until all columns of [X] have been computed.

3.5.19.5 Design Requirements

At least one column of [B] must be unpacked in core, along with one GINØ buffer.

3.5.19.6 Diagnostic Messages

If insufficient core is available for GFBS, fatal message 3008 occurs.

If a diagonal term does not exist for a column of [U], fatal message 3005 occurs. This is normally detected in DECØMP implying care was not taken in processing singular matrices in the calling routine calling DECØMP.

3.5.20 SØLVER (Simultaneous Equation Solution Routine)

3.5.20.1 Entry Point: SØLVER.

3.5.20.2 Purpose

To perform the following three functions:

1. Solve $[A] [X] = -[B]$ for X where A is a real symmetric matrix which has been decomposed by SDCØMP.
2. Evaluate the matrix equation $[E] = [D] + [B]^T [X]$
3. On option, compute

$$\epsilon = \frac{\| [E] \|}{\| [D] \|}$$

3.5.20.3 Calling Sequence

CALL SØLVER(L,U,X,B,D,E,EPS,FLAG,SCR)

where:

L, U are the GINØ file names of the data blocks containing the lower and upper triangular factors of A , respectively.

X is the GINØ file name of the data block where the solution matrix will be written.

B is the GINØ file name of the data block containing the right hand matrix in $[A] [X] = -[B]$

D, E are the GINØ file names of the data blocks in Equation (2) above.

EPS is the real single precision result of the division in Equation (3).

SCR is the GINØ file name of a scratch file for use by SØLVER.

FLAG $\begin{cases} \neq 0, & \text{compute } \epsilon \text{ from Equation (3) above and store in EPS} \\ = 0, & \text{do not compute } \epsilon \end{cases}$

/SØLVRX/ is open core for SØLVER

Note:

1. L must be output from subroutine SDCØMP.
2. If $D = A$, then E is the error residual matrix.

SUBROUTINE DESCRIPTIONS

3.5.20.4 Method

/FBSX/ is initialized and FBS is called to solve $[A] [X] = -[B]$.

/MPYADX/ is initialized and MPYAD is called to compute $[E] = [D] + [B]^T [A]$. FLAG is tested. If FLAG = 0, return is made. Otherwise, INTPK is called to unpack and interpret each non-zero term of E. The cumulative sum of the absolute value of all non-zero terms of E is formed in double precision. This operation is then performed on D. Finally ϵ is the division of these two quantities and is stored in EPS in single precision.

3.5.20.5 Design Requirements

/SØLVRX/ must be inserted at the end of the overlay segment containing SØLVER.

3.5.20.6 Diagnostic Messages

The following system fatal message may be issued by SØLVER:

3001

MATRIX SUBROUTINE DESCRIPTIONS

3.5.21 DMPY (Multiply a Diagonal Matrix by an Arbitrary Matrix).

3.5.21.1 Entry Point: DMPY.

3.5.21.2 Purpose

To pre or post multiply an arbitrary matrix [B] by a diagonal matrix, i.e. $[C] = [D][B]$ or $[C] = [B][D]$.

3.5.21.3 Calling Sequence

CALL DMPY(Z,Z)

COMMON/DMPYX/D(7),B(7),C(7),NZ,FLAG,SIGN

Z - An area of working storage.

NZ - The number of computer words at Z.

D - Matrix control block for the diagonal matrix.

B - Matrix control block for the general matrix.

C - Matrix control block for the product matrix.

C(1) must contain the GINØ file name prior to entry.

C(5) must contain the arithmetic type of the elements of C.

DMPY will accumulate C(2) and C(6) and set C(7) = 0.

C(3) and C(4) may be set by the user before or after the call.

FLAG - $\begin{cases} = 0, & \text{pre-multiply B by D.} \\ \neq 0, & \text{post-multiply B by D.} \end{cases}$

SIGN - $\begin{cases} +1, & \text{form positive product.} \\ -1, & \text{form negative product.} \end{cases}$

3.5.21.4 Method

The type of arithmetic is determined (real or complex). The elements of the diagonal matrix, D, are unpacked at Z. INTPK is used to read and interpret the non-zero elements of B, column by column. The following equations are used:

Pre-multiplication: $c_{ij} = d_i b_{ij}$

Post-multiplication: $c_{ij} = b_{ij} d_j$

SUBROUTINE DESCRIPTIONS

3.5.21.5 Design Requirements

Double precision arithmetic is used throughout.

The amount of core at Z must be sufficient to hold the unpacked diagonal terms of D and two GINØ buffers.

The dimensions of D and B must be compatible although this is not checked.

3.5.21.6 Diagnostic Messages

The following system fatal messages may be issued by DMPY:

3001

3002

3.5.22 ELIM (Perform a Matrix Reduction).

3.5.22.1 Entry Point: ELIM.

3.5.22.2 Purpose

To perform a matrix reduction on the structural model by computing the matrix equation:

$$[K_{ii}] = [K_{ii}] + [K_{ji}]^T [G_j] + [G_j]^T [K_{ji}] + [G_j]^T [K_{jj}] [G_j]$$

3.5.22.3 Calling Sequence

```
CALL ELIM(KIIB,KJIB,KJJB,GJ,KII,SCR1,SCR2,SCR3)
```

where KIIB, KJIB, KJJB, GJ and KII are the GINØ file names of each of the matrices in the above equation. SCR1, SCR2 and SCR3 are GINØ file names of three scratch files used by ELIM.

Open core for ELIM is defined by /ELIMX/.

3.5.22.4 Method

ELIM computes the above matrix equation by three successive calls to MPYAD. These are as follows:

$$[SCR1] = [K_{jj}] [G_j] + [K_{ji}]$$

$$[SCR2] = [K_{ji}]^T [G_j] + [K_{ii}]$$

$$[K_{ii}] = [G_j]^T [SCR1] + [SCR2]$$

3.5.22.5 Design Requirements

/ELIMX/ must be included at the end of the segment in which ELIM resides.

3.5.23 FACTØR (Decompose a Matrix Into Triangular Factors).

3.5.23.1 Entry Point: FACTØR.

3.5.23.2 Purpose

To decompose a symmetric matrix into its two triangular factors.

3.5.23.3 Calling Sequence

CALL FACTØR (A,L,U,SCR1,SCR2)

where A, L, U are the GINØ file names of the data blocks for the symmetric matrix to be decomposed, its lower triangular factor and its upper triangular factor, respectively. If U is negative, FACTØR will decompose the symmetric matrix [A] on file A such that

$$[A] = [C][C]^T ,$$

where the Cholesky decomposition is used. [C] will be written on U as a lower triangular matrix. L will contain the lower triangular factor of a standard non-Cholesky decomposition. SCR1 and SCR2 are GINØ file names of two scratch files for use by FACTØR. The common block /FACTRX/ is open core for FACTØR.

3.5.23.4 Method

FACTØR initializes /SFACT/ and calls SDCØMP to accomplish the decomposition.

3.5.23.5 Design Requirements

/FACTRX/ must be included at the end of the segment which contains FACTØR.

3.5.23.6 Diagnostic Messages

The following system fatal message may be issued by FACTØR:

3005

MATRIX SUBROUTINE DESCRIPTIONS

3.5.24 TRANP1 (Driver for TRNSP).

3.5.24.1 Entry Point: TRANP1.

3.5.24.2 Purpose

To drive TRNSP to compute $[B] = [A]^T$

3.5.24.3 Calling Sequence

CALL TRANP1(FILEA,FILAT,NSCRTH,SCR1,SCR2,SCR3,SCR4,SCR5,SCR6,SCR7,SCR8).

FILEA - GINØ name of [A] - integer - input.

FILAT - GINØ name of $[A]^T$ - integer - input.

NSCRTH- Number of scratch files - integer - input.

SCR1, ... , SCR8 - GINØ names of scratch files - integer - input.

3.5.24.4 Method

/TRNSPX/ is set based on the trailer of FILEA.

3.5.24.5 Design Requirements

Open core at /DTRANX/.

3.5.25 TRNSP (Matrix Transpose).

3.5.25.1 Entry Point: TRNSP.

3.5.25.2 Purpose

To compute $[A]^T$ given $[A]$.

3.5.25.3 Calling Sequence

CALL TRNSP(Z)

Z - Array of core.

COMMON/TRNSPX/MCBA(7),MCBAT(7),LCØRE,NSCRTH,SCR(8)

MCBA - Matrix control block for $[A]$ - input.MCBAT - Matrix control block for $[A]^T$ - input.

LCØRE - Length of Z array - integer - input.

NSCRTH - Number of scratch files available - integer - input. $1 \leq \text{NSCRTH} \leq 8$.

SCR - List of GINØ names of scratch files - integer - input.

3.5.25.4 Method

Three methods are possible:

Method 1 - In-core matrices.

If the full matrix can be held in core, $[A]$ is unpacked into core, and packed out onto $[A]^T$.

Method 2 - Simple sub-matrices.

The matrix $[A]$ is broken up one column at a time into submatrices. The submatrices are written on scratch files, read in and transposed one at a time. The break-up of $[A]$ can be pictured as follows:

$$[A] = \begin{bmatrix} \text{SCRATCH 1} \\ \text{-----} \\ \text{SCRATCH 2} \\ \text{-----} \\ \text{SCRATCH 3} \\ \text{-----} \\ \text{etc.} \end{bmatrix}$$

MATRIX SUBROUTINE DESCRIPTIONS

If insufficient scratch files are available to hold [A], multiple passes may be made

Method 3 - Multiple passes on submatrices.

If multiple passes on [A] are necessary in Method 2, it may be more efficient to create larger submatrices and pass each submatrix several times rather than several passes on [A]. The break-up of [A] can be pictured as follows:

$$[A] = \left[\begin{array}{c} \text{CORE LOAD 1} \\ \text{---} \\ \text{CORE LOAD 2} \\ \text{CORE LOAD 3} \\ \text{---} \\ \text{CORE LOAD 4} \\ \text{---} \\ \text{etc.} \end{array} \right\} \begin{array}{c} \text{SCRATCH 1} \\ \text{---} \\ \text{---} \\ \text{SCRATCH 2} \\ \text{---} \\ \text{---} \end{array} \right]$$

TRNSP will choose between the methods to minimize running time.

3.5.25.5 Design Requirements

Methods 2 and 3 require at least one scratch file.

Neither [A] nor $[A]^T$ may be purged.

One unpacked column of [A] and NSCRTH+1 GINØ buffers must fit into core.

3.5.25.6 Diagnostic Messages

A message indicating insufficient core is produced if the third of the above requirements is not satisfied. Either reduce the number of scratch files or increase the open core available to TRNSP.

SUBROUTINE DESCRIPTIONS

3.5.26 SADD (Matrix Addition Routine)

3.5.26.1 Entry Point: SADD

3.5.26.2 Purpose

To compute $[X] = \alpha[A] + \beta[B] + \gamma[C] + \delta[D] + \epsilon[E]$

3.5.26.3 Calling Sequence

CALL SADD (Z,Z)

Z -- Array of core

COMMON/SADDX/NOMAT,LCORE,MCBA(7),TYPA,ALPHA(4),MCBB(7),TYPB,
BETA(4),MCBC(7),TYPC,GAMMA(4),MCBD(7),TYPD,DELTA(4),
MCBE(7),TYPE,EPSLN(4),MC(7)

NOMAT - Number of matrices to be added - integer - input.

LCORE - Length of Z array - integer - input.

MCBA - Matrix Control Block for [A] - input

TYPA - Type of ALPHA - integer - input

1 - real single precision

2 - real double precision

3 - complex single precision

4 - complex double precision

ALPHA - α - input - type depends on TYPA

MCBB - Matrix Control Block for [B] - input

TYPB - Type of BETA - integer - input

BETA - β - input - type depends on TYPB

MCBC - Matrix Control Block for [C] - input

TYPC - Type of GAMMA - integer - input

GAMMA - γ - input - type depends on TYPC

MCBD - Matrix Control Block for [D] - input

TYPD - Type of DELTA - integer - input

DELTA - δ - input - type depends on TYPD

MATRIX SUBROUTINE DESCRIPTIONS

MCBE - Matrix Control Block for [E] - input
TYPE - Type of EPSLN - integer - input
EPSLN - ϵ - input - type depends on TYPE
MC - Matrix Control Block for [X] - input

3.5.26.4 Method

The type of arithmetic is determined to be the maximum type of [A], [B], [C], [D], [E], α , β , γ , δ , ϵ .

Initially a column of [X] is zeroed out. Each non zero element of [A] is obtained by ZNTPKI, multiplied by α , and added into a corresponding position of [X]. After processing a complete column of [A], columns from matrices [B], [C], etc., are treated in a similar manner. When the contribution from the last matrix has been added in, a complete column of [X] is output via PACK. This procedure is repeated until all columns of [X] are output.

3.5.26.5 Design Requirements

Z must be sufficient to hold one unpacked column of [X] plus n GINØ buffers (where $n = NØMAT+1$).

SUBROUTINE DESCRIPTIONS

THE MATERIAL PREVIOUSLY IN SECTIONS 3.5.27 THRU 3.5.29

HAS BEEN DELETED.

3.5.30 MAKMCB (Make a Matrix Control Block)

3.5.30.1 Entry Point: MAKMCB

3.5.30.2 Purpose

To fill out a Matrix Control Block

3.5.30.3 Calling Sequence

CALL MAKMCB (MCB,IFILE,IRØW,IFØRM,ITYPE)

where:

MCB = a seven word array to be filled in - output
 IFILE = the GINØ file name of the matrix - integer - input
 IRØW = the number of rows in the matrix - integer - input
 IFØRM = the matrix form ($1 \leq IFØRM \leq 8$) - integer - input
 ITYPE = the matrix type ($1 \leq ITYPE \leq 4$) - integer - input

3.5.30.4 Method

MAKMCB fills MCB as follows:

MCB(1) = IFILE
 MCB(2) = 0
 MCB(3) = IRØW
 MCB(4) = IFØRM
 MCB(5) = ITYPE
 MCB(6) = 0
 MCB(7) = 0

SUBROUTINE DESCRIPTIONS

3.5.31 ATEIG (Find the Eigenvector of an Upper Hessenberg Matrix)

3.5.31.1 Entry Point: ATEIG

3.5.31.2 Purpose

To find the eigenvalues of an Upper Hessenberg Matrix.

3.5.31.3 Calling Sequence

CALL ATEIG(N,A,RR,RI,IANA,IA)

N - order of matrix

A - input matrix order NxN - Real

RR - vector for real parts of eigenvalues - Real-N-Output

RI - vector for imaginary parts of eigenvalues - Real-N-Output

IANA - Storage for extracting order vector - N-Integer

IA - IA=N for vector stored matrix

3.5.31.4 Method

The QR transform method is used with double iteration. The A matrix is destroyed.

MATRIX SUBROUTINE DESCRIPTIONS

3.5.32 HSBG (Reduce a Matrix to Upper Hessenberg Form)

3.5.32.1 Entry Point: HSBG

3.5.32.2 Purpose

To reduce a matrix to Upper Hessenberg Form.

3.5.32.3 Calling Sequence

CALL HSBG(N,A,IA)

N - order of matrix

A - input matrix order NxN - Real

IA - size of first dimension when A is double subscripted; IA=N for vector stored matrix.

3.5.32.4 Method

Similarity transformation using eigenvector elimination matrices with partial pivoting.
Should be used with ATEIG (see Section 3.5.31).

SUBROUTINE DESCRIPTIONS

3.5.33 INCØRE (Incore Complex Solve)

3.5.33.1 Entry Point: INCØRE

3.5.33.2 Purpose

To solve the simultaneous equation $[A][X] = [B]$

3.5.33.3 Calling Sequence

CALL INCØRE(A,N,B,X,IX)

A - A-matrix - Complex A(N,N)

N - Size of A

B - B-matrix - Complex B(IX,N)

X - X-matrix - Complex X(IX,N)

IX - Number of vectors in B

3.5.33.4 Method

Unsymmetric decomposition with pivoting on a square matrix is used. Once the decomposition is complete, a backward pass is made to get X.

3.5.33.5 Design Requirements

The core needed to use INCØRE is just the space needed for A, B and X.

Core = $2*(N*N + 2*(IX*N))$. A and B are destroyed.

MATRIX SUBROUTINE DESCRIPTIONS

3.5.34 EGNVCT (Calculate Eigenvectors)

3.5.34.1 Entry Point: EGNVCT

3.5.34.2 Purpose

To obtain eigenvectors from a matrix for which the eigenvalue is known.

3.5.34.3 Calling Sequence

CALL EGNVCT(C1,C2,EIGN,C3,N1,N2,N)

C1 - Matrix - Input - Complex - NxN - Integer

C2 - Scratch storage for complex-N

EIGN - Eigenvalue for which the eigenvector is wanted

C3 - Eigenvector - Complex - Output - N

N1 - Scratch storage - N - Integer

N2 - Scratch storage - N - Integer

N - Order of matrix C1

3.5.34.4 Method

The method used is the direct method derived by J. H. Wilkinson (see pp. 323, 626-628, Wilkinson, J. H., The Algebraic Eigenvalue Problem, Oxford: Clarendon Press, 1965.)

MATRIX SUBROUTINE DESCRIPTIONS

3.5.35 MPY3 (Triple Matrix Multiply)

3.5.35.1 Entry Point: MPY3DR

3.5.31.2 Purpose

To perform the matrix product $C = A^T B A + E$ or $BA + E$ for sparse A matrix and full B matrix.

3.5.35.3 Calling Sequence

CALL MYP3DR (FA,FB,FE,FC,S1,S2,S3,ICODE,IPREC,Z,LKØRE,NØMØD)

where:

FA = matrix A GINØ file number - integer - input

FB = matrix B GINØ file number - integer - input

FE = matrix E GINØ file number - integer - input

FC = matrix C GINØ file number - integer - input

Note: Matrices A, B and E must be in packed format and the trailers of matrices A, B, E and C (words 2, 6, 7 = 0) must be written ahead of time.

S1 = $\left. \begin{array}{l} S2 \\ S3 \end{array} \right\}$ the GINØ file numbers of three scratch files - integer - input

ICODE = 0, $A^T B A + E$ will be performed
2, $BA + E$ will be performed

IPREC = 1, output matrix will be in real single precision
2, output matrix will be in real double precision

Z = open core array

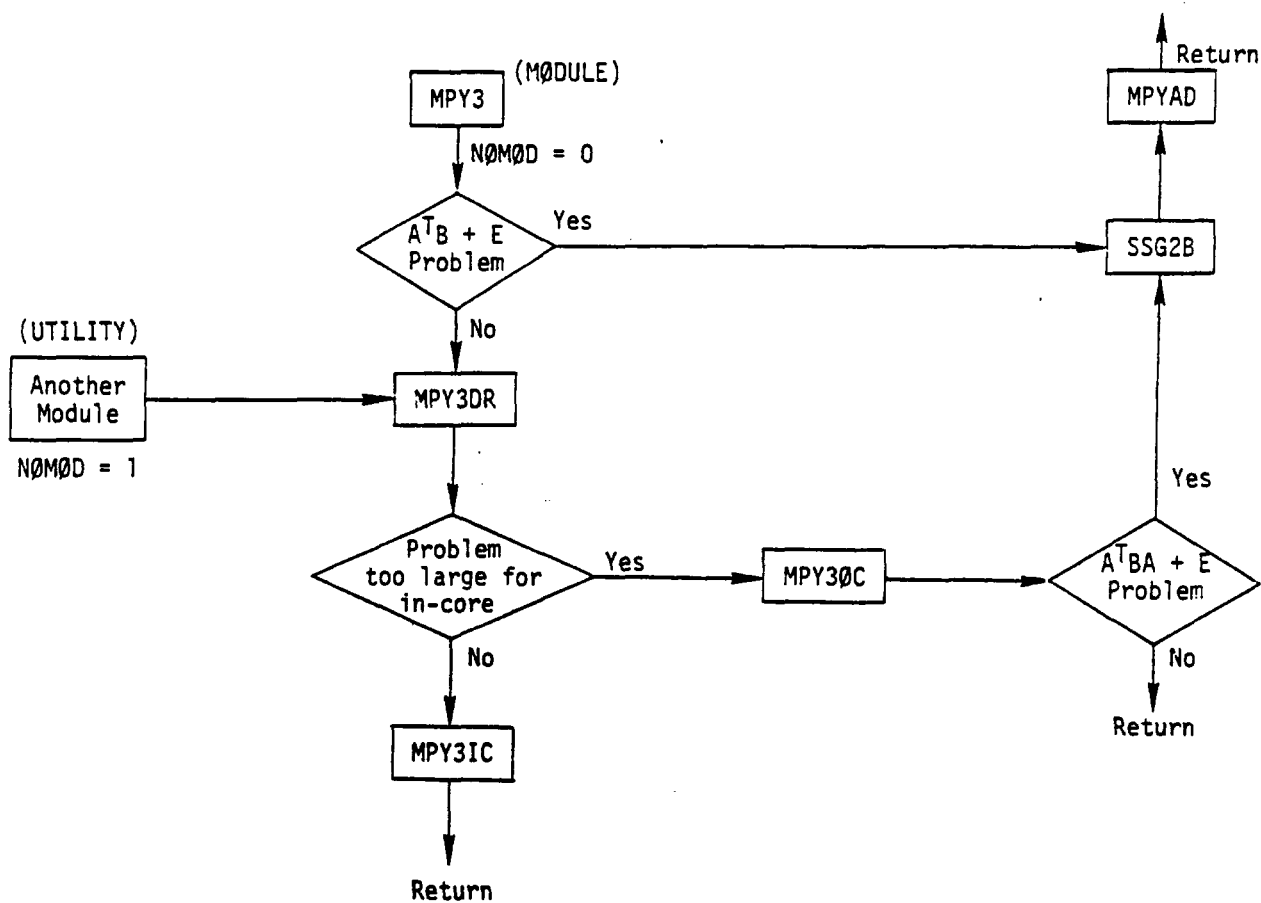
LKØRE = length of open core available

NØMØD = 1, a flag to indicate to MPY3 that it is to be used as a utility subroutine
(=0 if MPY3 treated as a functional module)

3.5.35.4 Method

The following chart describes the basic flow that MPY3 takes while executing as a module and as a utility subroutine.

SUBROUTINE DESCRIPTIONS



The following is a brief description of how MPY3 performs the computation. A more detailed description is found in Section 4.149.7.

3.5.35.4.1 In-Core and Out-of-Core Processing

MPY3DR, the entry point when MPY3 is executing as a utility subroutine, determines the size of the problem and branches to MPY3IC for in-core or MPY3OC for out-of-core processing accordingly.

The product matrix, C, is calculated one column at a time. MPY3B manages the computation until as many columns of the B matrix as possible are put in core. MPY3C then takes over managing the computation, substituting columns of B in core whenever necessary. Once MPY3B and MPY3C have made the necessary preparations, MPY3P, called by both MPY3B and MPY3C, performs the actual multiplications and sum accumulations.

3.5.35.4.2 Differences Between In-Core and Out-of-Core Processing

MPY3IC uses an in-core packed version of A^T to:

MATRIX SUBROUTINE DESCRIPTIONS

1. Provide terms when the problem is A^TBA .
2. Provide part of a system which manages the core-resident columns of the B matrix when the entire B matrix is too large to fit in core. MPY3NU determines which columns of B to be inserted and removed.

MPY3A sets up this packed version of A^T .

MPY30C calculates BA and then, if necessary, uses MPYAD to complete the triple product.

Hence:

1. A^T is not needed in core.
2. Management of the core-resident columns of B is done internally in MPY30C.

In MPY3IC, matrix E (if it exists) is summed one column at a time into the C matrix as each column of C is processed. In MPY30C, the same is done if the problem is $BA + E$. But if the problem is A^TBA , the entire summation of E is done in MPYAD.

3.5.35.5 Subroutines

The main subroutines used in this module are MPY3IC and MPY30C. These make use of the following subroutines:

MPY3A	Performs initial processing of A and B
MPY3B	Reads current column of A and performs initial computation of current column of A^TBA or BA
MPY3C	Continues computation until current column of A^TBA or BA is entirely computed
MPY3P	Called by MPY3B and MPY3C to perform actual computation
MPY3NU	Used in the in-core version to calculate next-time-used values
SSG2B	A matrix multiply routine described in Section 3.5.13

3.5.35.5.1 Subroutine Name: MPY3IC

1. Entry Point: MPY3IC
2. Purpose: To perform product with A matrix in core
3. Calling Sequence: CALL MPY3IC (Z,IZ,DZ,N0M00D)

Z Real single precision - open core

IZ Integer - open core

SUBROUTINE DESCRIPTIONS

DZ Real double precision - open core
NØMØD 0 - MPY3 executing as module
 1 - MPY3 executing as subroutine

3.5.35.5.2 Subroutine Name: MPY3ØC

1. Entry Point: MPY3ØC
2. Purpose: To perform product with A matrix out of core.
3. Calling Sequence: CALL MPY3ØC (Z,IZ,DZ,NØMØD)

Z Real single precision - open core
IZ Integer - open core
DZ Real double precision - open core
NØMØD 0 - MPY3 executing as module
 1 - MPY3 executing as subroutine

3.5.35.6 Common Blocks

1. /MPY3BC/ Type of solution; precision.
2. /MPY3TL/ Matrix trailers, open core positions of buffers, and a logical variable denoting the existence or non-existence of matrix E.
3. /MPY3CP/ Various parameters needed by more than one subroutine, including pointers to positions in open core.

3.6.36.7 Design Requirements

Open core is used as described in Section 4.149.10 of the Programmer's Manual.

3.5.35.8 Diagnostic Messages

Fatal messages 6651, 6552, 6553, 6554, 6555, 6556, and 6557 may be issued.

SUBROUTINE DESCRIPTIONS

3.6 SUBSTRUCTURE OPERATING FILE (SØF) UTILITIES

The Substructure Operating File (SØF) is a permanent storage file for the user substructuring data. It is physically stored on a user diskpack, drum, or equivalent device, and can therefore be used to communicate data between different phases of a multi-stage substructuring problem or different groups working on the same substructuring problem.

In addition to the user data, the SØF contains tables created by the SØF utility subroutines which allow the full trace back of the substructuring process.

To avoid confusion in what follows, a summary of the nomenclature used in defining the data structure is needed.

- The unit of data is a substructure.
- Each substructure has a fixed number of items associated with it. The number and properties of these items are described in the ITEM DT executive table (see section 2.4.2.10).
- Each item is divided into an arbitrary number of groups of variable length.

3.6.1 Tables Associated with the SØF

3.6.1.1 The Array NXT

The different tables associated with the SØF will grow at arbitrary times, and the components of different substructures will be created at arbitrary times. Storing data sequentially on the SØF will consequently be wasteful of storage space if not impossible. The SØF has therefore been divided into equal size blocks where data is stored sequentially. A block is also the unit of input/output operations, that is, only one block is read (from the SØF) or written (on the SØF) at a time. The size of a block is identical to the GINØ block size as specified in the /GINØX/ common block.

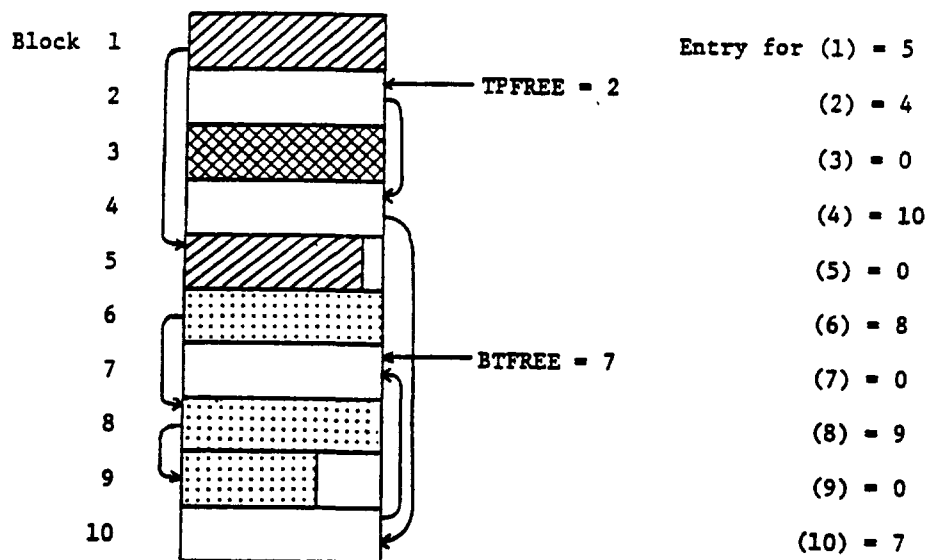
Different tables may extend over several noncontiguous blocks and data related to different items may also extend over several noncontiguous blocks. The array NXT is used to chain together all related data blocks.

Two pointers TPFREE, and BTFREE are used with NXT where TPFREE points to the first free block BTFREE to the last free blocks. Each block of the SØF has an entry in the array NXT. The array contains the following information:

SUBROUTINE DESCRIPTIONS

- If a block is being used, the entry for the block contains the index of the next block linked to it (e.g., block 1 is linked to block 5 in the sketch below) or zero if no further blocks are linked (e.g., see block 9 below).
- If a block is free, the entry for the block contains the index of the next free block (e.g., free blocks 2, 4, 10 and 7 are linked together). However, the entry for the block is zero if it corresponds to BTFREE (e.g., block 7).

When data is deleted, blocks that were used by it will be returned to the list of free blocks and BTFREE updated accordingly. BTFREE will likewise be updated when free storage blocks are used to store more data. If no free blocks are available, TPFREE = BTFREE = 0 (no blocks = no entries).



3.6.1.2 The Directory Index Table (DIT)

The Directory Index Table contains the names of all the substructures that are stored on the SØF. The name of each substructure consists of two BCD words of four characters each.

SUBSTRUCTURE OPERATING FILE (SØF) UTILITIES

DIT

1	—	SNAME1	—
2			
3			
4	—	SNAME2	—
5			
6			
...	...	SNAME3	...
...
...

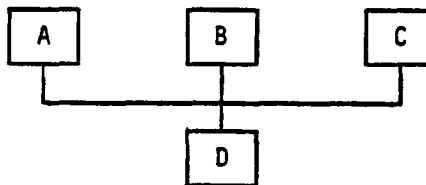
The DIT provides an easy way of checking for the existence of a substructure on the SØF.

3.6.1.3 The Master Data Index (MDI)

The Master Data Index contains a directory for each substructure on the SØF, that is, a collection of pointers to data related to the substructure. The size of a directory is $NITEM + 2$ words and is denoted by DIRSIZ. (NITEM is contained in the ITEMMDT table) Each directory should contain the information indicated in Figure 1.

A substructure having the I^{th} name in the DIT will have the I^{th} directory in the MDI and will have I for its internal index.

A few points should be made clear. If substructures A, B and C are combined to produce substructure D, then each of A, B and C is a lower level substructure to D, and D is a higher level substructure to A, B and C.



The entry LL of D will contain the index of A, B or C

The entry HL of A will contain the index of D

The entry HL of B will contain the index of D

The entry HL of C will contain the index of D

The entry CS of A will contain the index of B

The entry CS of B will contain the index of C

The entry CS of C will contain the index of A

Note that the link through combined substructures is circular.

SUBROUTINE DESCRIPTIONS

Position	MDI Contents											
	31	30	29	28	27	26	25	24	23	22	21	20
J + 1	TYPE						:SS				PS	
2							:CS				HL	
3	ITEM #1 Length						ITEM #1 Block Index					
4	ITEM #2 Length						ITEM #2 Block Index					
:												
:												
DIRSIZ	ITEM # NITEM Length						ITEM # NITEM Block Index					

$$J = (I-1) * DIRSIZ$$

TYPE = Substructure type bits. Current bit positions are

- 30 - Image substructure
- 29 - Combined substructure
- 28 - Guyan reduced substructure
- 27 - Modal reduced substructure
- 26 - Complex modal reduced substructure
- a basic substructure has bits 20-29 off.

SS = Index of the secondary equivalent substructure
 PS = Index of the primary equivalent substructure
 LL = Index of one of the lower level substructures
 CS = Index of one of the substructures that are combined together
 HL = Index of the higher level substructure

Figure 1. Ith Directory in the Master Data Index

SUBSTRUCTURE OPERATING FILE (SØF) UTILITIES

If a substructure does not have a lower level, or a higher level substructure, or is not combined with any substructure then its LL, HL or CS entries will respectively be equal to zero.

If substructure F is reduced from substructure E, then E is a lower level substructure to F, and F is a higher level substructure to E.



The entry LL of F will contain the index of E

The entry HL of E will contain the index of F

The entry CS of E will contain zero

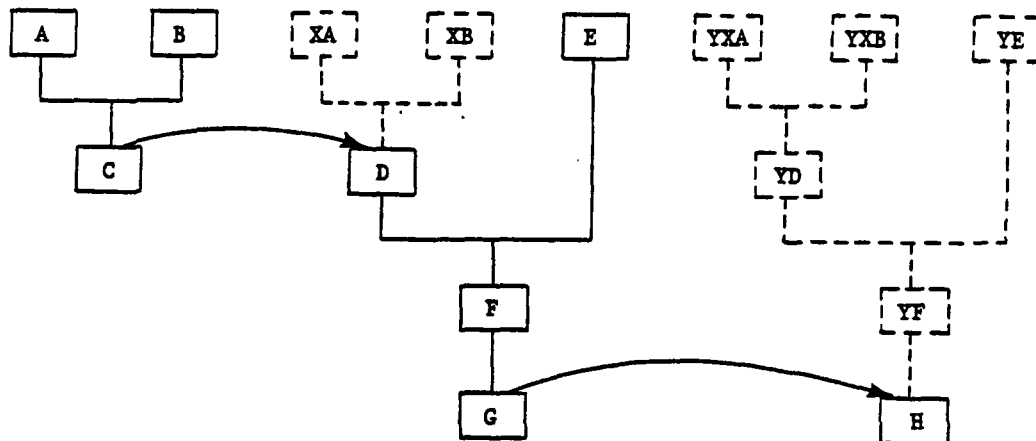


Figure 2. Multi-level substructuring.

Suppose that in Figure 3.6.1-2, D is set equivalent to C with prefix X and later H is set equivalent to G with prefix Y. The substructures A, B, C, E, F and G will be called primary substructures, D and H will be called secondary substructures and XA, XB, YXA, YXB, YD, YE and YF will be called image substructures. Image substructures are also secondary substructures.

The substructure A has two secondary substructures: XA and YXA. Its SS entry will however point to only one of them. Assume it points to XA, the SS entry of XA will then point to YXA. All secondary substructures relative to a primary substructure are thus chained together through the SS entry with the SS entry of the last substructure containing zero. The SS entry of YXA will

SUBROUTINE DESCRIPTIONS

therefore contain zero. The PS entry of XA and YXA will point to A, while that of A will contain zero because A is a primary substructure.

The substructure C has two secondary substructures D and YD, where YD is an image substructure and D is not. The IS bit of all image substructures will be set to one, while that of all other substructures will be zero.

A diagram of these pointers is illustrated in Figure 3.

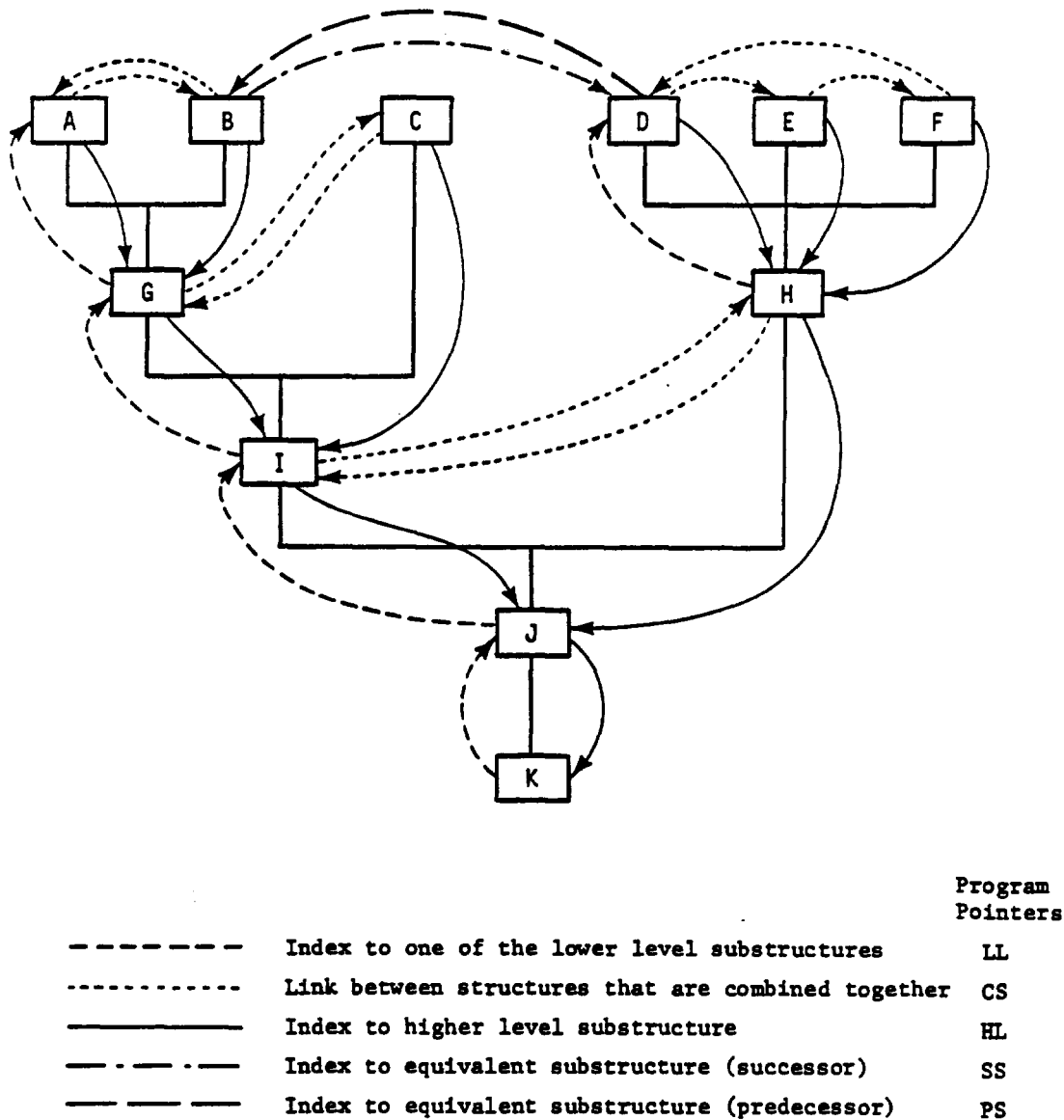


Figure 3. Diagram of Master Data Index (MDI) pointers.

SUBSTRUCTURE OPERATING FILE (SØF) UTILITIES

Data related to an item starts always at the beginning of a block and may, if needed, extend over several blocks with the possibility of the last block being only partially used. Those blocks are not necessarily contiguous, but are chained together through the array NXT. If an item's data is deleted at any point, the blocks used by the data are returned to the list of free blocks and may be used again.

The end of a group and the end of an item are both denoted by special symbols, (SEOG and SEOI) and stored as part of the data on the SØF. The entry in the MDI which indicates the length of an item is expressed in number of blocks and is used to compute the approximate item size for the table of contents printout. The entry for an item's block number contains the physical block number of the first SØF block on which the item's data is stored. If no data is stored for an item, both its length and block number entries will be set to zero.

3.6.2 Organization of the SØF

The user of the NASTRAN program is allowed to allocate several files of different sizes to the SØF. The number of files allocated by the user is saved in the common block /SØFCØM/ and is denoted by NFILES. The maximum number that may be allocated is ten. The size of each SØF file expressed in number of blocks is stored in the array FILSIZ and also saved in /SØFCØM/.

The first physical block of each SØF file is reserved to store the SØF password, the file sequence number, the common blocks /SØF/ and /SYS/ used by the SØF utility subroutines, and the item structure for the SØF. This information is saved at the end of each module using the SØF subroutines for protection reasons and because different overlays might wipe out the SØF common blocks. The exact format for this data is described in the commented listing for subroutine SØFINT (section 3.6.28).

At the first stage of a substructuring process, blocks that are chained together will be physically very close if not contiguous. However, as the process goes on data will be edited out and blocks used by that data will be used over for other data, thus causing chained blocks to be farther apart. It is desirable to keep chained blocks as close to each other as possible. The blocks of each SØF file have for that reason been grouped into super blocks. The number of blocks contained in a superblock is equal to $2 * (BLKSIZ - 1)$.

The partitioning of the SØF files is illustrated in Figure 4.

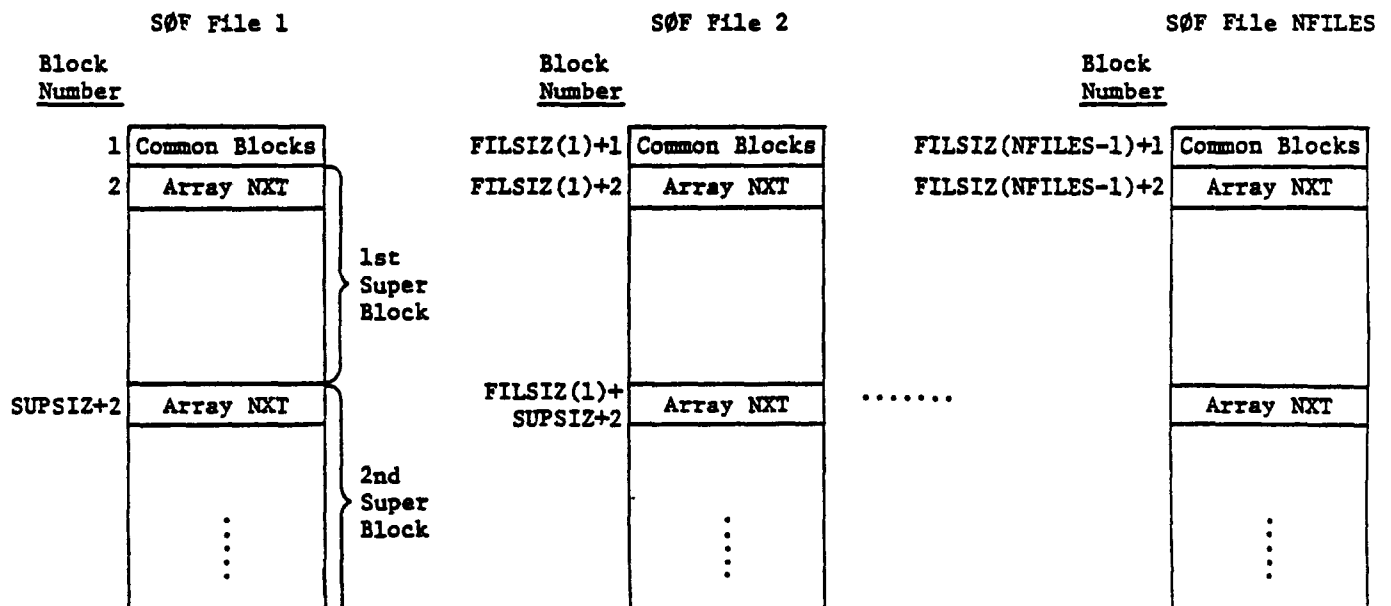


Figure 4. Partitioning of the SØF files.

SUBSTRUCTURE OPERATING FILE (SØF) UTILITIES

The first block of a superblock contains the array NXT for that particular superblock. The array NXT of the first SØF superblock contains the information indicated in Figure 5.

Word Number		31	16 15	0
1		TPFREE		BTFREE
2		Index of block chained to block 2		Index of block chained to block 3
3		Index of block chained to block 4		Index of block chained to block 5
		⋮		⋮
BLKSIZ		Index of block chained to block 2*BLKSIZ-2		(Not Used)

Figure 5. Array NXT of the first SØF superblock.

Note that the entry for even numbered blocks is in the left part of a word, whereas that of odd numbered blocks is in the right part. In order for that arrangement to hold true for the array NXT of each superblock, the size of each SØF file is required to be of an even number of blocks.

The SØF utility subroutines will use the first superblock until all its free blocks are exhausted and then will start using the second superblock, until it in turn is exhausted. However, before moving on to the third block, the list of free blocks of the first superblock and then that of the second will be examined. Thus, all free space of all previous superblocks will be used before a new superblock is acquired. Note that each superblock has its own list of free blocks, and its own pointers TPFREE and BTFREE which are stored in the first word of its array NXT. The procedure described above will be similarly followed to get access to more superblocks, and when one SØF file is exhausted, to another SØF file.

The last superblock on each one of the SØF files will not necessarily contain 2*(BLKSIZ-1) blocks, but will still be handled as a regular size superblock with its own array NXT, list of free blocks, and TPFREE and BTFREE pointers.

SUBROUTINE DESCRIPTIONS

3.6.3 In-core Handling of the SØF Data and Tables

As we have already seen, the array NXT may extend over more than one block. The MDI and the DIT will similarly grow and extend over several blocks. In order to store, modify, or delete information in those tables, portions of them must be in core.

Core is also the intermediate storage area for input/output data. To reduce the number of input/output operations, only one block of data is read or written at a time. A buffer is consequently reserved to store data temporarily until a whole block of data is available for output, or to store an input data block and then read portions of it as desired.

Three in-core buffers are needed by the SØF utility subroutines to handle the SØF data and tables. Those buffers have to be allocated by the module writer who uses the SØF routines, and passed as parameters through a call to the subroutine SØFØPN. The size of each buffer is equal to the contents of the first word of /SYSTEM/.

3.6.3.1 Buffer Used by the Array NXT and the DIT

Only one block of either the array NXT or the DIT remains in core at any time. The core occupied by that block is located by the SØF routines relative to the origin of blank common. It starts at location (DIT + 1) relative to the beginning of blank common and extends to location (DIT + BLKSIZ). DIT is used as a pointer into blank common which indicates the beginning of the block relative to the beginning of blank common. The buffer passed by the module writer starts at location (DIT - 2). No information is stored in the first three words of that buffer, thus making possible the use of someGINØ subroutines to enable the input/output operations between core and the SØF.

When a block of the array NXT is in core, the physical block number of that block is stored in the variable NXTPBN, and its logical block number is stored in NXTLBN. If no blocks of the array NXT are in core both NXTPBN and NXTLBN are set to zero. The same holds true for the DIT with NXTPBN and NXTLBN replaced by DITPBN and DITLBN.

3.6.3.2 Buffer used by the MDI

Only one block of the MDI remains in core at any time. The handling of the MDI buffer is similar to that of the buffer used by NXT and the DIT. The in-core block of the MDI is stored in core starting at location (MDI + 1) relative to the beginning of blank common and extending to location (MDI + BLKSIZ). The variable MDIPBN contains the physical block number of the in-core MDI block,

SUBSTRUCTURE OPERATING FILE (SOF) UTILITIES

and MDILBN contains its logical block number. Both variables are zero if no blocks of the MDI are in core.

3.6.3.3 Buffer Used for Input/Output of Data

Only one buffer is used for both input and output. The in-core buffer is stored in core starting at location $(I\emptyset + 1)$ relative to the beginning of blank common and extending to location $(I\emptyset + BLKSIZ)$. The input/output buffer is handled the same way the above two buffers are handled. The variable IOPBN contains the physical block number of the block being read or written and IOLBN contains its logical block number.

SUBROUTINE DESCRIPTIONS

3.6.4 Index for SØF Utility Functions and Subroutines

<u>Section Number</u>	<u>Function/Subroutine</u>	<u>Page Number</u>
3.6.5	CHKPØN	3.6-13
3.6.6	CRSUB	3.6-14
3.6.7	DELETE	3.6-15
3.6.8	DSTRØY	3.6-16
3.6.9	EDIT	3.6-18
3.6.10	EQSØF	3.6-19
3.6.11	ERRMKN	3.6-20
3.6.12	FDIT	3.6-21
3.6.13	FDNAME	3.6-22
3.6.14	FDSUB	3.6-23
3.6.15	FMDI	3.6-24
3.6.38	FNDLVL	3.6-54
3.6.16	FNDNXL	3.6-25
3.6.17	FNXT	3.6-26
3.6.18	GETBLK	3.6-27
3.6.19	ITCØDE	3.6-28
3.6.37	ITTYPE	3.6-53
3.6.20	MTRXI	3.6-29
3.6.21	MTRXØ	3.6-30
3.6.22	RETBK	3.6-31
3.6.23	SETEQ	3.6-32
3.6.24	SETLVL	3.6-34
3.6.25	SFETCH	3.6-36
3.6.26	SJUMP	3.6-38
3.6.35	SMSG	3.6-51
3.6.27	SØFCLS	3.6-39
3.6.28	SØFINT	3.6-40
3.6.29	SØFIØ	3.6-42
3.6.30	SØFØPN	3.6-43
3.6.37	SØFTRL	3.6-52
3.6.31	SØFSIZ	3.6-44
3.6.32	SUREAD	3.6-45
3.6.33	SUWRT	3.6-46
<u>SØF Utility Common Blocks</u>		
3.6.34.1	/SØF/	3.6-47
3.6.34.2	/SØFCØM/	3.6-49
3.6.34.3	/SYS/	3.6-50

SUBSTRUCTURE OPERATING FILE (SØF) UTILITIES

3.6.5 CHKØPN

3.6.5.1 Entry Point: CHKØPN

3.6.5.2 Purpose

To check if a call to SØFØPN has been made prior to calling any of the SØF utility subroutines.

3.6.5.3 Calling Sequence

CALL CHKØPN (SBRNM)

SBRNM - Name of the subroutine which issued the call to CHKØPN - array of dimension two
where each word is 4 characters long - BCD - input.

3.6.5.4 Method

The logical variable ØPNSØF contained in /SØFCØM/ is checked. If the value of the variable is .TRUE., return from CHKØPN is normal. If the value is .FALSE., a fatal error message is issued.

3.6.5.5 Diagnostic Messages

User Fatal Message 6204.

SUBROUTINE DESCRIPTIONS

3.6.6 CRSUB (Create Substructure)

3.6.6.1 Entry Point: CRSUB

3.6.6.2 Purpose

To create an entry for the substructure NAME in the DIT.

3.6.6.3 Calling Sequence

CALL CRSUB (NAME,I)

NAME - Name of the substructure to be created - array of dimension two where each word is
4 characters long - input

I - Index in the DIT of the substructure NAME - integer - output

3.6.6.4 Method

Stores NAME in the DIT. If there are any empty entries within the DIT, NAME is stored in the first one, otherwise NAME is stored at the end of the DIT.

3.6.6.5 Diagnostic Messages

System Fatal Message 6224.

3.6.7 DELETE

3.6.7.1 Entry Point: DELETE

3.6.7.2 Purpose

To delete an item which belongs to a given substructure.

3.6.7.3 Calling Sequence

CALL DELETE (NAME,ITEM,ITEST)

NAME - Name of the substructure whose item is to be deleted - array of dimension two
where each word is 4 characters long - input

ITEM - Name of the item which is to be deleted - four BCD characters - input

ITEST - Status of NAME and ITEM - integer - output

= {
1 if ITEM does exist
2 if ITEM pseudo exists
3 if ITEM does not exist
4 if NAME does not exist
5 if ITEM is an illegal item name

3.6.7.4 Method

The pointers belonging to ITEM are set to zero in the MDI. For primary substructures, all blocks on which data is stored for the item are returned to the list of free blocks. For secondary and image substructures, the blocks are returned only if the corresponding field in the ITEMMDT table is non-zero. If NAME is a primary substructure, ITEM is also deleted from each secondary substructure if it occupies the same blocks (i.e., KMTX, MMTX, etc.).

SUBROUTINE DESCRIPTIONS

3.6.8 DSTRØY

3.6.8.1 Entry Point: DSTRØY

3.6.8.2 Purpose

To remove from the SØF all the items belonging to a substructure and to all subsequent level substructures.

3.6.8.3 Calling Sequence

CALL DSTRØY (NAME, ITEST)

NAME - Name of the substructure which is to be destroyed - array of dimension two, where each word is 4 characters long - input

ITEST - Integer - output

= { 1 normal return
4 if NAME does not exist
6 if NAME is an image substructure

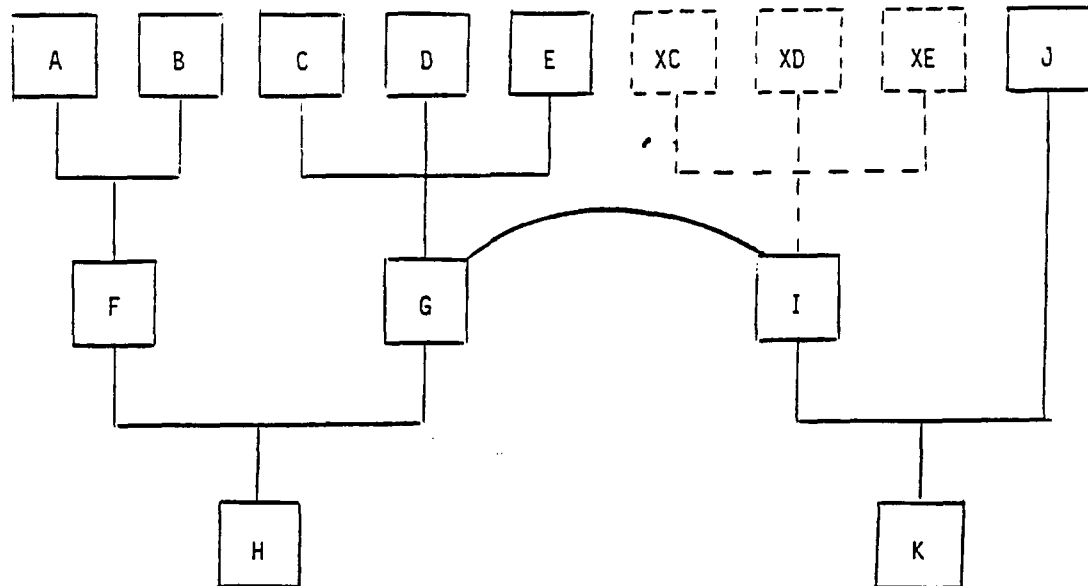
3.6.8.4 Method

Destroys the substructure NAME by deleting its directory from the MDI, and its name from the DIT. No operation will take place if NAME is an image substructure. If NAME is a secondary substructure, it is deleted from the list of secondary substructures to which it belongs, and its image contributing tree is destroyed. If NAME is a primary substructure, all its secondary substructures are also destroyed. In all cases, all the substructures derived from a substructure being destroyed are also destroyed, and connections with other substructures are deleted. If one of those that are destroyed is the result of combining several substructures together, or a reduction, higher level descriptive items (nonzero HIGHER entry in table ITEM DT) of those lower level substructures will be deleted. For primary substructures, all blocks on which data is stored for the substructure are returned to the list of free blocks. For secondary and image substructures, the blocks are returned only if the corresponding field in the ITEM DT table is nonzero.

Example:

Suppose that substructures are set up as follows:

SUBSTRUCTURE OPERATING FILE (SØF) UTILITIES



A call to DSTRØY with NAME equal to G will generate the following sequence of operations:

1. Destroy G

- Return all blocks used by primary substructure G to the list of free blocks.
- Put I on a stack of substructures to destroy it later.
- Destroy image substructures SC, SD and SE and return blocks unique to them (non zero image field in table ITEMØT).
- Delete G's directory from the MDI, and its name from the DIT.
- Delete link through substructures that were combined with G (i.e., F) to produce a higher level substructure (H), and delete higher level descriptive items from those substructures.
- Delete the link through the combined substructures C, D, and E, and their pointer to G. Also delete higher level descriptive items from C, D and E.

2. Destroy H

- Return all the blocks used by H to the list of free blocks.
- Delete H's directory from the MDI, and its name from the DIT.

3. Destroy I

- Return blocks unique to secondary substructure I (non zero secondary field in table ITEMØT).
- Delete I's directory from the MDI, and its name from the DIT.

SUBROUTINE DESCRIPTIONS

- c. Delete link through I and J (combined substructures), J's pointer to K, and higher level descriptive items of J.
- 4. Destroy K
 - a. Return all blocks used by K to the list of free blocks.
 - b. Delete K's directory from the MDI, and its name from the DIT.

SUBSTRUCTURE OPERATING FILE (SØF) UTILITIES

THIS PAGE HAS BEEN LEFT BLANK INTENTIONALLY.

SUBROUTINE DESCRIPTIONS

3.6.9.1 Entry Point: EDIT

3.6.9.2 Purpose

To remove selected items of a substructure from the SØF.

3.6.9.3 Calling Sequence

CALL EDIT (NAME,IØPT,ITEST)

NAME - Name of the selected substructure - array of dimension two where each word is 4 characters long - input.

IØPT - Option code - integer - input. The value of IØPT is the bit pattern representing the edit groups which are to be removed. See the description of the ITEMMDT table (section 2.4.2.10) for the available values.

ITEST = Integer - output.

1 normal return

*

4 if NAME does not exist

3.6.9.4 Method

The IØPT value is tested against the EDIT field in the ITEMMDT table for each item. If a match occurs, the pointers belonging to the item are removed from the directory of substructure NAME. For primary substructures, all blocks on which data is stored for the item are returned to the list of free blocks. For secondary and image substructures, the blocks are returned only if the corresponding field in the ITEMMDT table is non zero.

If NAME is a primary substructure, selected items are also deleted from each secondary substructure if it occupies the same blocks.

SUBSTRUCTURE OPERATING FILE (SØF) UTILITIES

3.6.10 EQSØF (Equate SØF)

3.6.10.1 Entry Point: EQSØF

3.6.10.2 Purpose

To set the pointers belonging to ITEM of NAME2 to the same value as those belonging to ITEM of NAME1.

3.6.10.3 Calling Sequence

CALL EQSØF (NAME1,ITEM,NAME2,ITEST)

NAME1 - Name of a substructure - array of dimension two where each word is 4 characters long - input.

ITEM - Selected item which is to be identical for substructures NAME1 and NAME2 - four BCD characters - input.

NAME2 - Name of substructure which is to have the pointers belonging to ITEM set to the same value as those belonging to ITEM of substructure NAME1 - array of dimension two where each word is 4 characters long - input

ITEST - Integer - output

1 normal return

4 if NAME1 does not exist

5 if ITEM is an illegal item name

7 if NAME 2 does not exist

3.6.10.4 Method

Fetches from NAME1's directory in the MDI the desired pointers, and stores them in NAME2's directory. It is not required that NAME1 and NAME2 be equivalent substructures. If they are not equivalent, it will be necessary that the user remembers to delete the pointers belonging to ITEM from one substructure's directory if the same ITEM has been deleted or edited from the other substructure, or if the other substructure has been destroyed.

SUBROUTINE DESCRIPTIONS

3.6.11 ERRMKN

3.6.11.1 Entry Point: ERRMKN

3.6.11.2 Purpose:

To write system error messages for the SØF subroutines.

3.6.11.3 Calling Sequence

CALL ERRMKN (N,IERR)

N - Code of the calling subroutine - integer - input

IERR - Error code - integer - input

SUBSTRUCTURE OPERATING FILE (SOF) UTILITIES

3.6.12 FDIT (Fetch DIT)

3.6.12.1 Entry Point: FDIT

3.6.12.2 Purpose

To fetch from direct access storage device, the desired block of the DIT, and store it into core in the array BUF.

3.6.12.3 Calling Sequence

CALL FDIT (I,K)

I - Index of the substructure whose DIT entry is desired - integer - input

K - Index in the array BUF of the desired entry - integer - output

3.6.12.4 Method

The DIT logical block number, on which the I^{th} substructure is written, is computed, and the corresponding physical block number is derived by threading through the linked blocks of the DIT. The desired block is stored in the array BUF starting at location (DIT + 1) and extending to location (DIT + BLKSIZ). The I^{th} substructure's index in BUF is returned in the variable K.

3.6.12.5 Diagnostic Messages

User Fatal Message 6223.

SUBROUTINE DESCRIPTIONS

3.6.13 FDNAME

3.6.13.1 Entry Point: FDNAME

3.6.13.2 Purpose

To find the name of the primary substructure equivalent to a given substructure.

3.6.13.3 Calling Sequence

CALL FDNAME (NAME,NEWNM)

NAME - Name of the substructure whose primary equivalent substructure's name is desired - array of dimension two, where each word is 4 characters long - input.

NEWNM - Name of the primary substructure equivalent to NAME. If no primary substructure is equivalent to NAME, NAME will be returned to NEWNM. If NAME is not known to the system, blanks will be returned to NEWNM - array of dimension two, where each word is 4 characters long - output.

3.6.13.4 Method

A search for NAME is carried in the DIT. If NAME is not found, blanks are returned in NEWNM. If NAME is found, its directory is fetched from the MDI. If NAME does not have a primary substructure, NAME will be returned in NEWNM, otherwise, the name of the primary substructure will be returned in NEWNM.

SUBSTRUCTURE OPERATING FILE (SØF) UTILITIES

3.6.14 FDSUB

3.6.14.1 Entry Point: FDSUB

3. .14.2 Purpose

To find the name of a substructure in the DIT.

3.6.14.3 Calling Sequence

CALL FDSUB (NAME,I)

NAME - Name of the substructure which is to be found in the DIT - array of dimension two, where each word is 4 characters long - input.

I - Index, in the DIT, of the substructure NAME. If NAME is not found in the DIT, I is set to -1 - integer - output.

3.6.14.4 Method

Successive blocks of the DIT are copied from the direct access storage device into core, and NAME is searched for in each block.

SUBROUTINE DESCRIPTIONS

3.6.15 FMDI (Fetch MDI)

3.6.15.1 Entry Point: FMDI

3.6.15.2 Purpose

To fetch from the direct access storage device the block of the MDI containing the directory of a particular substructure.

3.6.15.3 Calling Sequence

CALL FMDI (I,J)

I - Index of the substructure whose directory is desired - integer - input

J - (Index - 1) of the desired directory in the array BUF. The array BUF is located in the blank common block and is shared by all SØF utility subroutines - integer - output.

3.6.15.4 Method

The MDI logical block number of the block, in which the Ith substructure has its directory, is computed from the index I, and the index of the directory in that block is also computed. The desired block is then read into core.

3.6.15.5 Diagnostic Messages

User Fatal Message 6223.

SUBSTRUCTURE OPERATING FILE (SOF) UTILITIES

3.6.16 FNDNXL

3.6.16.1 Entry Point: FNDNXL

3.6.16.2 Purpose

To find the name of a higher level substructure to a given substructure.

3.6.16.3 Calling Sequence

CALL FNDNXL (NAME,NEWNM)

NAME - Name of the substructure whose higher level substructure's name is desired - array of dimension two where each word is 4 characters long - input.

NEWNM - Name of the higher level substructure to NAME. If NAME does not have a higher level substructure, NAME will be returned in NEWNM. If NAME is not known to the system, blanks will be returned in NEWNM - array of dimension two, where each word is 4 characters long - BCD - output.

3.6.16.4 Method

A search for NAME is carried in the DIT. If NAME is not found, blanks are returned in NEWNM. If NAME is found, its directory is fetched from the MDI. If NAME does not have a higher level substructure, NAME will be returned in NEWNM, otherwise the name of the higher level substructure will be returned in NEWNM.

SUBROUTINE DESCRIPTIONS

3.6.17 FNXT (Fetch NXT)

3.6.17.1 Entry Point: FNXT

3.6.17.2 Purpose

To fetch from the direct access storage device, the block of the array NXT on which a physical block number has an entry.

3.6.17.3 Calling Sequence

CALL FNXT (I,J)

I - Physical block number of the block whose entry in the array NXT is desired - integer - input.

J - Index of the desired entry in the array BUF. The array BUF is located in the blank common block and is shared by all SØF utility subroutines - integer - output.

3.6.17.4 Method

The physical block number of the desired block of the array NXT is computed from I, the desired block is then read into core.

3.6.17.5 Diagnostic Messages

System Fatal Message 6224

SUBSTRUCTURE OPERATING FILE (SOF) UTILITIES

3.6.18 GETBLK

3.6.18.1 Entry Point: GETBLK

3.6.18.2 Purpose

To return the physical block number of a free block.

3.6.18.3 Calling Sequence

CALL GETBLK (IOLD, INEW)

IOLD - Physical block number of a block which has been completely used up and needs to be extended with a new block. If IOLD is zero, it indicates that there is no old block that needs to be extended - integer - input.

INEW - Physical block number of a free block - INEW is set to -1 if there are no more free blocks available - integer - output.

3.6.18.4 Method

The current superblock is checked to see if there are any free blocks on it. If there are no more free blocks on it, the subroutine moves on to check the next superblock. If after checking all available superblocks, no free block can be found, the subroutine sets up a new superblock (unless all provided storage devices have been completely exhausted), and increments the number of available superblocks by one. INEW is set to the physical block number of the found free block, and the number of that block is removed from the list of free blocks in which it was found. If IOLD is not zero, the entry of IOLD in the array NXT is set to INEW indicating that the next block after IOLD is INEW.

3.6.18.5 Diagnostic Messages

System Fatal Message 6224.

SUBROUTINE DESCRIPTIONS

3.6.19 ITCØDE (Item Code)

3.6.19.1 Entry Point: ITCØDE

3.6.19.2 Purpose:

To associate with every item a code number which is the item's index in an MDI directory.

3.6.19.3 Calling Sequence

K = ITCØDE (ITEM) Integer function

ITEM - Any of the legal item names: four BCD characters - input.

The function will return the following

n - items index in a MDI entry

-1 - illegal item name

3.6.19.4 Method

The ITEMØT table is searched for the occurrence of the requested item. If no match is found, a negative number is returned. If a match is found its relative position in a MDI entry is computed and returned.

SUBSTRUCTURE OPERATING FILE (SØF) UTILITIES

3.6.20 MTRXI

3.6.20.1 Entry Point: MTRXI

3.6.20.2 Purpose

To copy a matrix from the SØF to a NASTRAN matrix file.

3.6.20.3 Calling Sequence

CALL MTRXI (FILE,NAME,ITEM,0,ITEST)

FILE - GINØ file name of the data block to be written - integer - input.

NAME - Name of substructure - array of dimension two, where each word is 4 characters long - BCD - input.

Item - Item name of the matrix to be copied - allowable names are any legal matrix item - four BCD characters - input.

0 - Dummy parameter to retain compatibility with old calling sequence.

ITEST - Integer - output

= {
1 if ITEM exists on the SØF
2 if ITEM pseudo-exists only on the SØF
3 if ITEM does not exist on the SØF
4 if NAME does not exist
5 if ITEM is not one of the allowable item names
6 if FILE has been purged

The subroutine will copy the matrix data only when ITEST takes the value 1.

3.6.20.4 Method

The GINØ file is opened and the SØF buffer is aligned to match the GINØ buffer. Each block of data is read from the SØF using SØFIØ and written to the GINØ file with WRTBLK. The matrix name is inserted into the first record of the file. The trailer information is extracted from the last SØF block and stored in the FIAT by a call to WRTTRL.

3.6.20.5 Diagnostic Messages

System Fatal Message 6224.

SUBROUTINE DESCRIPTIONS

3.6.21 MTRXØ

3.6.21.1 Entry Point: MTRXØ

3.6.21.2 Purpose

To copy a matrix from a NASTRAN matrix file to the SØF.

3.6.21.3 Calling Sequence

CALL MTRXØ (FILE,NAME,ITEM,O,ITEST)

FILE - GINØ file name of the data block to be read - input.

NAME - Name of substructure - array of dimension two, where each word is 4 characters long - input.

Item - Item name of the matrix to be copied - allowable names are any legal table item - four BCD characters - input.

O - Dummy parameter to retain compatibility with old calling sequence.

ITEST - Integer - output

= {
1 if ITEM already exists on the SØF
3 if ITEM does not exist on the SØF
4 if NAME does not exist
5 if ITEM is not one of the allowable item names
6 if FILE has been purged

The subroutine will copy the matrix data only when ITEST takes the value 3. If the user wishes to rewrite a matrix which already exists on the SØF, he will have to delete the old matrix data through a call to DELETE, and then write the new matrix data.

3.6.21.4 Method

The GINØ file is opened and the SØF buffer is aligned to match the GINØ buffer. Each block of the file is read with the GINØ entry point RDBLK and written to the SØF using SØFIØ. After the last block is read, a test is made to determine if there exists six unused words in this block. If the words exist, the matrix trailer is stored at the end of the block. If insufficient space remains, the matrix trailer is stored in a new block and then written to the SØF.

SUBSTRUCTURE OPERATING FILE (SØF) UTILITIES

If IFILE is less than zero, no data is copied to the SØF, but the new item is added to the SØF for substructure NAME with a block number of 65535 (i.e., $2^{16} - 1$).

3.6.21.5 Diagnostic Messages

Fatal Messages 3008, 3037, and 6223.

SUBROUTINE DESCRIPTIONS

THIS PAGE HAS BEEN LEFT BLANK INTENTIONALLY.

3.6.22 RETBLK

3.6.22.1 Entry Point: RETBLK

3.6.22.2 Purpose

To return a block and all blocks linked to it to the list of free blocks.

3.6.22.3 Calling Sequence

CALL RETBLK (IBL)

IBL - Physical block number of the block which is to be returned - integer - input

3.6.22.4 Method

The number of the superblock, to which block I belongs, is computed, and block I is returned to the list of free blocks belonging to that superblock. All blocks linked to block I are also returned to the list of free blocks of their respective superblocks.

3.6.22.5 Diagnostic Messages

System Fatal Message 6224.

SUBROUTINE DESCRIPTIONS

3.6.23 SETEQ

3.6.23.1 Entry Point: SETEQ

3.6.23.2 Purpose

To set a new substructure equivalent to an old substructure.

3.6.23.3 Calling Sequence

CALL SETEQ (NAME1,NAME2,PREFX,DRY,ITEST,IMØRE,IMAGE,LEN)

NAME1 - Name of old substructure - array of dimension two, where each word is 4 characters long - input.

NAME2 - Name of new substructure - array of dimension two, where each word is 4 characters long - input.

PREFX - The names of all substructures contributing to substructure NAME1 are equivalenced to a new set having the PREFX value appended to the names - single BCD character - input.

DRY - Integer

Input = $\left\{ \begin{array}{l} -1 \text{ if neither NAME2 nor any of the names of the image substructures} \\ \text{should exist before calling SETEQ.} \\ 0 \text{ if NAME2 and all the names of the image substructures should} \\ \text{exist before calling SETEQ.} \\ 1 \text{ if neither NAME2 nor any of the names of the image substructures} \\ \text{should exist before calling SETEQ.} \end{array} \right.$

ITEST - Integer - output.

= $\left\{ \begin{array}{l} 1 \text{ normal return} \\ 4 \text{ if NAME1 does not exist.} \\ 8 \text{ if DRY} = -1 \text{ or } 1 \text{ and NAME2 or one of the image substructures already} \\ \text{exists.} \\ 9 \text{ if DRY} = 0 \text{ and NAME2 or one of the image substructures does not exist.} \end{array} \right.$

SUBSTRUCTURE OPERATING FILE (SOF) UTILITIES

IMØRE - Scratch array where SETEQ will construct a list of all substructures contributing to substructure NAME1.

IMAGE - Scratch array where SETEQ will construct a list of all image substructures that will be created.

LEN - Maximum length of IMØRE and IMAGE arrays that is available to SETEQ - input.

3.6.23.4 Method

1. Make a list of all the substructures contributing to the substructure NAME1.
2. If DRY is equal to -1 or 1, create an image substructure for each substructure in the above list. The names of the image substructures are obtained by adding the prefix PREFIX to the old names. All the image substructures become contributing substructures to NAME2.
3. Build in the MDI the directory of NAME2, and of all image substructures. Items EQSS, LØDS, LØAP, and PLTS of NAME2 will be copied from those of NAME1, after replacing the old name (NAME1) by the new name (NAME2), and inserting the new prefix to the names of all contributing substructures.

3.6.23.5 Diagnostic Messages

User Information Message 6228.

User Warning Message 6236.

User Fatal Message 6223.

System Fatal Message 6224.

SUBROUTINE DESCRIPTIONS

3.6.24 SETLVL

3.6.24.1 Entry Point: SETLVL

3.6.24.2 Purpose

To make a new substructure known to the system.

3.6.24.3 Calling Sequence

CALL SETLVL (NEWNM,NUMB,ØLDNMS,ITEST,BIT)

NEWNM - Name of the new substructure - array of dimension two, where each word is 4 characters long - input.

NUMB - Integer - input and output

= {
 0 if NEWNM is a basic substructure
 1 if NEWNM results from reducing the first substructure in the array
 ØLDNMS
 i where $2 \leq i \leq 7$ if NEWNM results from combining the first i substructures
 in the array ØLDNMS

For output, if ITEST is set to 4, NUMB will be set to the number of substructures in the array ØLDNMS.

ØLDNMS - Array of dimension 14 containing the names of 7 substructures at most, and where each word is 4 characters long - input and output.

If ITEST is set to 4, and NUMB to the number of substructures in ØLDNMS that do not exist, the first NUMB names in ØLDNMS will be set to the names of those substructures that do not exist.

ITEST - Integer - output

= {
 1 normal return
 4 if one or more substructures in ØLDNMS do not exist
 7 if NEWNM already exists
 8 if one of the substructuess in ØLDNMS has already been used in a
 reduction or combination

BIT - Bit position of the type field in the MDI to be set for the new substructure.

SUBSTRUCTURE OPERATING FILE (SØF) UTILITIES

3.6.24.4 Method

Creates an entry for the substructure NEWNM in the DIT, and updates in the MDI the directories of NEWNM and of the substructures in the array ØLDNMS.

SUBROUTINE DESCRIPTIONS

3.6.25 SFETCH

3.6.25.1 Entry Point: SFETCH

3.6.25.2 Purpose

To position the SØF to read or write data.

3.6.25.3 Calling Sequence

CALL SFETCH (NAME(1),ITEM,IRW,ITEST)

NAME - Name of the substructure which is to be read or written - array of dimension two, where each word is 4 characters long - input.

ITEM - Item name of the table to be read or written - allowable names are any legal table item - four BCD characters - input. (Note - matrix items may be processed only with subroutines MTRXØ, MTRXI, or SØFTRL.)

IRW - Integer - input

= { 1 if ITEM is to be read
2 if ITEM is to be written
3 if ITEM is to be checked only

ITEST - Integer

If IRW = 1 then,

output = { 1 if ITEM exists
2 if ITEM pseudo-exists
3 if ITEM does not exist
4 if NAME does not exist
5 if ITEM is an illegal item name

If IRW = 2 then,

input = { 2 if ITEM is to be pseudo-written
3 if ITEM is to be written

output = { 1 if ITEM is already written
4 if NAME does not exist
5 if ITEM is an illegal item name

SUBSTRUCTURE OPERATING FILE (SØF) UTILITIES

3.6.25.4 Method

If ITEM is to be read or checked, ITEST is set to 1, 2 or 3 according to whether ITEM exists, pseudo-exists, or does not exist. If ITEM is to be checked only, SFETCH now returns. If ITEM exists, the first block on which ITEM is written is read in the buffer BUF starting at location $(IØ + 1)$ and extending to location $(IØ + BLKSIZ)$. The pointer into the buffer is also initialized.

If ITEM is to be written or pseudo-written, the MDI is examined and ITEST set to 1 if ITEM has already been written. Otherwise, if ITEM is to be pseudo-written, the block number for ITEM is set to 65535 (i.e., $2^{16} - 1$), and if it is to be written, the pointer in the array BUF is initialized.

SUBROUTINE DESCRIPTIONS

3.6.26 SJUMP

3.6.26.1 Entry Point: SJUMP

3.6.26.2 Purpose

To jump over groups within an item, when in the read mode.

3.6.26.3 Calling Sequence

CALL SJUMP (N)

N - Number of groups to skip over - integer - input

N will be set to:

-1 if the end-of-item was reached before skipping over N groups

-2 if in the write mode

3.6.26.4 Method

The SØF is positioned at the beginning of the group following the Nth end of group marker that has been skipped.

3.6.26.5 Diagnostic Messages

System Fatal Message 6224.

SUBSTRUCTURE OPERATING FILE (SØF) UTILITIES

3.6.27 SØFCLS

3.6.27.1 Entry Point: SØFCLS

3.6.27.2 Purpose

To save, at the termination of a module, all the in-core buffers and common blocks by writing them out on the direct access storage device.

3.6.27.3 Calling Sequence

CALL SØFCLS

3.6.27.4 Method

Each of the in-core blocks of the DIT, MDI and array NXT that has been updated is written on the direct access storage device. The data contained in the /SØF/ and /SYS/ common blocks, along with the SØF item structure in the ITEMMDT table are written in the first block of each physical SØF file.

If the SØF is not open, no action is taken.

SUBROUTINE DESCRIPTIONS

3.6.28 SØFINT

3.6.28.1 Entry Point: SØFINT

3.6.28.2 Purpose

To initialize and update the common blocks used by the SØF utility subroutines. The subroutine is called only once by SØFØPN at the beginning of every run using the SØF utility subroutines.

3.6.28.3 Calling Sequence

CALL SØFINT (IB1,IB2,NUMB,IBL1)

CØMMØN/SØFCØM/NFILES,FILNAM(10),FILSIZ(10),STATUS PSSWRD(2)

NFILES - User-specified number of SØF files where the maximum number of files is 10 -
integer - input

FILNAM - NASTRAN file names assigned to the SØF files - BCD - input

FILSIZ - Size of the SØF files expressed in terms of the number of blocks in each file.
The size of a block is calculated by GINØ and stored in /GINØX/ - integer -
input.

STATUS - Integer - input.

= $\begin{cases} 0 & \text{if the SØF is empty} \\ 1 & \text{if the SØF is not empty} \end{cases}$

PSSWRD - Password - BCD - input

IB1,IB2 - Two arrays where the dimension of each array is equal to the contents of the
first word of /SYSTEM/.

NUMB - Number of blocks to be added to the last superblock of the last SØF file.
Integer - output.

IBL1 - Block number of the first block to be added to the last superblock of the last
SØF file. Integer - output.

3.6.28.4 Method

a. If STATUS = 0, the following operations take place:

1. The SØF common blocks are initialized.
2. The array NXT, the DIT and the MDI are initialized and written out on the SØF.

SUBSTRUCTURE OPERATING FILE (SØF) UTILITIES

3. The password, the SØF common blocks and the SØF item structure are written on the first block of each SØF file together with the sequence number of that file.
- b. If STATUS = 1, the following operations take place:
 1. The password as well as the file sequence number are checked on each of the SØF files that were available in the previous run and are still available in the current run. If any of the old passwords is not identical to the new password, or if any of the SØF files is out of sequence, a fatal error is flagged.
 2. The buffer size is checked and not allowed to vary from that in the previous run.
 3. The number and sizes of the SØF files are checked and a fatal error is issued if the following rules are not respected:
 - i. The number of SØF files may always be incremental, but the new files should always follow the old files in the sequence.
 - ii. Only the size of the last file in previous run - "last old file" - can be increased.
 - iii. Any number of old files can be deleted from the end of the list of old files as long as no data was written on them.
 - iv. Only the size of the last old file can be reduced under the condition that no data was written on the portion of the file which is to be deleted.
 4. The item structure is retrieved from the SØF and stored in the ITEMDET table. If no item structure is present, the level 16.0 structure is used to keep compatibility with old systems.
 5. The SØF common blocks are updated, and the password, SØF common blocks and the SØF item structure are written on the first block of each SØF file available to the current run together with the sequence number of that file.

SUBROUTINE DESCRIPTIONS

3.6.29 SØFIØ

3.6.29.1 Entry Point: SØFIØ

3.6.29.2 Purpose

To perform the I/Ø operations between core and the SØF files.

3.6.29.3 Calling Sequence

CALL SØFIØ (IRW,IBLKNM,IBUFF)

CØMMØN/SØFCØM/NFILES,FILNAM(10),FILSIZ(10)

IRW - Integer - input

= $\begin{cases} 1 & \text{if a read operation is requested} \\ 2 & \text{if a write operation is requested} \end{cases}$

IBLKNM - Physical block number of the block of data which is to be read or written - integer - input.

IBUFF - Array which contains the data to be written on the SØF or in which data read from the SØF is to be stored. The dimension of the array is equal to the contents of the first word of /SYSTEM/. The I/Ø data is stored at starting location IBUFF(4).

NFILES - Number of SØF files - integer - input.

FILNAM - NASTRAN file names assigned to the SØF files - BCD - input.

FILSIZ - Size of the SØF files expressed in terms of the number of blocks in each file. the size of a block is 3 words smaller than the contents of the first word of /SYSTEM/ - integer - input.

3.6.29.4 Method

The SØF file number and the block number within that file are computed from the integer IBLKNM. The requested operation is then performed.

SØFIØ is a machine dependent subroutine.

3.6.29.5 Diagnostic Messages

System Fatal Messages 6225-6227.

SUBSTRUCTURE OPERATING FILE (SØF) UTILITIES

3.6.30 SØFØPN

3.6.30.1 Entry Point: SØFØPN

3.6.30.2 Purpose

To compute the addresses of the three in-core buffers, and read into core the common blocks /SØF/ and /SYS/. The subroutine should be called at the beginning of each module using the SØF utility subroutines.

3.6.30.3 Calling Sequence

CALL SØFØPN (B1,B2,B3)

CØMMØN/SØFCØM/NFILES,FILNAM(10),FILSIZ(10),STATUS,PSSWRD(2),FIRST

FIRST - Logical - input.

= { .TRUE. if SØFØPN is being called for the first time in the current run.
.FALSE. if SØFØPN has been previously called.

B1,B2, - Three arrays where the dimension of each array is equal to the contents of the
B3 first word of /SYSTEM/.

3.6.30.4 Method

The addresses of the three in-core buffers are computed and if FIRST is equal to .TRUE. the SØFINT is called. The common blocks /SØF/ and /SYS/ are then read into core.

3.6.30.5 Diagnostic Messages

User Fatal Message 6222.

SUBROUTINE DESCRIPTIONS

3.6.31 SØFSIZ

3.6.31.1 Entry Point: SØFSIZ

3.6.31.2 Purpose

To indicate how many available words remain on the SØF.

3.6.31.3 Calling Sequence

K = SØFSIZ(DUM) Integer function

DUM - Dummy Parameter

3.6.31.4 Method

SØFSIZ obtains the number of available blocks and the number of words per block from common block /SYS/. The product of these two terms is the number of available words on the SØF.

3.6.32 SUREAD

3.6.32.1 Entry Point: SUREAD

3.6.32.2 Purpose

To read a certain number of data words belonging to an item into a given array. The substructure and item names must have been specified through an earlier call to the subroutine SFETCH.

3.6.32.3 Calling Sequence

CALL SUREAD (IA,ND,NØUT,ITEST)

IA - Address of the array into which data is to be read. The array should be large enough to hold all the desired data.

ND - Integer - input

$$= \begin{cases} 1 & \text{where } i \geq 0 \text{ if } i \text{ words are to be read} \\ -1 & \text{if a read until end-of-group is requested} \\ -2 & \text{if a read until end-of-item is requested} \end{cases}$$

NØUT - Number of words that have been read - integer - output

ITEST - Integer - output

$$= \begin{cases} 1 & \text{normal return} \\ 2 & \text{if end-of-group has been encountered} \\ 3 & \text{if end-of-item has been encountered} \\ 4 & \text{if not in read mode} \end{cases}$$

3.6.32.4 Method

The pointer into the in-core buffer BUF is updated as words are read from BUF into the array IA. If the end of the in-core block is reached before the desired number of blocks has been read, the block next to the in-core block is copied into core, and the read operation continues until either the desired number of words has been read, or an end-of-group is encountered, or an end-of-item is encountered.

3.6.32.5 Diagnostic Messages

System Fatal Message 6224.

SUBROUTINE DESCRIPTIONS

3.6.33 SUWRT

3.6.33.1 Entry Point: SUWRT

3.6.33.2 Purpose

To copy a certain number of data words belonging to an item from a given array onto the random access storage device. The substructure and item names must have been specified through an earlier call to the subroutine SFETCH.

3.6.33.3 Calling Sequence

CALL SUWRT (IA,NWØRDS,ITEST)

IA - Address of the array from which data is to be copied

NWØRDS - Number of words that should be copied from the array IA into the random access storage device - integer - input

ITEST - Integer - input

= { 1 if there is more data to come
2 if an end-of-group should be written at the end of the copied data
3 if an end-of-item should be written at the end of the copied data

output = 4 if in the read mode

3.6.33.4 Method

The pointer into the in-core buffer BUF is updated as data is copied from IA into BUF. If the end of the in-core I/O block is reached before the desired number of words has been copied, the full block is written out on the random access storage device, a new free block is called for, and the array NXT is updated to indicate that the old block is followed by the new. The write operation continues after that and is terminated as specified by the variable ITEST. The directory of the item which is written is, however, not updated until an end-of-item is written. Thus, if the NASTRAN programmer forgets to specify an end-of-item at the end of his data, the data will be lost.

3.6.33.5 Diagnostic Messages

User Fatal Message 6223.

SUBSTRUCTURE OPERATING FILE (SØF) UTILITIES

3.6.34 Common Blocks Used by the SØF Utility Subroutines

1. CØMMØN/SØF/ DIT,DITPBN,DITLBN,DITSIZ,DITNSB,DITBL,
IØ,IØPBN,IØLBN,IØMØDE,IØPTR,IØSIND,IØITCD,IØBLK,
MDI,MDIPBN,MDILBN,MDIBL,
NXT,NXTPBN,NXTLBN,NXTTSZ,NXTFSZ(10),NXTCUR,
DITUP,MDIUP,NXTUP,NXTRST

- DIT - Pointer to an address in open core where the address is relative to the beginning of the common blank block. The in-core block of the DIT is stored in an array, starting at location (DIT + 1) and extending to location (DIT + BLKSIZ).
- DITPBN - Physical block number of the block of the DIT which is stored in core. If no block of the DIT is in core DITPBN is set to zero.
- DITLBN - Logical block number of the block of the DIT which is stored in core. If no block of the DIT is in core, DITLBN is set to zero.
- DITSIZ - Total number of words used by the DIT.
- DITNSB - Number of substructure names stored in the DIT.
- DITBL - Physical block number of the first block of the DIT. The value of DITBL is set in the subroutine SØFINT.
- IØ - Pointer to an address in open core, where the address is relative to the beginning of the common blank block. The input/output buffer extends from location (IØ + 1) to location (IØ + BLKSIZ).
- IØPBN - Physical block number of the data block being read or written.
- IØLBN - Logical block number of the data block being read or written.
- IØMØDE - Mode of current input/output operation. The variable takes one of the following values:
- 0 in the idle mode
 - 1 in the read mode
 - 2 in the write mode
- IØPTR - Pointer to an address in the input/output buffer. The address is relative to the beginning of the blank common block.

SUBROUTINE DESCRIPTIONS

- IØSIND - Index of the substructure whose data is being read or written.
- IØITCD - Code of the item whose data is being read or written. The item code is obtained by calling the function ITCØDE.
- IØBLK - In the case of a write operation, IØBLK contains the physical block number of the first data block in the chain of blocks that are written out.
- MDI - Pointer to an address in core where the address is relative to the beginning of the blank common block. The in-core block of the MDI is stored in an array starting at location (MDI + 1) and extending to location (MDI + BLKSIZ).
- MDIPBN - Physical block number of the block of the MDI which is stored in core. If no block of the MDI is stored in core, MDIPBN is set to zero.
- MDILBN - Logical block number of the block of the MDI which is stored in core. If no block of the MDI is in core, MDILBN is set to zero.
- MDIBL - Physical block number of the first block of the MDI. The value of MDIBL is set in the subroutine SØFINT.
- NXT - Pointer to an address in open core where the address is relative to the beginning of the blank common block. The value of NXT is the same as that of DIT since the array NXT and the DIT share the same in-core buffer. Only one of them is in core at any time. The in-core block of NXT also extends from location (NXT + 1) to location (NXT + BLKSIZ).
- NXTPBN - Physical block number of the block of the array NXT which is stored in core. If no block of NXT is in core, NXTPBN is set to zero.
- NXTLBN - Logical block number of the block of the array NXT which is stored in core. If no block of NXT is in core, NXTLBN is set to zero.
- NXTTSZ - Total number of superblocks that have been used so far.
- NXTFSZ - Array containing the number of superblocks that are on each one of the SØF files.
- NXTCUR - Number of the superblock which is currently being used.
- DITUP - Logical variable which takes one of the following values:
 .FALSE. if the in-core block of the DIT has not been updated
 .TRUE. if the in-core block of the DIT has been updated

SUBSTRUCTURE OPERATING FILE (SØF) UTILITIES

MDIUP - Logical variable which takes one of the following values:

- .FALSE. if the in-core block of the MDI has not been updated
- .TRUE. if the in-core block of the MDI has been updated

NXTUP - Logical variable which takes one of the following values:

- .FALSE. if the in-core block of the array NXT has not been updated
- .TRUE. if the in-core block of the array NXT has been updated

NXTRST - Logical variable which takes one of the following values:

- .FALSE. if no search for a free block has been made through already existing superblocks
- .TRUE. if a search for a free block is being carried through already existing superblocks

2. COMMON/SØFCØM/NFILES,FILNAM(10),FILSIZ(10),STATUS,PSSWRD(2),FIRST,ØPNSØF,ASØFCB

NFILES - Number of files allocated to the SØF. The maximum number of files that can be allocated is ten.

FILNAM - Array containing the names of the SØF files. Each name is four characters long.

FILSIZ - Array containing the sizes of the SØF files. The size is expressed in number of blocks where the number of words contained in a block is four words less than the content of the first word of /SYSTEM/. The size of each SØF file must be an even number of blocks ≥ 4 .

STATUS - Integer variable which takes one of the following values:

- 0 if the SØF is empty
- 1 if the SØF is not empty

PSSWRD - Array containing a password for the SØF. A substructuring problem must use the same password for all runs. The password is two words of four characters each.

FIRST - Logical variable which takes the value .TRUE. at the beginning of each substructuring run thus initiating a call from the subroutine SØFØPN to the subroutine SØFINT. FIRST is then set to .FALSE. in SØFINT.

ØPNSØF - Logical variable which is set to .TRUE. when SØFØPN is called and to .FALSE. when SØFCLS is called.

SUBROUTINE DESCRIPTIONS

ASØFCB - Address of control blocks for direct access SØF input and output on the IBM 360/370 computer.

3. CØMMØN/SYS/BLKSIZ,DIRSIZ,SUPSIZ,AVBLKS,HIBLK,IFIRST

BLKSIZ - Block size. The size of a block is three words smaller than the content of the first word of /SYSTEM/.

DIRSIZ - Size of a substructure's directory in the MDI. The value of DIRSIZ is set in the sub-routine SØFINT.

SUPSIZ - Superblock size. The size of a superblock is equal to $2 * (BLKSIZ - 1)$.

AVBLKS - Number of available free blocks on the SØF.

HIBLK - Block number of the highest numbered block written.

IFIRST - Index in the MDI of the word containing the information for the first item
(currently = 2)

SUBSTRUCTURE OPERATING FILE (SØF) UTILITIES

3.6.35 SMSG (Substructure Messages)

3.6.35.1 Entry Point: SMSG

3.6.35.2 Purpose

Message writer for substructure diagnostics.

3.6.35.3 Calling Sequence

CALL SMSG (N,ITEM,NAME)

N - Message number. N > 0 indicates warning only. N < 0 indicates fatal message.

Integer - input.

ITEM - Item name. BCD, input.

NAME - Substructure name. Two-word BCD array, input.

3.6.35.4 Method

The message number generated is $6100 + \text{IABS}(N)$.

For fatal messages, subroutine SØFCLS is called to save any SØF updates. Subroutine MESSAGE is then called to terminate execution.

SUBROUTINE DESCRIPTIONS

3.6.36.1 Entry Point: SØFTRL

3.6.36.2 Purpose

To obtain the matrix control block of a matrix stored on the SØF.

3.6.36.3 Calling Sequence

CALL SØFTRL (NAME,ITEM,MCB)

NAME - Name of substructure whose matrix item is to be examined - array of dimension two words where each word is 4 characters long - input.

ITEM - Name of matrix item - allowable names are: KMTX, MMTX, PVEC, PAPP, PØVE, UPRT, HØRG, UVEC, QVEC, LMTX - four BCD characters - input.

MCB - 7-word matrix control block. Item status is returned in MCB(1).

	1 if ITEM exists - matrix trailer is returned in words 2 through 5
	2 if ITEM pseudo-exists
MCB(1) =	3 if ITEM does not exist
	4 if NAME does not exist
	5 if ITEM is an illegal item name

3.6.36.4 Method

The existence of NAME and ITEM on the SØF is checked. The chain of blocks belonging to the item is followed until the last block is located. This block is then read and the trailer is extracted from the last six words.

SUBSTRUCTURE OPERATING FILE (SOF) UTILITIES

3.6.37.1 Entry Point: ITTYPE

3.6.37.2 Purpose

To return the type, table, or matrix of each item.

3.6.37.3 Calling Sequence

K = ITTYPE (ITEM) Integer function

K - Integer - output

0 if table item

=

1 if matrix item

ITEM - any of the legal item names - four BCD characters - input.

The function will return the following:

0 - table item

1 - matrix item

-1 - illegal item name

3.6.37.4 Method

The ITEM table is searched for the occurrence of the requested item. If no match is found, a negative number is returned. If a match is found the item TYPE field is returned.

SUBROUTINE DESCRIPTIONS

3.6.38 FNDLVL

3.6.38.1 Entry Point: FNDLVL

3.6.38.2 Purpose

To find the name of a lower level substructure to a given substructure.

3.6.38.3 Calling Sequence

CALL FNDLVL (NAME,NEWNM)

NAME - Name of the substructure whose lower level substructure's name is desired - array of dimension two words where each word is 4 characters long - BCD - input.

NEWNM - Name of a lower level substructure to NAME. If NAME does not have a lower level substructure, NAME will be returned in NEWNM. If NAME is not known to system, blanks will be returned in NEWNM. NEWNM is an array of dimension two words, where each word is 4 characters long - BCD - output.

3.6.38.4 Method

A search for NAME is carried in the DIT. If NAME is not found, blanks are returned in NEWNM. If NAME is found, its directory is fetched from the MDI. If NAME does not have a lower level substructure, NAME will be returned in NEWNM, otherwise the name of the lower level substructure will be returned in NEWNM. Note that only one of the possible lower level substructures will be returned. To get all of them the CS loop of that substructure must be traversed.

4. MODULE FUNCTIONAL DESCRIPTIONS

4.1 GENERAL COMMENTS AND INDEXES

The NASTRAN modules (a module is a logical group of subroutines) documented in this section have been classified into 7 categories: 1) Executive Preface modules, 2) Executive modules, 3) Executive DMAP instructions, 4) Executive DMAP modules, 5) functional modules, 6) output modules, and 7) matrix modules.

Executive Preface modules are those which are executed prior to the execution of the first modules in a DMAP sequence. They consist of: 1) XCSA (Executive Control Section Analysis), which processes the NASTRAN Executive Control Deck; 2) IFP1 (Input File Processor, Part 1), which processes the NASTRAN Case Control Deck; 3) XSORT (Executive Bulk Data Card Sort), which sorts the NASTRAN Bulk Data Deck; 4) IFP (Input File Processor), which processes the sorted Bulk Data Deck; 5) IFP3, IFP4 and IFP5 (Input File Processor 3, 4 and 5), which process bulk data cards unique to an axisymmetric conical shell, hydroelastic, or acoustic problem; 6) XGPI (Executive General Problem Initialization), the heart of the Preface, which a) translates (compiles) a DMAP sequence into an internal form, the ØSCAR, - see Section 2.4.2.1, and b) for problem restarts, initializes data blocks and labeled common blocks; and 7) UMFEDIT (User Master File Editor), which creates and manipulates User Files.

The only module classified as an Executive module, per se, is XSFA (Executive Segment File Allocator), which is the "administrative manager" of files for NASTRAN.

Executive DMAP instructions documented in this section are REPT, JUMP, CØND, EXIT, and END. The DMAP instructions BEGIN, FILE, LABEL, PRECHK, and XDMAP are not allocated a separate section and, therefore, brief descriptions follow.

1. The BEGIN and XDMAP instructions are declarative DMAP instructions which denote the beginning of a DMAP sequence. They are analogous to a computer operating system control card which calls a system compiler. The BEGIN card is used in cases where all DMAP compiler default options are elected.

2. The FILE DMAP instruction is a declarative DMAP instruction which alters the normal attributes of a data block in an ØSCAR entry (see an explanation of the attributes AP, LTU, TP, NTU of a data block in the Data Section Format for functional modules section in the description of the ØSCAR, section 2.4.2.1). These attributes of a data block are used by the Executive Segment File Allocator (XSFA) module in performing its task.

MODULE FUNCTIONAL DESCRIPTIONS

3. The LABEL DMAP instruction is used to label a location in a DMAP sequence so that the location may be referenced by the DMAP instructions JUMP, COND, and REPT.

4. The PRECHK DMAP instruction is a declarative DMAP instruction allowing the user to make a single, or limited number of statements which will generate CHKPNT instructions for the entire DMAP sequence automatically.

A more detailed description of these five Executive DMAP modules can be found in Section 5 of the User's Manual.

Executive DMAP modules consist of CHKPNT, SAVE, PURGE, EQUIV, PARAM and SETVAL. In addition to the descriptions in this section, the reader is referred to section 5 of the User's Manual, where further explanations of the uses of EQUIV, PURGE and CHKPNT can be found.

Functional modules comprise the bulk of the descriptions in this section. Functional modules perform the actual structural problem solution. The reader is referred to section 5 of the User's Manual for a) general comments on DMAP rules and b) the syntactical rules of the DMAP calling sequences referring to functional modules.

Output modules are those whose entire output is directed a) to the system output file and/or b) to a tape which will drive a plotting device.

Matrix modules are those which, although no different operationally from functional modules, are most likely to be used by the program user who wishes to take advantage of the Direct Matrix Abstraction capabilities of NASTRAN; therefore, they are separately categorized.

4.1.1 Use of Module Functional Descriptions

Each module documented by means of a Module Functional Description (MFD) has been assigned an integer *i*, and its MFD is documented in section 4.1. For functional modules, a consistent numbering scheme has been followed, wherever possible, in the MFD's. For a functional module whose assigned integer is *i*, then,

- 4.1 Title
- 4.1.1 Entry Point
- 4.1.2 Purpose
- 4.1.3 DMAP Calling Sequence
- 4.1.4 Input Data Blocks
- 4.1.5 Output Data Blocks
- 4.1.6 Parameters

- 4.i.7 Method
- 4.i.8 Subroutines
- 4.i.9 Design Requirements
- 4.i.10 Diagnostic Messages

comprises this numbering scheme. The title: a) classifies the module into one of the seven categories defined in the first paragraph of section 4.1; b) defines the module name, which, if it is a DMAP module (one which is called by a DMAP instruction), is the name by which it must be called in the DMAP calling sequence; and c) defines, parenthetically, the phrase from which the name was derived. Comments on the remaining sections follow.

1. Entry Point

This section defines the entry point of the module. A module's entry point usually agrees with the module name, but there are exceptions. For example, the READ (Real Eigenvalue Analysis Displacement) module has the entry point REIG (the entry point READ is a subroutine in the GINØ collection of routines).

2. Purpose

A brief description of the purpose of the module is given. The casual or first-time reader will perhaps go no further than read the purpose.

3. DMAP Calling Sequence

The DMAP calling sequence as it appears in a Rigid Format is given (see section 3 of the User's Manual for a detailed description of the Rigid Formats in NASTRAN). DMAP calling sequences for Executive DMAP instructions and for Executive DMAP modules follow no fixed format. Refer to the individual Module Functional Descriptions for details on their DMAP calling sequences. Functional modules, which are always "called" via a DMAP calling sequence, do have a fixed format. Consider the following DMAP calling sequence for functional module SMA2, which generates the mass matrix, $[M_{gg}]$, and the damping matrix, $[B_{gg}]$:

```
SMA2  CSTM,MPT,ECPT,GPCT,DIT/MGG,BGG/V,Y,WTMASS = 1.0/V,N,NØMGG/V,N,NØBGG/V,Y,
      CØUPBAR = -1 $
```

SMA2 is the module name. The name of a module must begin with an alphabetic character

followed by up to seven additional alphanumeric characters. Following the name is a blank field. Following this blank field is the list {CSTM,MPT,ECPT,GPCT,DIT} of data blocks input to the module. The list is terminated by a slash (/). Each item in this list is separated by a comma. Note that the number of commas for this list is one less than the number of input data blocks. The second slash terminates the list {MGG,BGG} of data blocks output from the module. The rule for naming input and output data blocks is the same as for module names. Each subsequent slash terminates a parameter field. Each parameter field contains three parts separated by commas. The first part is either the letter "V" or the letter "C", defining the parameter as a variable or as a constant respectively. The second part is either the letter "Y" or the letter "N". "Y" implies "yes" the value of the parameter may be specified on a PARAM bulk data card, and "N" implies "no" the value of the parameter may not be specified on a PARAM bulk data card. The third part may be either: (a) the name of the parameter, (b) the value of the parameter, or (c) the name and the value of the parameter. A variable parameter must have a name, hence a variable parameter may not be specified only by its value. The rule governing the names of parameters is the same as that for module names. The value of a parameter may be complex double precision, complex single precision, double precision, real, integer or BCD. In the example given, the name of the first parameter is WTMASS, and its initial value (which can be overridden by a value on a PARAM card because of "Y" prior to the name) is 1.0. Note that the slash terminating the last parameter field is omitted. Although one can terminate the last parameter field with a slash, this final slash is usually omitted. A dollar sign, "\$", terminates a DMAP statement.

4. Input Data Blocks

A short description of each of the module's input data blocks is given along with notes explaining what the module's design requires about the status (purged or not purged) of the data blocks. Detailed data block descriptions are found in section 2 of the Programmer's Manual.

5. Output Data Blocks

A short description of each of the module's output data blocks and an explanation of the action taken when an output data block has been pre-purged are given in this section. An output data block is said to be pre-purged if the data block has been

explicitly purged in a previous PURGE DMAP instruction, or if the data block does not appear in the DMAP calling sequence for the module.

6. Parameters

The order of DMAP parameters in a DMAP calling sequence is the same as the order of the FORTRAN variables corresponding to the parameters in blank common at module execution time. Each variable DMAP parameter is defined as whether a) it is input data into, or output data from, the module, or both (e.g., a DMAP loop counter which is incremented within the module); b) the type of the parameter: integer, real, double precision, complex single precision, complex double precision, or BCD; and c) the default value of the parameter as defined either i) in the Module Properties List (MPL) Executive table, ii) by means of a PARAM or SETVAL DMAP instruction, or iii) by means of the DMAP statement itself. An example of the third type of default value is

```
MODULEA  A,B,C/D,E/V,N,UVW/V,Y,XYZ=-1 $.
```

The parameter XYZ is set to -1 by the above statement. For further information on DMAP parameters see paragraph 3 above, section 2.4.2.2 in the Programmer's Manual and section 5 of the User's Manual.

7. Method

A discussion of the method used by the module writer to achieve the purpose of the module is given in this section.

8. Subroutines

The subroutines which comprise the module are described in this section. However, not all subroutines capable of being called by a module are listed here. Utility routines and matrix routines that are in the root segment are not listed in this section. These include: MAPFNS, all the GINØ routines (ØPEN, WRITE, CLØSE, READ, FWDREC, BCKREC, REWIND, EØF, SKPFIL, XGINØ, GINØ, ØPNCØR), FREAD, GØPEN, WRTTRL, FNAME, CLSTAB, PRELØC, PEXIT, TMTØGØ, MESSAGE, and the matrix packing and unpacking routines (BLDPK, PACK, INTPK, UNPACK). Descriptions for these routines are found in section 3.

MODULE FUNCTIONAL DESCRIPTIONS

9. Design Requirements

Design requirements peculiar to the module are presented.

10. Diagnostic Messages

Diagnostic messages unique to the module are given in this section. A detailed list of NASTRAN diagnostic messages can be found in section 6 of the User's Manual.

GENERAL COMMENTS AND INDEXES

4.1.2 Alphabetical Index of Module Functional Descriptions

<u>Section Number</u>	<u>Module Name</u>	<u>Section Number</u>	<u>Module Name</u>
4.78	ADD	4.32	GPSP
4.96	ADD5	4.29	GPWG
4.114	AMG	4.21	GP1
4.115	AMP	4.22	GP2
4.112	APD	4.25	GP3
4.127	ASDMAP	4.31	GP4
***	BEGIN	4.5	IFP*
4.90	BMG	4.3	IFP1
		4.6	IFP3*
4.56	CASE	4.89	IFP4*
4.59	CEAD	4.91	IFP5*
4.10	CHKPNT	4.97	INPUT
4.128	CØMB1	4.98	INPUTT1
4.129	CØMB2	4.99	INPUTT2
4.13	CØND	**	INPUTT3
4.148	CØPY	**	INPUTT4
4.110	CYCT1		
4.111	CYCT2	4.12	JUMP
**	DDR	***	LABEL
4.141	DDRMM		
4.67	DDR1	4.72	MATGPR
4.68	DDR2	4.71	MATPRN
4.81	DECØMP	4.73	MATPRT
4.143	DIAGØNAL	4.33	MCE1
4.47	DPD	4.34	MCE2
4.121	DSCHK	4.84	MERGE
4.49	DSMG1	**	MØDA
4.51	DSMG2	4.126	MØDACC
**	DUMMØD1	**	MØDB
**	DUMMØD2	**	MØDC
**	DUMMØD3	4.79	MPYAD
**	DUMMØD4	4.57	MTRXIN
4.123	EMA	4.70	ØFP
4.124	EMG	4.120	ØPTPR1
4.18	END	4.142	ØPTPR2
4.17	EQUIV	**	ØUTPUT
4.130	EXIØ	4.100	ØUTPUT1
4.14	EXIT	4.101	ØUTPUT2
		4.102	ØUTPUT3
4.116	FA1	**	ØUTPUT4
4.117	FA2		
4.82	FBS	4.19	PARAM
***	FILE	4.118	PARAML
4.61	FRRD	4.119	PARAMR
		4.83	PARTN
4.113	GI	**	PARTVEC
4.58	GKAD	4.52	PLA1
4.66	GKAM	4.53	PLA2
4.109	GPCYC	4.54	PLA3
4.146	GPFDR	4.55	PLA4
		4.24	PLØT

*Executive System Internal Module

**Dummy Module

***Executive System Instruction (No Module Functional Descriptions)

GENERAL COMMENTS AND INDEXES

Alphabetical Index of Module Functional Descriptions (Continued)

<u>Section Number</u>	<u>Module Name</u>	<u>Section Number</u>	<u>Module Name</u>
4.139	PLTMRG	4.122	TABPCH
4.23	PLTSET	4.103	TABPRT
4.92	PLTTRAN	4.75	TABPT
4.76	PRTMSG	4.26	TA1
4.77	PRTPARM	4.140	TIMETEST
4.16	PURGE	4.65	TRD
4.145	PVECIi	4.107	TRHT
		4.106	TRLG
		4.85	TRNSP
4.64	RANDØM		
4.37	RMBG1	4.94	UMERGE
4.38	RMBG2	4.8	UMFEDIT*
4.39	RMBG3	4.93	UPARTN
4.40	RMBG4		
4.131	RCØVR		
4.132	RCØVR3	4.60	VDR
4.48	READ	4.95	VEC
4.133	REDUCE		
4.11	REPT	4.2	XCSA*
4.104	RMG	4.7	XGPI*
		4.9	XSFA*
4.15	SAVE	4.4	XSØRT*
4.144	SCALAR	4.69	XYPLØT
4.35	SCE1	**	XYPRNPLT
4.108	SDRHT	4.63	XYTRAN
4.45	SDR1		
4.46	SDR2		
4.62	SDR3		
4.74	SEEMAT		
4.20	SETVAL		
4.134	SGEN		
4.27	SMA1		
4.28	SMA2		
4.30	SMA3		
4.86	SMPYAD		
4.36	SMP1		
4.50	SMP2		
4.135	SØFI		
4.136	SØFØ		
4.137	SØFUT		
4.80	SØLVE		
4.105	SSGHT		
4.41	SSG1		
4.42	SSG2		
4.43	SSG3		
4.44	SSG4		
4.138	SUBPH1		
4.147	SWITCH		

*Executive System Internal Module

**Dummy Module

***Executive System Instruction (No Module Functional Description)

MODULE FUNCTIONAL DESCRIPTIONS

4.1.3 Alphabetical Index of Entry Points in Module Functional Descriptions

<u>Section Number</u>	<u>Entry Point</u>	<u>Module Name</u>	<u>Page Number</u>
4.46.8	AI	SDR2	4.46-7
4.59.8.25	ALLMAT	CEAD	4.59-18
4.46.8	AMATRX	SDR2	4.46-7
4.114.1	AMG	AMG	4.114-1
4.115.1	AMP	AMP	4.115-1
4.115.8.1	AMPA	AMP	4.115-8
4.115.8.2	AMPB	AMP	4.115-9
4.115.8.3	AMPB1	AMP	4.115-9
4.115.8.4	AMPB2	AMP	4.115-10
4.115.8.5	AMPC	AMP	4.115-10
4.115.8.6	AMPC1	AMP	4.115-10
4.115.8.7	AMPC2	AMP	4.115-12
4.115.8.8	AMPD	AMP	4.115-12
4.112.1	APD	APD	4.112-1
4.112.8.2	APDF	APD	4.112-3
4.112.8.1	APD1	APD	4.112-3
4.48.8.25	ARRM	READ	4.48-18
4.127.1	ASDMAP	ASDMAP	4.127-1
4.127.8.1	ASPRØ	ASDMAP	4.127-6
4.7.5.13	AUTØCK	XGPI	4.7-6
4.7.5.14	AUTØSV	XGPI	4.7-7
4.41.11.35	BAR	SSG1	4.41-27
4.41.11.21	BASGLB	SSG1	4.41-22
4.128.8.4	BDAT01	CØMB1	4.128-11
4.128.8.5	BDAT02	CØMB1	4.128-12
4.128.8.8	BDAT03	CØMB1	4.128-14
4.128.8.10	BDAT04	CØMB1	4.128-20
4.128.8.6	BDAT05	CØMB1	4.128-12
4.128.8.7	BDAT06	CØMB1	4.128-13

GENERAL COMMENTS AND INDEXES

<u>Section Number</u>	<u>Entry Point</u>	<u>Module Name</u>	<u>Page Number</u>
4.46.8	BINT	SDR2	4.46-7
4.128.8.25	BITPAT	CØMB1	4.128-41
4.90.1	BMG	BMG	4.90-1
4.90.8	BMGTNS	BMG	4.90-7
4.28.8.16	BVISC	SMA2	4.28-8
4.56.1	CASE	CASE	4.56-1
4.59.8.14	CDETM	CEAD	4.59-12
4.59.8.15	CDETM2	CEAD	4.59-12
4.59.8.18	CDETM3	CEAD	4.59-14
4.59.8.9	CDIFBS	CEAD	4.59-8
4.59.8.19	CDIVID	CEAD	4.59-14
4.59.8.17	CDTFBS	CEAD	4.59-13
4.59.1	CEAD	CEAD	4.59-1
4.59.8.1	CEAD1A	CEAD	4.59-3
4.59.8.23	CFACTR	CEAD	4.59-17
4.59.8.24	CFBSØR	CEAD	4.59-17
4.59.8.8	CINFBS	CEAD	4.59-8
4.59.8.2	CINVPR	CEAD	4.59-3
4.59.8.3	CINVP1	CEAD	4.59-4
4.59.8.4	CINVP2	CEAD	4.59-4
4.59.8.5	CINVP3	CEAD	4.59-6
4.128.8.13	CMAUTØ	CØMB1	4.128-24
4.128.8.2	CMCASE	CØMB1	4.128-9
4.128.8.12	CMCKCD	CØMB1	4.128-23
4.128.8.29	CMCKDF	CØMB1	4.128-45
4.128.8.16	CMCØMB	CØMB1	4.128-29
4.128.8.11	CMCØNT	CØMB1	4.128-21
4.128.8.17	CMDISC	CØMB1	4.128-32
4.128.8.19	CMHGEN	CØMB1	4.128-35

MODULE FUNCTIONAL DESCRIPTIONS

<u>Section Number</u>	<u>Entry Point</u>	<u>Module Name</u>	<u>Page Number</u>
4.128.8.15	CMMCØN	CØMB1	4.128-29
4.128.8.14	CMRELS	CØMB1	4.128-28
4.128.8.9	CMSFIL	CØMB1	4.128-14
4.128.8.18	CMSØFØ	CØMB1	4.128-34
4.59.8.10	CMTIMU	CEAD	4.59-9
4.128.8.3	CMTØC	CØMB1	4.128-10
4.59.8.6	CNØRM	CEAD	4.59-7
4.59.8.7	CNØRM1	CEAD	4.59-7
4.23.8.3	CNSTRC	PLTSET	4.23-4
4.41.11.7	CØMBIN	SSG1	4.41-17
4.127.9.1	CØMBØ	ASDMAP	4.127-8
4.128.1	CØMB1	CØMB1	4.128-1
4.129.1	CØMB2	CØMB2	4.129-1
4.23.8.2	CØMECT	PLTSET	4.23-3
4.41.11.32	CØNE	SSG1	4.41-26
4.148.1	CØPY	CØPY	4.148-1
4.4.5.7	CRDFLG	XSØRT	4.4-5
4.41.11.18	CRØSS	SSG1	4.41-21
4.59.8.12	CSUB	CEAD	4.59-10
4.59.8.16	CSUMM	CEAD	4.59-13
4.59.8.11	CXTRNY	CEAD	4.59-9
4.110.1	CYCT1	CYCT1	4.110-1
4.111.1	CYCT2	CYCT2	4.111-1
4.111.8.1	CYCT2A	CYCT2	4.111-5
4.111.8.2	CYCT2B	CYCT2	4.111-5
4.78.1	DADD	ADD	4.78-1
4.96.1	DADD5	ADD5	4.96-1
4.49.8.6	DBAR	DSMG1	4.49-6
4.49.8.10	DCØNE	DSMG1	4.49-7

GENERAL COMMENTS AND INDEXES

<u>Section Number</u>	<u>Entry Point</u>	<u>Module Name</u>	<u>Page Number</u>
4.81.1	DDCØMP	DECØMP	4.81-1
4.141.1	DDRMM	DDRMM	4.141-1
4.141.8.4	DDRMMMA	DDRMM	4.141-7
4.141.8.2	DDRMM1	DDRMM	4.141-4
4.141.8.3	DDRMM2	DDRMM	4.141-6
4.67.1	DDR1	DDR1	4.67-1
4.68.1	DDR2	DDR2	4.68-1
4.68.8.1	DDR1A	DDR2	4.68-4
4.68.8.2	DDR1B	DDR2	4.68-5
4.28.8.30	DELTKL	SMA2	4.28-11
4.27.8.5	DETCK	SMA1	4.27-9
4.123.8.1	DETCKX	EMA	4.123-6
4.48.8.24	DETDET	READ	4.48-17
4.48.8.28	DETFBS	READ	4.48-19
4.48.8.15	DETM	READ	4.48-13
4.48.8.16	DETM1	READ	4.48-15
4.48.8.17	DETM2	READ	4.48-16
4.48.8.18	DETM3	READ	4.48-16
4.48.8.19	DETM4	READ	4.48-16
4.48.8.20	DETM5	READ	4.48-16
4.48.8.21	DETM6	READ	4.48-16
4.82.1	DFBS	FBS	4.82-1
4.143.1	DIAGØN	DIAGØNAL	4.143-1
4.49.8.16	DIHEX	DSMG1	4.49-7a
4.41.11.8	DIRECT	SSG1	4.41-18
4.27.8.21	DKI	SMA1	4.27-13
4.27.8.22	DKINT	SMA1	4.27-14
4.27.8.24	DK100	SMA1	4.27-15
4.27.8.25	DK211	SMA1	4.27-16
4.27.8.23	DK89	SMA1	4.27-15

MODULE FUNCTIONAL DESCRIPTIONS

<u>Section Number</u>	<u>Entry Point</u>	<u>Module Name</u>	<u>Page Number</u>
4.114.8.2	DLAMG	AMG	4.114-4
4.114.8.11	DLPT2	AMG	4.114-9
4.27.8.27	DMATRX	SMA1	4.27-17
4.26.8.7	DMFGR	TA1	4.26-15
4.28.8	DMI	SMA2	4.28-3
4.28.8	DMINT	SMA2	4.28-3
4.79.1	DMPYAD	MPYAD	4.79-1
4.28.8	DM100	SMA2	4.28-3
4.28.8	DM211	SMA2	4.28-3
4.28.8	DM89	SMA2	4.28-3
4.47.1	DPD	DPD	4.47-1
4.47.8.1	DPDAA	DPD	4.47-7
4.47.9.2	DPDCBD	DPD	4.47-8
4.47.7.1	DPD1	DPD	4.47-3
4.47.7.1	DPD2	DPD	4.47-3
4.47.7.1	DPD3	DPD	4.47-3
4.47.7.1	DPD4	DPD	4.47-3
4.47.7.1	DPD5	DPD	4.47-3
4.24.1	DPLØT	PLØT	4.24-1
4.23.1	DPLTST	PLTSET	4.23-1
4.114.8.4	DPPS	AMG	4.114-4
4.49.8.9	DQDMEM	DSMG1	4.49-7
4.49.8.12	DQUAD	DSMG1	4.49-7a
4.49.8.14	DQUADS	DSMG1	4.49-7a
4.24.8.6	DRAW	PLØT	4.24-7
4.49.8.5	DRØD	DSMG1	4.49-6
4.121.1	DSCHK	DSCHK	4.121-1
4.49.8.7	DSHEAR	DSMG1	4.49-6
4.49.1	DSMG1	DSMG1	4.49-1
4.51.1	DSMG2	DSMG2	4.51-1

GENERAL COMMENTS AND INDEXES

<u>Section Number</u>	<u>Entry Point</u>	<u>Module Name</u>	<u>Page Number</u>
4.49.8.1	DS1	DSMG1	4.49-5
4.49.8.2	DS1A	DSMG1	4.49-5
4.49.8.4	DS1ABD	DSMG1	4.49-6
4.49.8.3	DS1B	DSMG1	4.49-5
4.85.1	DTRANP	TRNSP	4.85-1
4.49.8.13	DTRBSC	DSMG1	4.49-7a
4.49.8.11	DTRIA	DSMG1	4.49-7
4.49.8.15	DTRIAS	DSMG1	4.49-7a
4.49.8.8	DTRMEM	DSMG1	4.49-7
4.94.1	DUMERG	UMERGE	4.94-1
4.7.5.10	DUMPER	XGPI	4.7-6
4.93.1	DUPART	UPARTN	4.93-1
4.24.8.11	DVECTR	PLØT	4.24-10
4.48.8.23	EADD	READ	4.48-17
4.41.11.4	EDTL	SSG1	4.41-16
4.24.8.16	ELELBL	PLØT	4.24-12
4.41.11.59	ELTKL	SSG1	4.41-34
4.123.1	EMA	EMA	4.123-1
4.124.1	EMG	EMG	4.124-1
4.124.8.2	EMGCNG	EMG	4.124-5
4.124.8.3	EMGCØR	EMG	4.124-6
4.124.8.5	EMGFIN	EMG	4.124-7
4.124.8.6	EMGØUT	EMG	4.124-7
4.124.8.4	EMGPRØ	EMG	4.124-6
4.124.8.1	EMGTAB	EMG	4.124-4
4.48.8.36	EMPCØR	READ	4.48-19c
4.112.8.4	EMSG	APD	4.112-4
4.128.8.24	ENCØDE	CØMB1	4.128-40
4.128.8.20	EQSCØD	CØMB1	4.128-38
4.128.8.26	EQSØUT	COMB1	4.128-41

MODULE FUNCTIONAL DESCRIPTIONS

<u>Section Number</u>	<u>Entry Point</u>	<u>Module Name</u>	<u>Page Number</u>
4.128.8.27	EQOUT1	CØMB1	4.128-42
4.130.8.6	EXFØRT	EXIØ	4.130-7
4.130.8.5	EXI2	EXIØ	4.130-6
4.130.1	EXIØ	EXIØ	4.130-1
4.130.8.2	EXIØ1	EXIØ	4.130-3
4.130.8.3	EXIØ2	EXIØ	4.130-4
4.130.8.7	EXLVL	EXIØ	4.130-7
4.130.8.4	EXØ2	EXIØ	4.130-5
4.41.11.2	EXTERN	SSG1	4.41-14
4.4.5.8	EXTINT	XSØRT	4.4-5
4.116.1	FA1	FA1	4.116-1
4.116.8.2	FA1K	FA1	4.116-3
4.117.1	FA2	FA2	4.117-1
4.61.8.7	FACTRU	FRRD	4.61-7
4.65.8.11	FBSINT	TRD	4.65-15
4.41.11.31	FCURL	SSG1	4.41-26
4.41.11.25	FDCSTM	SSG1	4.41-23
4.48.8.22	FDVECT	READ	4.48-17
4.41.11.26	FEDT	SSG1	4.41-24
4.41.11.37	FEDTED	SSG1	4.41-28
4.41.11.36	FEDTST	SSG1	4.41-27
4.46.8	FF100	SDR2	4.46-7
4.48.8.37	FILCØR	READ	4.48-19d
4.24.8.2	FIND	PLØT	4.24-4
4.128.8.21	FINDER	CØMB1	4.128-39
4.128.8.23	FNDGRD	CØMB1	4.128-40
4.119.8.1	FNDPAR	PARAMR	4.119-3

GENERAL COMMENTS AND INDEXES

<u>Section Number</u>	<u>Entry Point</u>	<u>Module Name</u>	<u>Page Number</u>
4.41.11.17	FNDPNT	SSG1	4.41-21
4.24.8.12	FNDSET	PLØT	4.24-11
4.41.11.20	FNDSIL	SSG1	4.41-22
4.73.8.4	FØRMAT	MATPRT	4.73-4
4.31.8.3	FØRMGG	GP4	4.31-6
4.65.8.4	FØRM1	TRD	4.65-12
4.65.8.10	FØRM2	TRD	4.65-15
4.41.11.10	FPØNT	SSG1	4.41-19
4.61.1	FRRD	FRRD	4.61-1
4.61.8.1	FRRD1A	FRRD	4.61-5
4.61.8.2	FRRD1B	FRRD	4.61-6
4.61.8.3	FRRD1C	FRRD	4.61-6
4.61.8.4	FRRD1D	FRRD	4.61-6
4.61.8.5	FRRD1E	FRRD	4.61-7
4.61.8.6	FRRD1F	FRRD	4.61-7
4.46.8	F6211	SDR2	4.46-7
4.46.8	F89	SDR2	4.46-7
4.41.11.60	GBTRAN	SSG1	4.41-35
4.114.8.3	GEND	AMG	4.114-4
4.24.8.4	GETDEF	PLØT	4.24-6
4.113.8.1	GI	GI	4.113-8
4.113.8.2	GIGGKS	GI	4.113-8
4.113.8.4	GIGTKA	GI	4.113-8
4.113.8.3	GIPSST	GI	4.113-8
4.58.1	GKAD	GKAD	4.58-1
4.58.8.1	GKAD1A	GKAD	4.58-7
4.58.8.2	GKAD1B	GKAD	4.58-7
4.58.8.3	GKAD1C	GKAD	4.58-8
4.58.8.4	GKAD1D	GKAD	4.58-8

MODULE FUNCTIONAL DESCRIPTIONS

<u>Section Number</u>	<u>Entry Point</u>	<u>Module Name</u>	<u>Page Number</u>
4.66.1	GKAM	GKAM	4.66-1
4.66.8.2	GKAM1A	GKAM	4.66-4
4.66.8.1	GKAM1B	GKAM	4.66-3
4.41.11.22	GLBBAS	SSG1	4.41-23
4.109.1	GPCYC	GPCYC	4.109-1
4.146.1	GPFDR	GPFDR	4.146-1
4.32.1	GPSP	GPSP	4.32-1
4.24.8.10	GPTLBL	PLØT	4.24-10
4.24.8.9	GPTSYM	PLØT	4.24-9
4.29.1	GPWG	GPWG	4.29-1
4.29.8.1	GPWG1A	GPWG	4.29-5
4.29.8.2	GPWG1B	GPWG	4.29-5
4.29.8.3	GPWG1C	GPWG	4.29-6
4.21.1	GP1	GP1	4.21-1
4.22.1	GP2	GP2	4.22-1
4.25.1	GP3	GP3	4.25-1
4.25.8.3	GP3A	GP3	4.25-4
4.25.8.4	GP3B	GP3	2.25-7
4.25.8.2	GP3C	GP3	4.25-2
4.25.8.5	GP3D	GP3	4.25-8a
4.31.1	GP4	GP4	4.31-1
4.31.8.1	GP4PRT	GP4	4.31-6
4.41.11.15	GRAV	SSG1	4.41-20
4.41.11.5	GRAVL1	SSG1	4.41-16
4.41.11.6	GRAVL2	SSG1	4.41-16
4.41.11.27	GRAVL3	SSG1	4.41-24
4.128.8.22	GRIDIP	CØMB1	4.128-39
4.128.8.28	GTMAT1	CØMB1	4.128-42
4.128.8.28	GTMAT2	CØMB1	4.128-42

GENERAL COMMENTS AND INDEXES

<u>Section Number</u>	<u>Entry Point</u>	<u>Module Name</u>	<u>Page Number</u>
4.128.8.28	GTMAT3	CØMB1	4.128-42
4.41.11.38	HBDY	SSG1	4.41-28
4.59.8.20	HESS1	CEAD	4.59-14
4.59.8.21	HESS2	CEAD	4.59-15
4.27.8.34	HRING	SMA1	4.27-19
4.112.8.3	IAPD	APD	4.112-3
4.114.8.9	IDF1	AMG	4.114-8
4.114.8.10	IDF2	AMG	4.114-8
4.5.1	IFP	IFP	4.5-1
4.5.7.9	IFPDCØ	IFP	4.5-7
4.3.1	IFP1	IFP1	4.3-1
4.3.7.1	IFP1B	IFP1	4.3-2
4.3.7.2	IFP1C	IFP1	4.3-3
4.3.7.3	IFP1D	IFP1	4.3-4
4.3.7.4	IFP1E	IFP1	4.3-4
4.3.7.5	IFP1F	IFP1	4.3-4
4.3.7.6	IFP1G	IFP1	4.3-5
4.3.7.8	IFP1XY	IFP1	4.3-6
4.6.1	IFP3	IFP3	4.6-1
4.6.8.1	IFP3B	IFP3	4.6-15
4.89.1	IFP4	IFP4	4.89-1
4.89.8.1	IFP4A	IFP4	4.89-15
4.89.8.2	IFP4B	IFP4	4.89-15
4.89.8.3	IFP4C	IFP4	4.89-16
4.89.8.4	IFP4E	IFP4	4.89-16
4.89.8.5	IFP4F	IFP4	4.89-17
4.89.8.6	IFP4G	IFP4	4.89-17
4.91.1	IFP5	IFP	4.91-1
4.91.8.1	IFP5A	IFP	4.91-7

MODULE FUNCTIONAL DESCRIPTIONS

<u>Section Number</u>	<u>Entry Point</u>	<u>Module Name</u>	<u>Page Number</u>
4.5.7.8	IFS1P	IFP	4.5-6
4.5.7.8	IFS2P	IFP	4.5-6
4.5.7.8	IFS3P	IFP	4.5-6
4.5.7.8	IFS4P	IFP	4.5-6
4.5.7.8	IFS5P	IFP	4.5-6
4.5.7.1	IFX1BD	IFP	4.5-5
4.5.7.2	IFX2BD	IFP	4.5-5
4.5.7.3	IFX3BD	IFP	4.5-5
4.5.7.4	IFX4BD	IFP	4.5-6
4.5.7.5	IFX5BD	IFP	4.5-6
4.5.7.6	IFX6BD	IFP	4.5-6
4.5.7.7	IFX7BD	IFP	4.5-6
4.41.11.54	IHEX	SSG1	4.41-33
4.27.8.42	IHEXSD	SMA1	4.27-20
4.46.8.48	IHEXSS	SDR2	4.46-22
4.114.8.7	INCRØ	AMG	4.114-6
4.4.5.3	INITCØ	XSØRT	4.4-4
4.65.8.2	INITL	TRD	4.65-11
4.97.8	INPABD	INPUT	4.97-3
4.98.1	INPTT1	INPUTT1	4.98-1
4.99.1	INPTT2	INPUTT2	4.99-1
4.97.1	INPUT	INPUT	4.97-1
4.4.5.9	INTEXT	XSØRT	4.5-5
4.65.8.7	INTFBS	TRD	4.65-13
4.73.8.1	INTPRT	MATPRT	4.73-1
4.24.8.7	INTVEC	PLØT	4.24-8
4.48.8.40	INVERT	READ	4.48-19e
4.48.8.14	INVFBFS	READ	4.48-12
4.48.8.6	INVPWR	READ	4.48-8

GENERAL COMMENTS AND INDEXES

<u>Section Number</u>	<u>Entry Point</u>	<u>Module Name</u>	<u>Page Number</u>
4.48.8.7	INVP1	READ	4.48-8
4.48.8.8	INVP2	READ	4.48-9
4.48.8.9	INVP3	READ	4.48-9
4.48.8.41	INVTR	READ	4.48-19f
4.4.5.11	ISFT	XSØRT	4.4-6
4.97.8	IUNIØN	INPUT	4.97-3
4.27.8.7	KBAR	SMA1	4.27-10
4.27.8.17	KCØNE	SMA1	4.27-12
4.27.8.17	KCØNEX	SMA1	4.27-12
4.27.8.16	KELAS	SMA1	4.27-12
4.27.8.28	KFLUD2	SMA1	4.27-17
4.27.8.29	KFLUD3	SMA1	4.27-17
4.27.8.30	KFLUD4	SMA1	4.27-18
4.27.8.41	KIHEX	SMA1	4.27-20
4.27.8.9	KPANEL	SMA1	4.27-10
4.27.8.35	KPLTST	SMA1	4.27-19
4.27.8.11	KQDMEM	SMA1	4.27-11
4.27.8.36	KQDMM1	SMA1	4.27-19
4.27.8.14	KQDPLT	SMA1	4.27-12
4.27.8.39	KQUADS	SMA1	4.27-20
4.27.8.6	KRØD	SMA1	4.27-10
4.27.8.31	KSLØT	SMA1	4.27-18
4.27.8.33	KSØLID	SMA1	4.27-18
4.27.8.32	KTETRA	SMA1	4.27-18
4.27.8.20	KTØRDR	SMA1	4.27-13
4.27.8.38	KTPZ	SMA1	4.27-19
4.27.8.19	KTRAPR	SMA1	4.27-13
4.27.8.12	KTRBSC	SMA1	4.27-11
4.27.8.37	KTRIA	SMA1	4.27-19
4.27.8.40	KTRIAS	SMA1	4.27-20

MODULE FUNCTIONAL DESCRIPTIONS

<u>Section Number</u>	<u>Entry Point</u>	<u>Module Name</u>	<u>Page Number</u>
4.27.8.15	KTRIQQ	SMA1	4.27-12
4.27.8.18	KTRIRG	SMA1	4.27-13
4.27.8.10	KTRMEM	SMA1	4.27-11
4.27.8.13	KTRPLT	SMA1	4.27-11
4.27.8.8	KTUBE	SMA1	4.27-10
4.2.5.2	LDi	XCSA	4.2-2
4.24.8.18	LINEL	PLØT	4.24-12a
4.7.5.15	LINKUP	XGPI	4.7-7
4.46.8.35	MAGPHA	SDR2	4.46-18
4.74.8.1	MAP	SEEMAT	4.74-4
4.74.8.1	MAPSET	SEEMAT	4.74-4
4.28.8.12	MASSD	SMA2	4.28-7
4.28.8.7	MASSTQ	SMA2	4.28-5
4.72.1	MATGPR	MATGPR	4.72-1
4.71.1	MATPRN	MATPRN	4.71-1
4.73.8.2	MATPRT	MATPRT	4.73-2
4.65.8.5	MATVEC	TRD	4.65-13
4.28.8.8	MBAR	SMA2	4.28-6
4.28.8.9	MCBAR	SMA2	4.28-6
4.33.1	MCE1	MCE1	4.33-1
4.33.8.1	MCE1A	MCE1	4.33-3
4.33.8.2	MCE1B	MCE1	4.33-3
4.33.8.3	MCE1C	MCE1	4.33-3
4.33.8.4	MCE1D	MCE1	4.33-3
4.34.1	MCE2	MCE2	4.34-1
4.28.8.11	MCØNE	SMA2	4.28-6
4.28.8.10	MCØNMX	SMA2	4.28-6
4.28.8.17	MCRØD	SMA2	4.28-8
4.59.8.22	MERGED	CEAD	4.59-16
4.84.1	MERGE1	MERGE	4.84-1

GENERAL COMMENTS AND INDEXES

<u>Section Number</u>	<u>Entry Point</u>	<u>Module Name</u>	<u>Page Number</u>
4.31.8.4	MFGR	GP4	4.31-7
4.28.8.22	MFLUD2	SMA2	4.28-9
4.28.8.23	MFLUD3	SMA2	4.28-9
4.28.8.24	MFLUD4	SMA2	4.28-9
4.28.8.25	MFREE	SMA2	4.28-10
4.28.8.28	MHBDY	SMA2	4.28-10
4.28.8.35	MIHEX	SMA2	4.28-12
4.24.8.13	MINMAX	PLØT	4.24-11
4.116.8.3	MINTRP	FA1	4.116-4
4.126.1	MØDACC	MØDACC	4.126-1
4.126.8.1	MØDAC1	MØDACC	4.126-3
4.126.8.2	MØDAC2	MØDACC	4.126-3
4.7.5.11	MPLPRT	XGPI	4.7-6
4.41.11.23	MPYL	SSG1	4.41-23
4.41.11.24	MPYLT	SSG1	4.41-23
4.28.8.32	MPZDA	SMA2	4.28-11
4.28.8.20	MQDPLT	SMA2	4.28-9
4.28.8.33	MQUADS	SMA2	4.28-11
4.28.8.29	MRING	SMA2	4.28-10
4.28.8.5	MRØD	SMA2	4.28-5
4.28.8.26	MSLØT	SMA2	4.28-10
4.28.8.27	MSØLID	SMA2	4.28-10
4.28.8.31	MSTRIA	SMA2	4.28-11
4.48.8.11	MTIMSU	READ	4.48-10
4.28.8.15	MTØRDR	SMA2	4.28-7
4.28.8.14	MTRAPR	SMA2	4.28-7
4.28.8.18	MTRBSC	SMA2	4.28-8
4.28.8.34	MTRIAS	SMA2	4.28-12
4.28.8.21	MTRIQD	SMA2	4.28-9
4.28.8.13	MTRIRG	SMA2	4.28-7

MODULE FUNCTIONAL DESCRIPTIONS

<u>Section Number</u>	<u>Entry Point</u>	<u>Module Name</u>	<u>Page Number</u>
4.28.8.19	MTRPLT	SMA2	4.28-8
4.57.1	MTRXIN	MTRXIN	4.57-1
4.28.8.6	MTUBE	SMA2	4.28-5
4.41.11.19	NQRM	SSG1	4.41-21
4.48.8.10	NQRM1	READ	4.48-10
4.70.1	QFP	QFP	4.70-1
4.70.8.1	QFPUN	QFP	4.70-2
4.70.8.2	QFP1	QFP	4.70-3
4.70.8.3	QFP1A	QFP	4.70-3
4.70.8.4	QFP1BD	QFP	4.70-3
4.70.8.5	QFP5BD	QFP	4.70-3
4.70.8.6	QF1PBD	QFP	4.70-3
4.70.8.7	QF2PBD	QFP	4.70-3
4.70.8.8	QF3PBD	QFP	4.70-3
4.70.8.9	QF4PBD	QFP	4.70-3
4.70.8.10	QF5PBD	QFP	4.70-4
4.70.8.11	QF6PBD	QFP	4.70-4
4.70.8.12	QF7PBD	QFP	4.70-4
4.70.8.13	QF8PBD	QFP	4.70-4
4.70.8.14	QF9PBD	QFP	4.70-4
4.120.8.3	QPTP1A	QPTPR1	4.120-5
4.120.8.4	QPTP1B	QPTPR1	4.120-5
4.120.8.5	QPTP1C	QPTPR1	4.120-6
4.120.8.6	QPTP1D	QPTPR1	4.120-7
4.120.1	QPTPR1	QPTPR1	4.120-1
4.142.1	QPTPR2	QPTPR2	4.142-1
4.142.8.1	QPT2A	QPTPR2	4.142-3
4.142.8.2	QPT2B	QPTPR2	4.142-4

GENERAL COMMENTS AND INDEXES

<u>Section Number</u>	<u>Entry Point</u>	<u>Module Name</u>	<u>Page Number</u>
4.142.8.3	ØPT2C	ØPTPR2	4.142-5
4.142.8.4	ØPT2D	ØPTPR2	4.142-6
4.120.8.1	ØPTPX	ØPTPR1	4.120-3
4.120.8.2	ØPTPX1	ØPTPR1	4.120-4
4.48.8.5	ØRTCK	READ	4.48-7
4.59.8.13	ØRTHØ	CEAD	4.59-10
4.7.5.12	ØSCXRF	XGPI	4.7-6
4.7.5.16	ØUTPAK	XGPI	4.7-8
4.100.1	ØUTPT1	ØUTPUT1	4.100-1
4.101.1	ØUTPT2	ØUTPUT2	4.101-1
4.102.1	ØUTPT3	ØUTPUT3	4.102-1
4.24.8.1	PARAM	PLØT	4.24-4
4.118.1	PARAML	PARAML	4.118-1
4.83.1	PARTN1	PARTN	4.83-1
4.83.8.1	PARTN2	PARTN	4.83-3
4.83.8.2	PARTN3	PARTN	4.83-4
4.41.11.16	PERMUT	SSG1	4.41-21
4.24.8.14	PERPEC	PLØT	4.24-11
4.102.8.1	PHDMIA	ØUTPUT3	4.102-2
4.55.8.5	PKBAR	PLA4	4.55-4
4.55.8.10	PKQAD1	PLA4	4.55-5
4.55.8.11	PKQAD2	PLA4	4.55-5
4.55.8.7	PKQDM	PLA4	4.55-5
4.55.8.18	PKQDMS	PLA4	4.55-7
4.55.8.13	PKQDM1	PLA4	4.55-6
4.55.8.22	PKQDPL	PLA4	4.55-8
4.55.8.4	PKRØD	PLA4	4.55-4
4.55.8.14	PKTQ1	PLA4	4.55-6
4.55.8.16	PKTQ2	PLA4	4.55-7

MODULE FUNCTIONAL DESCRIPTIONS

<u>Section Number</u>	<u>Entry Point</u>	<u>Module Name</u>	<u>Page Number</u>
4.55.8.20	PKTRBS	PLA4	4.55-8
4.55.8.8	PKTRI1	PLA4	4.55-5
4.55.8.9	PKTRI2	PLA4	4.55-5
4.55.8.6	PKTRM	PLA4	4.55-4
4.55.8.17	PKTRMS	PLA4	4.55-7
4.55.8.12	PKTRM1	PLA4	4.55-6
4.55.8.21	PKTRPL	PLA4	4.55-8
4.55.8.19	PKTRQD	PLA4	4.55-7
4.55.8.15	PKTRQ2	PLA4	4.55-6
4.52.1	PLA1	PLA1	4.52-1
4.53.1	PLA2	PLA2	4.53-1
4.54.1	PLA3	PLA3	4.54-1
4.54.8.1	PLA31	PLA3	4.54-3
4.54.8.2	PLA32	PLA3	4.54-4
4.55.1	PLA4	PLA4	4.55-1
4.55.8.1	PLA41	PLA4	4.55-3
4.55.8.2	PLA42	PLA4	4.55-3
4.55.8.3	PLA4B	PLA4	4.55-4
4.41.11.12	PLØAD	SSG1	4.41-19
4.41.11.53	PLØAD3	SSG1	4.41-32
4.24.8.3	PLØT	PLØT	4.24-5
4.139.1	PLTMRG	PLTMRG	4.139-1
4.24.8.5	PLTØPR	PLØT	4.25-6a
4.92.1	PLTTRA	PLTTRAN	4.92-1
4.41.11.14	PRESAX	SSG1	4.41-20
4.24.8.15	PRØCES	PLØT	4.24-12
4.73.1	PRTINT	MATPRT	4.73-1
4.73.8.2	PRTMAT	MATPRT	4.73-2
4.76.1	PRTMSG	PRTMSG	4.76-1
4.77.1	PRTPRM	PRTPARM	4.77-1

GENERAL COMMENTS AND INDEXES

<u>Section Number</u>	<u>Entry Point</u>	<u>Module Name</u>	<u>Page Number</u>
4.73.8.3	PRTVEC	MATPRT	4.73-3
4.54.8.4	PSBAR	PLA3	4.54-4
4.54.8.9	PSQAD1	PLA3	4.54-5
4.54.8.10	PSQAD2	PLA3	4.54-5
4.54.8.6	PSQDM	PLA3	4.54-4
4.54.8.12	PSQDM1	PLA3	4.54-6
4.54.8.16	PSQPL1	PLA3	4.54-7
4.54.8.3	PSRØD	PLA3	4.54-4
4.54.8.15	PSTPL1	PLA3	4.54-6
4.54.8.13	PSTQ1	PLA3	4.54-6
4.54.8.18	PSTQ2	PLA3	4.54-7
4.54.8.14	PSTRB1	PLA3	4.54-6
4.54.8.7	PSTRI1	PLA3	4.54-5
4.54.8.8	PSTRI2	PLA3	4.54-5
4.54.8.5	PSTRM	PLA3	4.54-4
4.54.8.11	PSTRM1	PLA3	4.54-5
4.54.8.17	PSTRQ2	PLA3	4.54-7
4.127.12.1	PULL	ASDMAP	4.127-11
4.127.11.1	PUSH	ASDMAP	4.127-11
4.145.8.4	PVEC	PVEC11	4.145-4
4.145.8.5	PVECA	PVEC11	4.145-5
4.145.8.7	PVECK	PVEC11	4.145-9
4.145.8.6	PVECP	PVEC11	4.145-8
4.145.8.8	PVECS	PVEC11	4.145-9
4.145.8.1	PVEC05	PVEC11	4.145-3
4.145.8.2	PVEC10	PVEC11	4.145-4
4.145.8.3	PVEC20	PVEC11	4.145-4
4.41.11.33	QDMEM	SSG1	4.41-26
4.41.11.52	QDMM1	SSG1	4.41-32
4.41.11.40	QDPLT	SSG1	4.41-28

MODULE FUNCTIONAL DESCRIPTIONS

<u>Section Number</u>	<u>Entry Point</u>	<u>Module Name</u>	<u>Page Number</u>
4.41.11.39	QHBDY	SSG1	4.41-28
4.41.11.51	QLØADL	SSG1	4.41-32
4.19.1	QPARAM	PARAM	4.19-1
4.119.1	QPARMR	PARAMR	4.119-1
4.48.8.38	QRITER	READ	4.48-19d
4.41.11.50	QVØL	SSG1	4.41-31
4.64.1	RANDØM	RANDØM	4.64-1
4.64.8.4	RAND1	RANDØM	4.64-6
4.64.8.5	RAND2	RANDØM	4.64-6
4.64.8.5	RAND2A	RANDØM	4.64-6
4.64.8.6	RAND3	RANDØM	4.64-7
4.64.8.7	RAND4	RANDØM	4.64-7
4.64.8.2	RAND5	RANDØM	4.64-6
4.64.8.8	RAND6	RANDØM	4.64-7
4.64.8.1	RAND7	RANDØM	4.64-5
4.64.8.3	RAND8	RANDØM	4.64-6
4.37.1	RBMG1	RBMG1	4.37-1
4.38.1	RBMG2	RBMG2	4.38-1
4.39.1	RBMG3	RBMG3	4.39-1
4.40.1	RBMG4	RBMG4	4.40-1
4.131.8.2	RCØVA	RCØVR	4.131-4
4.131.8.3	RCØVB	RCØVR	4.131-5
4.131.8.4	RCØVC	RCØVR	4.131-9
4.131.1	RCØVR	RCØVR	4.131-1
4.132.1	RCØVR3	RCØVR3	4.132-1
4.48.8.1	READ1	READ	4.48-4
4.48.8.2	READ2	READ	4.48-5
4.48.8.3	READ3	READ	4.48-6
4.48.8.4	READ4	READ	4.48-7
4.48.8.2	READ5	READ	4.48-5

GENERAL COMMENTS AND INDEXES

<u>Section Number</u>	<u>Entry Point</u>	<u>Module Name</u>	<u>Page Number</u>
4.48.8.42	READ6	READ	4.48.19f
4.48.8.43	READ7	READ	4.48-19g
4.127.10.1	REDU	ASDMAP	4.127-10
4.133.1	REDUCE	REDUCE	4.133-1
4.48.1	REIG	READ	4.48-1
4.41.11.13	RFØRCE	SSG1	4.41-20
4.104.1	RMG	RMG	4.104-1
4.41.11.41	RØD	SSG1	4.41-28
4.27.8.26	RØMBDK	SMA1	4.27-16
4.46.8	RØMBER	SDR2	4.46-7
4.48.8.35	RØTATE	READ	4.48-19c
4.48.8.34	RØTAX	READ	4.48-19c
4.4.5.2	RPAGE	XSØRT	4.4-3
4.46.8.36	SAXIF1	SDR2	4.46-18
4.46.8.37	SAXIF2	SDR2	4.46-18
4.46.8.14	SBAR1	SDR2	4.46-12
4.46.8.31	SBAR2	SDR2	4.46-17
4.46.8.26	SBSPL2	SDR2	4.46-16
4.144.1	SCALAR	SCALAR	4.144-1
4.31.8.2	SCALEX	GP4	4.31-6
4.35.1	SCE1	SCE1	4.35-1
4.46.8.15	SCONE1	SDR2	4.46-12
4.46.8.29	SCONE2	SDR2	4.46-17
4.46.8.30	SCONE3	SDR2	4.46-17
4.46.8.44	SDHTFF	SDR2	4.46-21
4.46.8.43	SDHTF1	SDR2	4.46-20
4.46.8.45	SDHTF2	SDR2	4.46-21
4.46.8.42	SDRETD	SDR2	4.46-20
4.108.1	SDRHT	SDRHT	4.108-1

MODULE FUNCTIONAL DESCRIPTIONS

<u>Section Number</u>	<u>Entry Point</u>	<u>Module Name</u>	<u>Page Number</u>
4.45.1	SDR1	SDR1	4.45-1
4.45.8.1	SDR1A	SDR1	4.45-6
4.45.8.2	SDR1C	SDR1	4.45-7
4.45.8.3	SDR1D	SDR1	4.45-7
4.46.1	SDR2	SDR2	4.46-1
4.46.8.2	SDR2A	SDR2	4.46-8
4.46.8.1	SDR2AA	SDR2	4.46-8
4.46.8.3	SDR2B	SDR2	4.46-9
4.46.8.20	SDR2C	SDR2	4.46-13
4.46.8.21	SDR2D	SDR2	4.46-14
4.46.8.22	SDR2E	SDR2	4.46-15
4.62.1	SDR3	SDR3	4.62-1
4.62.8.1	SDR3A	SDR3	4.62-9
4.74.1	SEEMAT	SEEMAT	4.74-1
4.46.8.12	SELAS1	SDR2	4.46-11
4.46.8.25	SELAS2	SDR2	4.46-16
4.23.8.1	SETINP	PLTSET	4.23-3
4.20.1	SETVAL	SETVAL	4.20-1
4.134.1	SGEN	SGEN	4.134-1
4.24.8.8	SHAPE	PLØT	4.24-8
4.48.8.32	SICØX	READ	4.48-19b
4.46.9.47	SIHEX1	SDR2	4.46-21
4.46.8.49	SIHEX2	SDR2	4.46-22
4.48.8.33	SINCAS	READ	4.48-19b
4.41.11.11	SLØAD	SSG1	4.41-19
4.27.8.1	SMA1	SMA1	4.27-8
4.27.8.2	SMA1A	SMA1	4.27-8
4.27.8.3	SMA1B	SMA1	4.27-9
4.27.8.4	SMA1BD	SMA1	4.27-9
4.28.1	SMA2	SMA2	4.28-1

GENERAL COMMENTS AND INDEXES

<u>Section Number</u>	<u>Entry Point</u>	<u>Module Name</u>	<u>Page Number</u>
4.28.8.2	SMA2A	SMA2	4.28-4
4.28.8.3	SMA2B	SMA2	4.28-4
4.28.8.4	SMA2BD	SMA2	4.28-4
4.30.1	SMA3	SMA3	4.30-1
4.30.8.1	SMA3A	SMA3	4.30-5
4.30.8.2	SMA3B	SMA3	4.30-5
4.30.8.4	SMA3BD	SMA3	4.30-6
4.30.8.3	SMA3C	SMA3	4.30-6
4.48.8.30	SMLEIG	READ	4.48-19a
4.86.1	SMPYAD	SMPYAD	4.86-1
4.36.1	SMP1	SMP1	4.36-1
4.50.1	SMP2	SMP2	4.50-1
4.114.8.6	SNPDF	AMG	4.114-5
4.136.1	SØFØ	SØFØ	4.136-1
4.135.1	SØFI	SØFI	4.135-1
4.137.1	SØFUT	SØFUT	4.137-1
4.41.11.42	SØLID	SSG1	4.41-29
4.80.1	SØLVE	SØLVE	4.80-1
4.46.8.6	SPANL1	SDR2	4.46-10
4.46.8.24	SPANL2	SDR2	4.46-15
4.134.8	SPLT10	SGEN	4.134-5
4.46.8.19	SQDM11	SDR2	4.46-13
4.46.8.46	SQDM12	SDR2	4.46-21
4.46.8.11	SQDME1	SDR2	4.46-11
4.46.8.9	SQDPL1	SDR2	4.46-11
4.46.8.27	SQRTM	READ	4.48-19
4.46.8.52	SQUAD1	SDR2	4.46-22
4.46.8.54	SQUAD2	SDR2	4.46-23
4.46.8.4	SRØD1	SDR2	4.46-10

MODULE FUNCTIONAL DESCRIPTIONS

<u>Section Number</u>	<u>Entry Point</u>	<u>Module Name</u>	<u>Page Number</u>
4.46.8.23	SRØD2	SDR2	4.46-15
4.41.11.48	SSGETD	SSG1	4.41-31
4.105.1	SSGHT	SSGHT	4.105-1
4.105.8.1	SSGHTP	SSGHT	4.105-6
4.105.8.2	SSGHT1	SSGHT	4.105-6
4.105.8.3	SSGHT2	SSGHT	4.105-7
4.41.11.47	SSGKHI	SSG1	4.41-30
4.41.11.49	SSGSLT	SSG1	4.41-31
4.41.1	SSG1	SSG1	4.41-1
4.41.11.1	SSG1A	SSG1	4.41-14
4.42.1	SSG2	SSG2	4.42-1
4.42.9	SSG2B1	SSG2	4.42-4
4.43.1	SSG3	SSG3	4.43-1
4.44.1	SSG4	SSG4	4.44-1
4.46.8.38	SSLØT1	SDR2	4.46-19
4.46.8.39	SSLØT2	SDR2	4.46-19
4.46.8.40	SSØLD1	SDR2	4.46-19
4.46.8.41	SSØLD2	SDR2	4.46-20
4.65.8.6	STEP	TRD	4.65-13
4.46.8.18	STØRD1	SDR2	4.46-13
4.46.8.34	STØRD2	SDR2	4.46-18
4.46.8.50	STPAX1	SDR2	4.46-22
4.46.8.56	STPAX2	SDR2	4.46-23
4.46.8.57	STPAX3	SDR2	4.46-23
4.46.8.27	STQME2	SDR2	4.46-16
4.46.8.17	STRAP1	SDR2	4.46-12
4.46.8.33	STRAP2	SDR2	4.46-18
4.46.8.51	STRAX1	SDR2	4.46-22
4.46.8.58	STRAX2	SDR2	4.46-24
4.46.8.59	STRAX3	SDR2	4.46-24

GENERAL COMMENTS AND INDEXES

<u>Section Number</u>	<u>Entry Point</u>	<u>Module Name</u>	<u>Page Number</u>
4.46.8.7	STRBS1	SDR2	4.46-10
4.46.8.53	STRIA1	SDR2	4.46-23
4.46.8.55	STRIA2	SDR2	4.46-23
4.46.8.16	STRIR1	SDR2	4.46-12
4.46.8.32	STRIR2	SDR2	4.46-17
4.46.8.10	STRME1	SDR2	4.46-11
4.46.8.8	STRPL1	SDR2	4.46-10
4.46.8.13	STRQD1	SDR2	4.46-12
4.46.8.28	STRQD2	SDR2	4.46-16
4.46.8.5	STUBE1	SDR2	4.46-10
4.48.8.13	SUB	READ	4.48-11
4.114.8.5	SUBP	AMG	4.114-5
4.138.1	SUBPH1	SUBPH1	4.138-1
4.48.8.26	SUMM	READ	4.48-18
4.24.8.19	SUPLT	PLØT	4.24-12c
4.147.1	SWITCH	SWITCH	4.147-1
4.3.7.7	SWSRT	IFP1	4.3-6
4.103.1	TABFMT	TABPRT	4.103-1
4.122.1	TABPCH	TABPCH	4.122-1
4.75.1	TABPT	TABPT	4.75-1
4.26.8.1	TA1	TA1	4.26-14
4.26.8.2	TA1A	TA1	4.26-14
4.26.8.3	TA1B	TA1	4.26-15
4.26.8.5	TA1C	TA1	4.26-15
4.26.8.6	TA1CA	TA1	4.26-15
4.26.8.8	TA1ETD	TA1	4.26-15
4.26.8.4	TA1H	TA1	4.26-15
4.41.11.3	TEMPL	SSG1	4.41-15
4.41.11.43	TETRA	SSG1	4.41-29
4.140.1	TIMTST	TIMETEST	4.140-1

MODULE FUNCTIONAL DESCRIPTIONS

<u>Section Number</u>	<u>Entry Point</u>	<u>Module Name</u>	<u>Page Number</u>
4.140.8.1	TIMTS1	TIMETEST	4.140-2
4.140.8.2	TIMST2	TIMETEST	4.140-3
4.140.8.3	TIMTS3	TIMETEST	4.140-3
4.140.8.4	TIMTS4/TIMTS5	TIMETEST	4.140-3
4.140.8.5	TIMTS6	TIMETEST	4.140-3
4.140.8.6	TIMTS7	TIMETEST	4.140-4
4.140.8.7	TIMTS8	TIMETEST	4.140-4
4.114.8.8	TKER	AMG	4.114-7
4.41.11.9	TPØNT	SSG1	4.41-19
4.41.11.57	TPZTEM	SSG1	4.41-34
4.41.11.55	TQUADS	SSG1	4.41-33
4.41.11.44	TRBSC	SSG1	4.41-29
4.65.1	TRD	TRD	4.65-1
4.65.8.1	TRD1A	TRD	4.65-11
4.65.8.3	TRD1C	TRD	4.65-11
4.65.8.8	TRD1D	TRD	4.65-14
4.65.8.9	TRD1E	TRD	4.65-15
4.107.1	TRHT	TRHT	4.107-1
4.107.8.1	TRHT1A	TRHT	4.107-5
4.107.8.2	TRHT1B	TRHT	4.107-6
4.107.8.3	TRHT1C	TRHT	4.107-6
4.48.8.31	TRIDI	READ	4.48-19a
4.41.11.34	TRIMEM	SSG1	4.41-26
4.41.11.45	TRIQD	SSG1	4.41-30
4.106.1	TRLG	TRLG	4.106-1
4.106.8.1	TRLGA	TRLG	4.106-8
4.106.8.2	TRLGB	TRLG	4.106-8
4.106.8.3	TRLGC	TRLG	4.106-8
4.106.8.4	TRLGD	TRLG	4.106-9

GENERAL COMMENTS AND INDEXES

<u>Section Number</u>	<u>Entry Point</u>	<u>Module Name</u>	<u>Page Number</u>
4.85.1	TRNSP	TRNSP	4.85-1
4.41.11.58	TRTTEM	SSG1	4.41-34
4.41.11.46	TRPLT	SSG1	4.41-30
4.41.11.30	TTØRDR	SSG1	4.41-25
4.41.11.29	TTRAPR	SSG1	4.41-25
4.41.11.56	TTRIAS	SSG1	4.41-33
4.41.11.28	TTRIRG	SSG1	4.41-25
4.8.1	UMFEDT	UMFEDIT	4.8-1
4.8.6	UMFZBD	UMFEDIT	4.8-2
4.48.8.29	VALVEC	READ	4.48-19
4.60.8.1	VDR	VDR	4.60-6
4.60.8.2	VDRA	VDR	4.60-6
4.60.8.3	VDRB	VDR	4.60-6
4.60.9.2	VDRBD	VDR	4.60-7
4.95.1	VEC	VEC	4.95-1
4.73.8.3	VECPRT	MATPRT	4.73-3
4.48.8.39	WILVEC	READ	4.48-19e
4.76.8.2	WRTMSG	PRTMSG	4.76-2
4.24.8.17	WRTPRT	PLØT	4.24-12a
4.4.5.5	XBCDBI	XSØRT	4.4-4
4.7.6.2	XBSBD	XGPI	4.7-10
4.11.1	XCEI	REPT	4.11-1
4.11.6.1	XCEI	REPT	4.11-2
4.12.1	XCEI	JUMP	4.12-1
4.13.1	XCEI	CØND	4.13-1
4.14.1	XCEI	EXIT	4.14-1
4.18.1	XCEI	END	4.18-1
4.10.1	XCHK	CHKPNT	4.10-1
4.9.5.2	XCLEAN	XSFA	4.9-4

MODULE FUNCTIONAL DESCRIPTIONS

<u>Section Number</u>	<u>Entry Point</u>	<u>Module Name</u>	<u>Page Number</u>
4.2.1	XCSA	XCSA	4.2-1
4.9.5.4	XDPH	XSFA	4.9-6
4.17.1	XEQUIV	EQUIV	4.17-1
4.4.5.4	XFADJ	XSØRT	4.4-4
4.4.5.10	XFADJ1	XSØRT	4.4-5
4.9.5.7	XFLIPS	XSFA	4.9-7
4.7.5.9	XFLDEF	XGPI	4.7-5
4.7.5.8	XFLØRD	XGPI	4.7-5
4.7.1	XGPI	XGPI	4.7-1
4.7.6.2	XGPIBD	XGPI	4.7-10
4.7.5.2	XGPBS	XGPI	4.7-3
4.7.5.1	XGPIDG	XGPI	4.7-3
4.7.5.1	XGPIMW	XGPI	4.7-3
4.7.5.5	XIPFL	XGPI	4.7-4
4.7.5.4	XLNKHD	XGPI	4.7-4
4.7.6.2	XMP LBD	XGPI	4.7-10
4.7.5.5	XØPFL	XGPI	4.7-4
4.7.5.3	XØSGEN	XGPI	4.7-3
4.7.5.6	XPARAM	XGPI	4.7-4
4.9.5.6	XPLEQK	XSFA	4.9-6
4.9.5.5	XPØLCK	XSFA	4.9-6
4.4.5.6	XPRETY	XSØRT	4.4-4
4.9.5.3	XPUNP	XSFA	4.9-5
4.16.1	XPURGE	PURGE	4.16-1
4.4.5.1	XRECPS	XSØRT	4.4-3
4.2.5.1	XRGDFM	XCSA	4.2-1
4.15.1	XSAVE	XSAVE	4.15-1
4.2.5.3	XSBSET	XCSA	4.2-2
4.7.5.7	XSCNDM	XGPI	4.7-4
4.9.1	XSFA	XSFA	4.9-1

GENERAL COMMENTS AND INDEXES

<u>Section Number</u>	<u>Entry Point</u>	<u>Module Name</u>	<u>Page Number</u>
4.4.1	XSØRT	XSØRT	4.4-1
4.9.5.1	XSØSGN	XSFA	4.9-3
4.48.8.12	XTRNSY	READ	4.48-11
4.63.8.7	XYCHAR	XYTRAN	4.63-7
4.63.8.1	XYDUMP	XYTRAN	4.63-5
4.63.8.2	XYFIND	XYTRAN	4.63-5
4.63.8.8	XYGRAF	XYTRAN	4.63-8
4.63.8.4	XYLØG	XYTRAN	4.63-6
4.63.8.3	XYØUT	XYTRAN	4.63-6
4.69.1	XYPLØT	XYPLØT	4.69-1
4.63.8.6	XYPRPL	XYTRAN	4.63-7
4.63.8.5	XYTICS	XYTRAN	4.63-7
4.63.1	XYTRAN	XYTRAN	4.63-1

EXECUTIVE PREFACE MODULE XCSA (EXECUTIVE CONTROL SECTION ANALYSIS)

4.2 EXECUTIVE PREFACE MODULE XCSA (EXECUTIVE CONTROL SECTION ANALYSIS)

4.2.1 Entry Point: XCSA

4.2.2 Purpose:

To process the NASTRAN Executive Control Deck.

4.2.3 Calling Sequence

CALL XCSA. XCSA is called only by subroutine SEMINT.

4.2.4 Method

The cards of the Executive Control Deck are read and processed with checks being made for illegal formats, duplication and errors peculiar to the particular card being processed. When all of the control cards have been processed (i.e., CEND control card found), the Executive Control Table (XCSA) is written on the Problem Tape and XCSA returns to the calling routine.

4.2.5 Subroutines

4.2.5.1 Subroutine Name: XRGDFM

1. Entry Point: XRGDFM

2. Purpose: To select a rigid format based on the SØL card in the Executive Control Deck.

3. Calling Sequence: CALL XRGDFM (NEWSØL, ØLDSØL, IAPP)

NEWSØL - Two-word array containing solution and subset numbers taken from SØL control card.

ØLDSØL - Two-word array containing solution and subset numbers taken from the Old Problem Tape if the problem is a restart. If not a restart, ØLDSØL = 0.

IAPP - Approach code (1 = HEAT, 2 = DISPL, 3 = DMAP) taken from the APP card in the Executive Control Deck.

4. Method: If the problem is being restarted, a check is made for a solution (Rigid Format) change. If the solution has been changed, a bit is set in table MEDMSK in named common block /XMDMSK/.

MODULE FUNCTIONAL DESCRIPTIONS

A check is made for a legal solution number, and, if acceptable, a branch is made on the solution number, and subroutine LDi (i = solution number) is called to create the DMAP and MED records for the XCSA table. XRGDFM then returns to the calling routine XCSA.

4.2.5.2 Subroutine Name: LDi, where i = solution number, $i = 01, 02, \dots, 12$.

1. Entry Point: LDi

2. Purpose: To write the DMAP sequence and MED records of the XCSA Executive Table for solution (Rigid Format) i (see XCSA Executive Control Table description, Section 2.4.2.5).

3. Calling Sequence: CALL LDi (SUBSET)

SUBSET - Solution subset number from the SØL control card.

4. Method: The packed DMAP program is generated, and then subroutine XSBSET is called to select the proper solution subset for the DMAP program by altering the IS array. Upon return from XSBSET the arrays IS, JNM and INM (see 4.2.6.2 below) are written on the New Problem Tape to complete the MED record for Executive Table XCSA. LDi then returns to calling routine.

4.2.5.3 Subroutine Name: XSBSET

1. Entry Point: XSBSET

2. Purpose: To eliminate DMAP instructions not belonging to the specified subset by altering the IS array.

3. Calling Sequence: CALL XSBSET (ID,NSS,SUBSET,IS,NDI,NWPI)

- ID - Table containing DMAP instruction numbers of those instructions that are not part of the specified subset.
- NSS - Number of subsets in table ID.
- SUBSET - Subset to be selected from table ID.
- IS - Module execution decision table.
- NDI - Number of DMAP instructions in DMAP program.
- NWPI - Number of words per IS entry.

4. Method: Table ID is searched and the proper subset is selected. Each DMAP instruction has a corresponding entry in table IS. If the IS entry for an instruction is zero, then the instruction is eliminated from the DMAP program. Therefore zeroing the IS entries of those instructions specified in table ID yields the proper subset. XSBSET then returns to the calling routine.

4.2.6 Design Requirements

4.2.6.1 Use of Open Core

Open core is used for GINØ buffers, for generating the XPTDIC Executive Table (see Section 2.4) on restarts, and for storing user generated DMAP programs. Named common block /XCSABF/ defines the beginning of open core for module XCSA. Since XPTDIC is not stored permanently in open core and because the use of open core to store a DMAP program and a call to LDi are mutually exclusive, the LDi subroutines can be originated for overlay purposes at the same location as /XCSABF/.

4.2.6.2 Restart Tables Initialized in the Routines

The following tables are initialized by the LDi (i = solution number) subroutine or its associated Block Data program and are used to aid module XGPI in restarting a problem which uses Rigid Format i.

1. IS - Module Execution Decision Table: This table when used in conjunction with table MEDMSK in named common block /XMDMSK/ will provide module XGPI with the information needed to decide whether or not to set the execute flag in an ØSCAR entry. Each DMAP instruction

MODULE FUNCTIONAL DESCRIPTIONS

in a Rigid Format has a corresponding entry in IS. An Entry in IS can be one to five words in length, and only bits 1 through 31 are used to form a truth table. Note that if an IS entry is zero, the corresponding DMAP instruction is unconditionally excluded from the Rigid Format DMAP program being compiled by module XGPI.

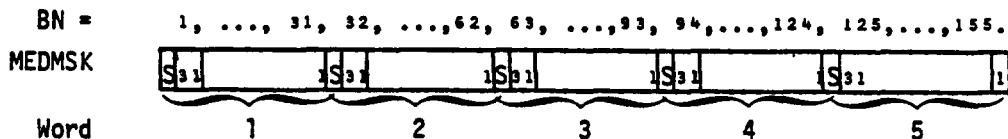
2. JNM - File Name Restart Table: The JNM table provides module XGPI with the capability of regenerating data blocks which are missing in the restart dictionary and which are needed to restart the problem. Note that the restart must be a modified restart. If a data block is missing, the JNM table will indicate which bit to set in table MEDMSK. MEDMSK is then used with IS to determine which DMAP modules must be re-executed in order to correctly regenerate the missing data block.

Sample JNM Entry:

Word 1	---	Data Block	---
2		Name (BCD)	
3	BN (integer)		

BN is the bit number of the bit which is to be set in table MEDMSK to regenerate the specified data block. The usable bits (bits 1-31) of MEDMSK are numbered sequentially starting from bit 31 of the first word. MEDMSK is five words long.

Example:



3. INM - Card Name Restart Table: When the problem is being restarted and input data (Bulk Data and/or Case Control Data) has been modified, table INM tells module XGPI whether or not the modifications affect the compilation of the DMAP program associated with the Rigid Format. Table MJCD in named common block /IFPX1/ and table MJMSK in named common block /IFPX0/ indicate which cards have been modified. If INM has an entry for a modified card, the INM entry will indicate which bit to set in table MEDMSK. MEDMSK is then used with IS to determine which DMAP modules must be re-executed.

EXECUTIVE PREFACE MODULE XCSA (EXECUTIVE CONTROL SECTION ANALYSIS)

Sample INM Entry:

Word 1	<div style="border: 1px solid black; padding: 5px; text-align: center;"> <div>--- Card Name (BCD) ---</div> <hr style="border-top: 1px dashed black;"/> <div>BN (integer)</div> </div>
2	
3	

BN is the number of the bit which is to be set in table MEDMSK if the associated card name has been modified. See sample JNM entry for further description of BN.

4. ID - Subset Table: DMAP instructions in a DMAP program are numbered sequentially starting with "BEGIN" as instruction number 1. Table ID contains the instruction numbers of those instructions that are not to be included in a subset.

Sample ID Entry:

Word 1	N_1 (integer)	<div style="display: inline-block; vertical-align: middle;"> $\left. \begin{array}{c} \vdots \\ N_1 \text{ words} \end{array} \right\}$ </div>	<div style="display: inline-block; vertical-align: middle;"> $\left. \begin{array}{c} \text{Repeat for all subsets} \end{array} \right\}$ </div>
2	I_j (integer)		
\vdots	\vdots		
$1+N_1$	I_k		
	\vdots		

N_1 = number of instructions to delete in subset 1. ($N_1 \geq 0$).

$I_j - I_k$ = DMAP instruction numbers of instructions to be deleted.

MODULE FUNCTIONAL DESCRIPTIONS

4.2.7 Diagnostic Messages

Every effort is made to get through module XCSA so that the modules IFP1 and IFP can process the Case Control Deck and the Bulk Data Deck. XCSA sets the NØGØ flag in named common block /SYSTEM/ according to the severity of the errors found.

NØGØ = 0 - no errors found.

1 - job will terminate after module XGPI.

2 - job will terminate after IFP modules.

3 - job terminates in XCSA.

See diagnostic message section of User's Manual (section 6.2) for a detailed discussion of XCSA error messages. XCSA messages include numbers 501 thru 526.

4.3 EXECUTIVE PREFACE MODULE IFP1 (INPUT FILE PROCESSOR, PART 1)

4.3.1 Entry Point: IFP1

4.3.2 Purpose

To process the Case Control Deck. See section 2.3 of the User's Manual for a discussion of the Case Control Deck.

4.3.3 Calling Sequence

CALL IFP1. IFP1, a Preface module, is called only by subroutine SEMINT.

4.3.4 Input Data

The input data consists of the Case Control Deck and the CASECC data block from the Old Problem Tape if the problem is a restart.

4.3.5 Output Data Blocks

CASECC - Case Control Data Table.

PCDB - Plot Control Data Table (for the structure plotter).

XYCDB - XY output Control Data Block.

Notes:

1. CASECC will always exist.
2. PCDB will exist only if a structure plotter package is included in the Case Control Deck.
3. XYCDB will exist only if a XYOUT plotter package is included in the Case Control Deck.

4.3.6 Method

The Case Control Cards are read and stored on a scratch file for later use. The title cards are abstracted to form page headings. Title card abstraction is stopped by a SYM, SUBCASE, SYMCØM or REPCASE card. IFP1 is stopped by a BEGIN BULK card.

MODULE FUNCTIONAL DESCRIPTIONS

The construction of CASECC is as follows:

1. The scratch tape is read one card at a time, and subroutine XRCARD is called to translate the card.
2. The first four characters beginning with a non-blank are identified in a key word table, and card type dependent routines are executed. (See Case Control Deck in User's Manual).
3. When "SUBCASE" type cards are encountered, a CASECC record is written out.
4. If the card OUTPUT (PLØT) is encountered, XRCARD images of succeeding cards are written on PCDB.
5. If the card OUTPUT (XYØUT) is encountered, IFPIXY processes the succeeding cards into the XYCDB.

The module conclusion is as follows:

1. A copy of CASECC is placed on the NPTP for use in restart.
2. If this run is a restart, IFPIB is called to analyze CASECC changes and set modify flags for later use in Executive Preface module XGPI (see section 4.7).

4.3.7 Subroutines

4.3.7.1 Subroutine Name: IFPIB

1. Entry Point: IFPIB
2. Purpose: To set modify flags for use in modified restart.
3. Calling Sequence: CALL IFPIB (ICASE, ØPTP, CASECC, IBUF1, IBUF2, LENCC)

ICASE - A two-dimensional array (LENCC,2) for storage of both copies of CASECC - array - input.

ØPTP - GINØ file name of the Old Problem Tape - BCD - input.

CASECC - GINØ file name of CASECC - BCD - input.

IBUF1}
IBUF2} - GINØ buffer pointers - integer - input.

LENCC - Row dimension of ICASE - integer - input.

EXECUTIVE PREFACE MODULE IFP1 (INPUT FILE PROCESSOR, PART 1)

COMMON/IFPX0/SPACE(3),NWØRDS

NWØRDS - Pointer into /IFPX0/ such that IFP1 modify flags are in SPACE(NWØRDS).

4. Method:

CASECC and the copy of CASECC on the ØPTP are compared according to the following scheme. The local array IWØRD classifies each word in CASECC into 0 and 1. 0 words: If the CASECC word is non-zero and IBIT is non-zero in this position, the IBIT bit is turned on in /IFPX0/. 1 word: If the CASECC word is different from the ØPTP word and IBIT is non-zero in this position, the IBIT is turned on in /IFPX0/.

IFP1B also determines the loop nature of the problem. The looping rules are as follows:

- a. The current problem will loop under the following conditions: SPC set changes; MPC set changes; direct input matrix changes; transfer function set changes; transient load changes; frequency set changes; differential stiffness coefficient set is greater than zero; and Piecewise Linear coefficient set is greater than zero. If any of the above conditions are met, LØØP\$ is turned on in /IFPX0/.
- b. The old problem might have been a looping problem if the above conditions were present in the ØPTP CASECC. If the old problem was a looping problem as determined in (a) and the number of records in CASECC changes, LØØP1\$ is set (this should force the re-execution of the entire loop).
- c. If the problem is not a looping problem, NØLØØP\$ is set.

4.3.7.2 Subroutine Name: IFP1C

1. Entry Point: IFP1C
2. Purpose: To construct set lists from SET cards.
3. Calling Sequence: CALL IFP1C (ISUB,IØ1,CØRE,SCR1,NWPC,ICC,NZ,THRU,NSET)
ISUB - 1 - master set. 2 - set belongs to a subcase - integer - input.
IØ1 - Pointer to storage for set in core - integer - input/output.
CØRE - Open core array.
SCR1 - GINØ file number of scratch file containing card images - integer - input.

MODULE FUNCTIONAL DESCRIPTIONS

- NWPC - Number of words per card - integer - input.
- ICC - Current line count of Case Control card echo - integer - input/output.
- NZ - Length of open core - integer - input/output.
- THRU - BCD value "THRU" - BCD - input.
- NSET - Number of sets found to current set - integer - input.

4.3.7.3 Subroutine Name: IFP1D

1. Entry Point: IFP1D
2. Purpose: To write user diagnostic messages from IFP1.
3. Calling Sequence: CALL IFP1D (MSGN)

MSGN - User message number - integer - input.

4.3.7.4 Subroutine Name: IFP1E

1. Entry Point: IFP1E
2. Purpose: To write out CASECC and update sets.
3. Calling Sequence: CALL IFP1E (CASE,ISUBC,SYMSEQ,NWDSC,I81)

CASE - Array containing Case Control record to be written out (CASE (LENCC,2)).

ISUBC - Five word BCD array containing current subcase number - BCD - output.

SYMSEQ - Symmetry coefficient array - real - input.

NWDSC - Pointer to beginning of set lists - integer - input/output.

I81 - Pointer to end of set list - integer - output.

4.3.7.5 Subroutine Name: IFP1F

1. Entry Point: IFP1F
2. Purpose: To find the first four characters beginning with a non-blank on one input card.

EXECUTIVE PREFACE MODULE IFP1 (INPUT FILE PROCESSOR, PART 1)

3. Calling Sequence: CALL IFP1F (\$n,IWØRD,IS,IBEN,II)

\$n - FØRTRAN statement number which defines the return to be taken if the entire card is blank.

IWØRD - First four characters beginning with a non-blank, left justified - BCD - output.

IS - Number of bits/character times (number of characters/word-1) - integer - input.

IBEN - Mask used to determine if character is blank 'b000' - input.

II - Pointer to word in which IWØRD begins - integer - output.

COMMON/IFP1X/CØRE(20)

IFP1X - 20-word array holding card image - BCD - input.

COMMON/IFP1A/

IFP1A - See /IFP1A/ description under Design Requirements (section 4.3.8).

4.3.7.6 Subroutine Name: IFP1G

1. Entry Point: IFP1G

2. Purpose: To find an equal sign and copy the remainder of the data into a specified arrays

3. Calling Sequence: CALL IFP1G (ITYPE,CASE,ISUB1)

ITYPE - Indicates area in which to store data.

- | | |
|---------------------------|---------------|
| 1. TITLE | } of /ØUTPUT/ |
| 2. SUBTITLE | |
| 3. LABEL | |
| 4. HEAD1 | |
| 5. HEAD2 | |
| 6. HEAD3 | |
| 7. PLØTID | |
| 8. First 32 words of CASE | |

- integer - input.

CASE - Case control array (132,2) unless ITYPE = 8, when it may be only 32 word array.

ISUB1 - Subcase number -1 or 2 of CASE array.

COMMON/IFP1X/CØRE(20)

IFP1X - 20-word array holding card image.

MODULE FUNCTIONAL DESCRIPTIONS

COMMON/OUTPUT/

OUTPUT - Output common block - holds BCD titles for NASTRAN pages.

COMMON/IFP1A/

IFP1A - See /IFP1A/ description under Design Requirements (section 4.3.8).

4.3.7.7 Subroutine Name: SWSRT

1. Entry Point: SWSRT
2. Purpose: To check set lists for duplicates and overlapping intervals. SWSRT also sorts lists into increasing order.
3. Calling Sequence: CALL SWSRT (LIST,ISTØR,NLIST)

LIST - Array of set members.

ISTØR - Scratch space of length NLIST.

NLIST - Number of members in LIST.

4.3.7.8 Subroutine Name: IFP1XY

1. Entry Point: IFP1XY
2. Purpose: To construct the XYCDB data block.
3. Calling Sequence: CALL IFP1XY (FLAG)

FLAG - 0, first entry for initialization; 1, data entry; -1, last entry. FLAG is set to 1 on return from the first entry - integer - input/output.

COMMON/IFP1X/CARD(20), CARD1(20)

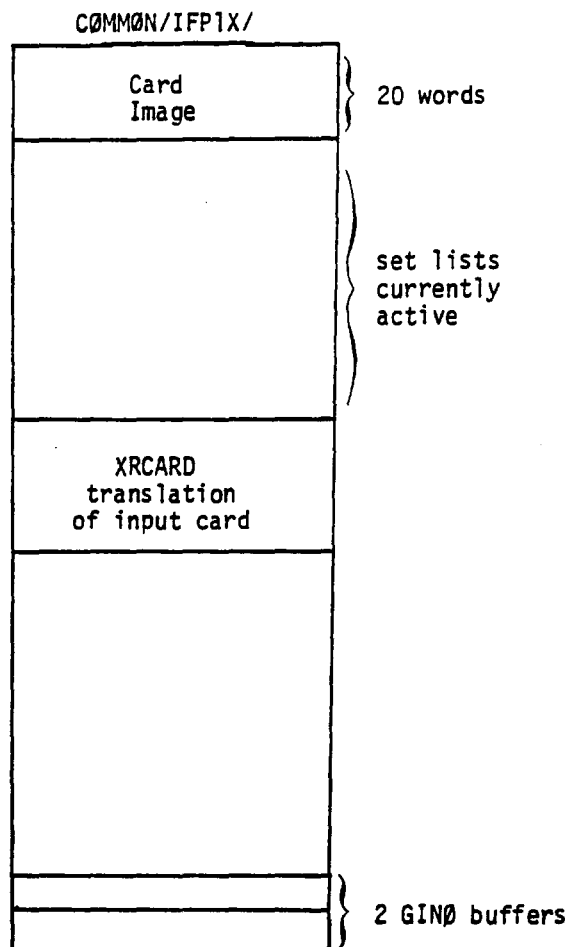
CARD - Contains card data as read from the input file.

CARD1 - Contains XRCARD translation of CARD unless CARD(1) contains 'XTIT', 'YTIT', 'TCUR', 'YTTI' or 'YBTI'. In this case CARD1 (20) contains BCD data occurring after the equal sign.

4.3.8 Design Requirements

1. One scratch file.
2. Open core at /IFP1X/.

EXECUTIVE PREFACE MODULE IFP1 (INPUT FILE PROCESSOR, PART 1)



3. Common block IFP1A.

<u>Name</u>	<u>Length</u>	<u>Meaning</u>	<u>Initialized to</u>
NAME	2	Name of data block for error messages.	CASE CC
SUBC	1	Subcase key word	SUBC
SET	1	Set key word	SETb
SYMS	1	Symmetry sequence key word	SYMS
TSTE	1	Time step card selection key word	TSTE
LABE	1	Label key word	LABE
SUBT	1	Subtitle key word	SUBT
SCR1	1	GINØ file number of scratch file	301
CASECC	1	GINØ file name of CASECC	CASE
BLANK	1	Blank word	bbbb

MODULE FUNCTIONAL DESCRIPTIONS

<u>Name</u>	<u>Length</u>	<u>Meaning</u>	<u>Initialized to</u>
CARD	1	Heading word card	CARD
CØUN	1}	Heading word CØUNT	CØUN
T	1}		T
BEGI	1	Begin bulk key word	BEGI
TITL	1	Title key word	TITL
CASEN	11	Case Control page heading	Case Control Deck echo
SPCF	1	Forces of constraint key word	SPCF
VELØ	1	Velocity key word	VELØ
ACCE	1	Acceleration key word	ACCE
ELFØ	1	Element forces key word	ELFØ
STRE	1	Element stress key word	STRE
DISP	1	Displacement key word	DISP
ØUTP	1	Øutput key word	ØUTP
SYM	1	Symmetry subcase key word	SYM
FREQ	1	Frequency set key word	FREQ
DLØA	1	Dynamic loading key word	DLØA
TEMP	1	Temperature field key word	TEMP
DEFØ	1	Deformation set key word	DEFØ
TIME	1	Time key word	TIME
SPC	1	Single-point constraint set key word	SPC
MAXL	1	Maximum number of output lines key word	MAXL
IC	1	Initial condition set selection id	IC
METH	1	Real eigenvalue or buckling method selection key word	METH
LØAD	1	Load set selection key word	LØAD
MPC	1	Multipoint set selection key word	MPCb
STIF	1	Stiffness thermal field key word	STIF
ALL	1	ALL key word	ALLb
THRU	1	THRU key word	THRU
SØRT	1	Sorted echo key word	SØRT
UNSØ	1	Unsorted echo key word	UNSØ

EXECUTIVE PREFACE MODULE IFP1 (INPUT FILE PROCESSOR, PART 1)

<u>Name</u>	<u>Length</u>	<u>Meaning</u>	<u>Initiated to</u>
ECHØ	1	Bulk data echo key word	ECHØ
PLØT	1	Plot key word	PLØT
MØDE	1	Modes key word	MØDE
PUNC	1	Punch key word	PUNC
PRIN	1	Print key word	PRIN
NWPC	1	Number of words per card	20
NCPW	1	Number of characters per word (NASTRAN)	4
BØTH	1	Echo-sorted and unsorted	BØTH
NØNE	1	None key word	NØNE
PCDB	1	Plot control data GINØ file name	PCDB
NAME	2	GINØ error message for PCDB file	PCDB bbbb
VECT	1	Alternate displacement key word	VECT
SYMC	1	Symcom key word	SYMC
EQUAL1	1	Equal sign left adjusted	=bbb
NMØDES	1	Value of modes card	1
IBØB	1	Structure plot flag 0, not currently in structure plot mode 1, in structure plot mode	0
IEND	1		0
ISYCMC	1	Symcom flag	0
LØADN	1	Current Subcase ID number	1
IØUT2	1		0
ICC	1	Printed card count	1
NSET	1	Number of current set lists	0
NSYM	1	Number of 'SYM' subcases	1
INØMØR	1	Flag to turn off Title card search	0
NPTP	1	GINØ file name of the New Problem Tape	NPTP
ØPTP	1	GINØ file name of the Old Problem Tape	ØPTP
DØL	1	Dollar sign	\$bbbb
ZZZZBB	1	Hollerith zeros	0000bb

MODULE FUNCTIONAL DESCRIPTIONS

<u>Name</u>	<u>Length</u>	<u>Meaning</u>	<u>Initiated to</u>
ISTR	1	Storage flag for IFPIG titles	1
ISUB	1	Subcase or master CASECC pointer	1
K2PP	1}	Key words for direct input matrix selection	K2PP
B2PP			B2PP
M2PP			M2PP
DSCØ	1	Key word for differential stiffness set selection	DSCØ
REPC	1	Key word for repeat subcase subcase	REPC
LENCC	1	Length of Case Control Record	166
LINE	1	Key word for LINE/page count	LINE
ØM	1	Word to distinguish between SUBCØM SUBCASE	ØMbb
TFL	1	Key word for transfer function set selection	TFL
DEFA	1	Key word for default specification	DEFA
ELST	1	Key word for element stress set selection	ELST
MAT	1	Key word for thermal material set selection	MATE
ØFRE	1	Key word for output frequency set selection	ØFRE
IMAG	1	Key word for real/imaginary printout	IMAG
PHAS	1	Key word for magnitude/phase printout	PHAS
REAL	1	Key word for real or real/imaginary printout	REAL
CMET	1	Key word for complex eigenvalue set selection	CMET
SDAM	1	Key word for Structural Damping Table for use in modal formulation	SDAM
INER	1	Key word for Inertia Relief Element set selection	INER
ADIS	1	Key word for solution set displacement selection	SDIS
AVEL	1	Key word for solution set velocity selection	SVEL
AACC	1	Key word for solution set acceleration selection	SACC
NØNL	1	Key word for non-linear load set selection	NØNL
CØNF	1	Not used	
XYPL	1	Key word for XYPLØT packet delimiter	XYPL
PLCØ	1	Key word for Piecewise Linear set selection	PLCØ

EXECUTIVE PREFACE MODULE IFP1 (INPUT FILE PROCESSOR, PART 1)

<u>Name</u>	<u>Length</u>	<u>Meaning</u>	<u>Initiated to</u>
AXIS	1	Key word for selection of Axis symmetric boundary condition	AXIS
NLLØ	1	Key word for non-linear output set selection	NLLØ
DELE	1	Key word for element deletion set selection	DELE
XYCB	1	GINØ file name of XY control data block	XYCB
ØNEB	1	BCD one	1bbb
HARM	1	Key word for harmonic output control	HARM
SINE	1	Key word for sine boundary conditions	SINE
CØSI	1	Key word for cosine boundary conditions	CØSI
FLUID	1	Key word for fluid boundary conditions	FLUI
SUBS	1	Key word for SUBSEQ	SUBS
AVEC	1	Key word for solution set vector output	SVEC
FØRC	1	Not used	
RAND	1	Key word for random set selection	RAND
XYØU	1	Key word for XYPLØT packet delimiter	XYØU
ØLØA	1	Key word for output load set selection	ØLØA
PLT1	1	GINØ file name of BCD plot tape	PLT1
PLT2	1	GINØ file name of binary plot tape	PLT2
XTIT	1	Key words for XY output titles	XTIT
YTIT	1		YTIT
TCUR	1		TCUR
YTTI	1		YTTI
YBTI	1		YBTI
IBEN		Right shifted blank '000b'	----
EQUAL		Right shifted equal '000='	----
PRES	1	Alternate displacement key word	PRES
TEMP	1	Alternate displacement key word	TEMP

4. Interface with /SYSTEM/ (See Section 2.4).

IFP1 can set the following cells of SYSTEM:

- NØGØ - (NØGØ flag). If a fatal error is detected.
- NLPP - (Number of lines per page). If a LINE card is supplied by the user.
- STFTEM - (Material Temperature Set ID). If a TEMP(MATE) card is supplied.

MODULE FUNCTIONAL DESCRIPTIONS

d. ECHØ - (Echo flag). If an ECHØ request is made.

5. Interface with /ØUTPUT/.

IFP1 supplies the problem title, subtitle, and label as well as the Plot ID.

4.3.9 Diagnostic Messages

IFP1 makes every attempt to process the entire Case Control Deck so that the complete Preface will run. Hence all fatal messages only cause the NØGØ flag to turn on.

IFP1 causes messages 601-699. For the exact nature of these messages, refer to the Diagnostic Message section of the User's Manual.

EXECUTIVE PREFACE MODULE XSØRT (EXECUTIVE BULK DATA CARD SØRT)

4.4 EXECUTIVE PREFACE MODULE XSØRT (EXECUTIVE BULK DATA CARD SØRT)

4.4.1 Entry Point: XSØRT

4.4.2 Purpose

The function of XSØRT is to prepare a file on the New Problem Tape containing the sorted bulk data. The operation of XSØRT is influenced by the type of run. If a cold start, the bulk data is read from the system input stream (or the User's Master File), sorted and written on the New Problem Tape. If an unmodified restart, the bulk data is copied from the Old Problem Tape onto the New Problem Tape. If a modified restart, the bulk data is read from the Old Problem Tape, and cards are deleted and/or added in accordance with cards in the system input stream. Additionally, flags are set within restart tables for each card type changed in any way. Again, the sorted bulk data is written onto the New Problem Tape. A print of the unsorted and/or sorted bulk data is made on request. XSØRT processes all data cards between the BEGIN BULK and ENDDATA cards in the input stream. Both cards must be present to properly bracket the NASTRAN Bulk Data Deck.

4.4.3 Calling Sequence

CALL XSØRT. XSØRT, a Preface module, is called only by the Preface driver, SEMINT.

4.4.4 Method

If the input is to be from a User Master File, XSØRT begins by positioning the file to the beginning of the proper subset of bulk data cards. INITCØ is then called to initialize machine dependent masks and constants. The open core below XSØRT (/ESØRT/) is divided into 5 GINØ buffers and a work buffer. This work buffer will contain each data card and a chaining pointer to indicate its sorted position. That is, the cards will be placed into the work buffer in the same order as read, but their sorted order will be shown by a chaining word with each card pointing to the position of the next card in alphanumeric sort. If the work buffer is unable to hold all of the bulk data cards, each subset that fills the buffer is unchained and written in sorted order onto a scratch file. This writing onto a scratch file frees the work buffer for another subset of data cards.

MODULE FUNCTIONAL DESCRIPTIONS

Three scratch files may become involved in sorting a large number of bulk data cards. After the first two scratches are filled with sorted subsets, they are merged, while maintaining the sorted order, onto a third scratch. From this point, after each new subset is written onto a scratch, it is merged with the scratch containing all previous subsets. As an example, assume three scratches are named A, B, and C. Scratch A is written with the first subset of data from a filled work buffer. Scratch B is written with the second subset. Scratch A and B are then merged to form scratch C. This frees scratch A and B. Scratch A is then written with the third subset of data. A and C are merged to form a new B. A is then written with the fourth subset. A and B are merged to form a new C. This process continues until all bulk data has been sorted. Following the final merge, one of the scratch files will contain all of the sorted bulk data cards.

As the sort and merge operations are being performed, any continuation cards or delete cards encountered are written onto separate holding files. After all data cards in the input stream have been processed, each of these holding files is processed. The delete card values are placed in ascending order and any overlaps or redundancies are removed. The continuation cards are checked and flagged for duplication and an in-core dictionary of their connection words is formed.

XSORT may now make a pass through the scratch file containing all of the sorted bulk data cards within the input stream. During this pass, the User Master File (UMF) or the Old Problem Tape (OPTP) data cards are merged with those from the input stream. Both the UMF and OPTP data cards were properly sorted during their preparation. As this merge progresses, any data cards designated for removal by delete control cards are discarded. If the NASTRAN run for which XSORT is operating is a restart, all data cards within the input stream plus any deleted from the OPTP will cause data card type flags to be set within restart tables. This entire pass is not performed if the run does not require either a UMF or OPTP.

Now a final pass of the resulting sorted data is made to introduce any continuation cards and write the completed Bulk Data Deck onto the New Problem Tape. The continuation cards are connected to the sorted data cards by matching connection words (duplicate continuation cards are ignored). Continuation cards can in no way affect the sorted order. If a print of the resulting sorted deck is requested, it is performed during this pass.

After all the cards are sorted, any continuation cards that are unused (because they are

EXECUTIVE PREFACE MODULE XSORT (EXECUTIVE BULK DATA CARD SORT)

parentless and/or are duplicates) are properly identified. Also, those continuation cards that are valid, but cannot yet be processed because of errors on other related continuation cards, are also properly identified.

During any sort collation the data cards are ordered by comparing half-fields from left to right. Each bulk data card may contain up to ten, eight column (character) fields. Because of computer word size constraints, each data card is stored into twenty memory words, four characters (half a card field) per word. Sorting proceeds by comparing the first words (4 characters) from each card are compared, and so on, until an order is established or total duplication is determined. Each field (8 characters) is left (BCD) or right (integer) justified prior to sorting to eliminate leading or trailing blanks. The characters within the first field of each card are converted to a special internal character set prior to comparing to eliminate machine dependent collation sequences which might order the same cards differently on different machines. This internal set forces the collation order to be ascending from blank through all numbers then all letters. A flowchart is given in Figure 1.

4.4.5 Subroutines

In the following, note that XRECPS, RPAGE, INITCØ, XFADJ, XBCDBI, XPRETY, CRDFLG, EXTINT, INTEXT are secondary entry points in XRECPS.

4.4.5.1 Subroutine Name: XRECPS

1. Entry Point: XRECPS
2. Purpose: Positions the continuation card file to the proper record (card image) as determined from the in-core continuation card dictionary.
3. Calling Sequence: CALL XRECPS (NEW, ØLD) where:
NEW = the file position being requested
ØLD = the file position last requested. Both arguments are integer record numbers.

4.4.5.2 Subroutine Name: RPAGE

1. Entry Point: RPAGE
2. Purpose: Counts output print lines for XSORT, and performs the necessary interface with the system subroutine PAGE.
3. Calling Sequence: CALL RPAGE (NLINE) where:
NLINE = integer number of lines being output. If $NLINE \geq 100$ a page eject is forced and the line count is set to $NLINE - 100$.

MODULE FUNCTIONAL DESCRIPTIONS

4.4.5.3 Subroutine Name: INITCØ

1. Entry Point: INITCØ
2. Purpose: Initializes machine dependent masks and constants within XSØRT.
3. Calling Sequence: CALL INITCØ

4.4.5.4 Subroutine Name: XFADJ

1. Entry Point: XFADJ
2. Purpose: Adjusts four character fields, left or right, two or four fields at a time. If the fields contain only integers, the shift is right, otherwise the shift will be left. This routine determines only the direction of shift required. Actual shifting is performed by XFADJ1.
3. Calling Sequence: CALL XFADJ (BUF,SD,K) where:
BUF = field array to be shifted
$$SD = \begin{cases} 0, & \text{shift two fields at a time} \\ 1, & \text{shift four fields at a time} \end{cases}$$
$$K = \begin{cases} 0, & \text{returned if right shift was done.} \\ 1, & \text{returned if left shift was done.} \end{cases}$$

4.4.5.5 Subroutine Name: XBCDBI

1. Entry Point: XBCDBI
2. Purpose: Converts two, four character BCD integer fields (right adjusted in the left most four characters of the computer word) into a single binary integer (right adjusted in the second of the two input words).
3. Calling Sequence: CALL XBCDBI (BUF) where:
BUF = two word array to be converted.

4.4.5.6 Subroutine Name: XPRETY

1. Entry Point: XPRETY
2. Purpose: "Pretties-up" printed output by left adjusting all fields to eliminate any leading zeros introduced when integer fields are right adjusted.
3. Calling Sequence: CALL XPRETY (BUF) where:
BUF = card image array.

EXECUTIVE PREFACE MODULE XSORT (EXECUTIVE BULK DATA CARD SORT)

4.4.5.7 Subroutine Name: Subroutine Name: CRDFLG

1. Entry Point: CRDFLG
2. Purpose: Sets the card type flags within the restart tables.
3. Calling Sequence: CALL CRDFLG (CARD) where:
CARD = first of two word card type field.

4.4.5.8 Subroutine Name: EXTINT

1. Entry Point: EXTINT
2. Purpose: Converts card type field from the machine dependent character code to an internal machine independent code.
3. Calling Sequence: CALL EXTINT (CTYBF) where:
CTYBF = first of two word card type field.

4.4.5.9 Subroutine Name: INTEXT

1. Entry Point: INTEXT
2. Purpose: Converts the card type field from an internal machine independent code to the machine dependent character code.
3. Calling Sequence: CALL INTEXT (CTYBF) where:
CTYBF = first of two word card type field.

4.4.5.10 Subroutine Name: XFADJ1

1. Entry Point: XFACJ1
2. Purpose: Adjust four character fields left or right, two or four fields at a time.
This routine performs actual shifting with the direction of shift controlled through the calling sequence. (Note entry point XFADJ).
3. Calling Sequence: CALL XFADJ1 (BUF,SHIFT,SD) where:
BUF = Field Array to be shifted.
SHIFT = Function LSHIFT or RSHIFT.
SD = $\begin{cases} 0, & \text{shift two fields at a time.} \\ 1, & \text{shift four fields at a time.} \end{cases}$

4.4.5.11 Function Name: ISFT

1. Entry Point: ISFT
2. Purpose: Performs special shifting functions for subroutine XFAJ1.

MODULE FUNCTIONAL DESCRIPTIONS

3. Calling Sequence: CALL ISFT

Result = ISFT(BUF,SFTCNT,J) where:

BUF = Word to be shifted.

SFTCNT = Bits to be shifted.

J = Shift direction control; 3 = right, 4 = left.

4.4.6 Design Requirements

1. Data cards operated upon by XSORT must conform to the NASTRAN format for bulk data cards (ten, eight character fields per card). See section 2 of the User's Manual for details.
2. Data cards must contain only valid BCD key punch codes or blanks. Non-standard multi-punched code (e.g., some IBM EBCDIC) will cause unpredictable results.
3. XSORT requires sufficient open core to contain five GINØ buffers and a work buffer for at least ten data cards. (Each data card requires twenty-one core locations). Sort efficiency increases in proportion to the size of the work buffer.
4. The continuation card dictionary must fit into the core work buffer during the final pass. Each continuation card requires four dictionary locations.
5. XSORT logic is biased toward input that is already sorted. That is, the program will operate at a much greater speed if verifying a sort rather than producing a sort.

4.4.7 Diagnostic Messages

XSORT can produce two categories of diagnostic messages. The first are termed USER messages and deal with bulk data card errors. The second are termed SYSTEM messages which are generally fatal in nature and indicate serious I/O malfunctions.

XSORT message numbers includes 201 through 216. These messages are listed and explained in Section 6.2.1 of the User's Manual.

EXECUTIVE PREFACE MODULE XSORT (EXECUTIVE BULK DATA CARD SORT)

(THIS PAGE IS LEFT BLANK INTENTIONALLY)

MODULE FUNCTIONAL DESCRIPTIONS

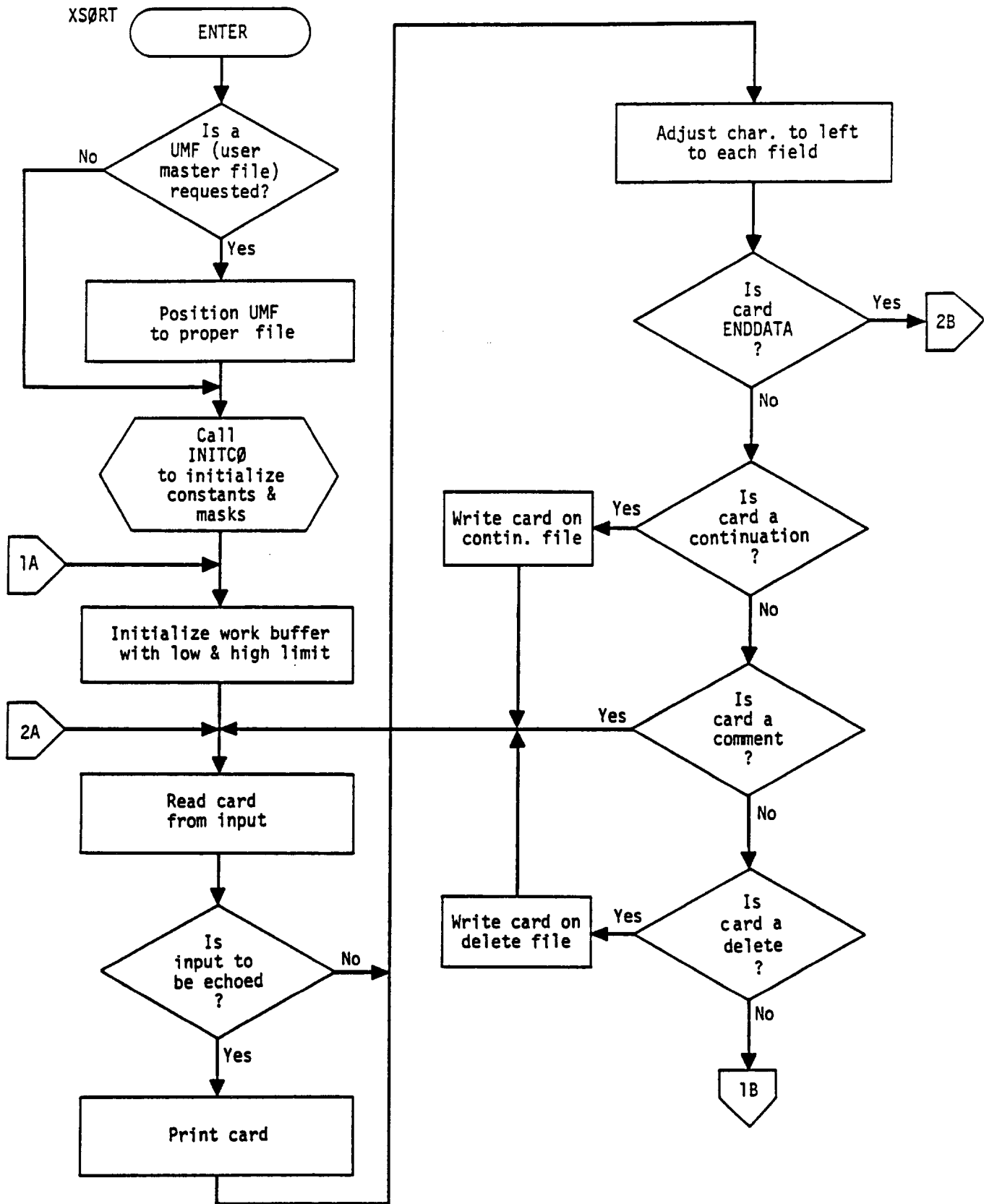


Figure 1.(a) Flowchart for module XSORT.

EXECUTIVE PREFACE MODULE XSORT (EXECUTIVE BULK DATA CARD SORT)

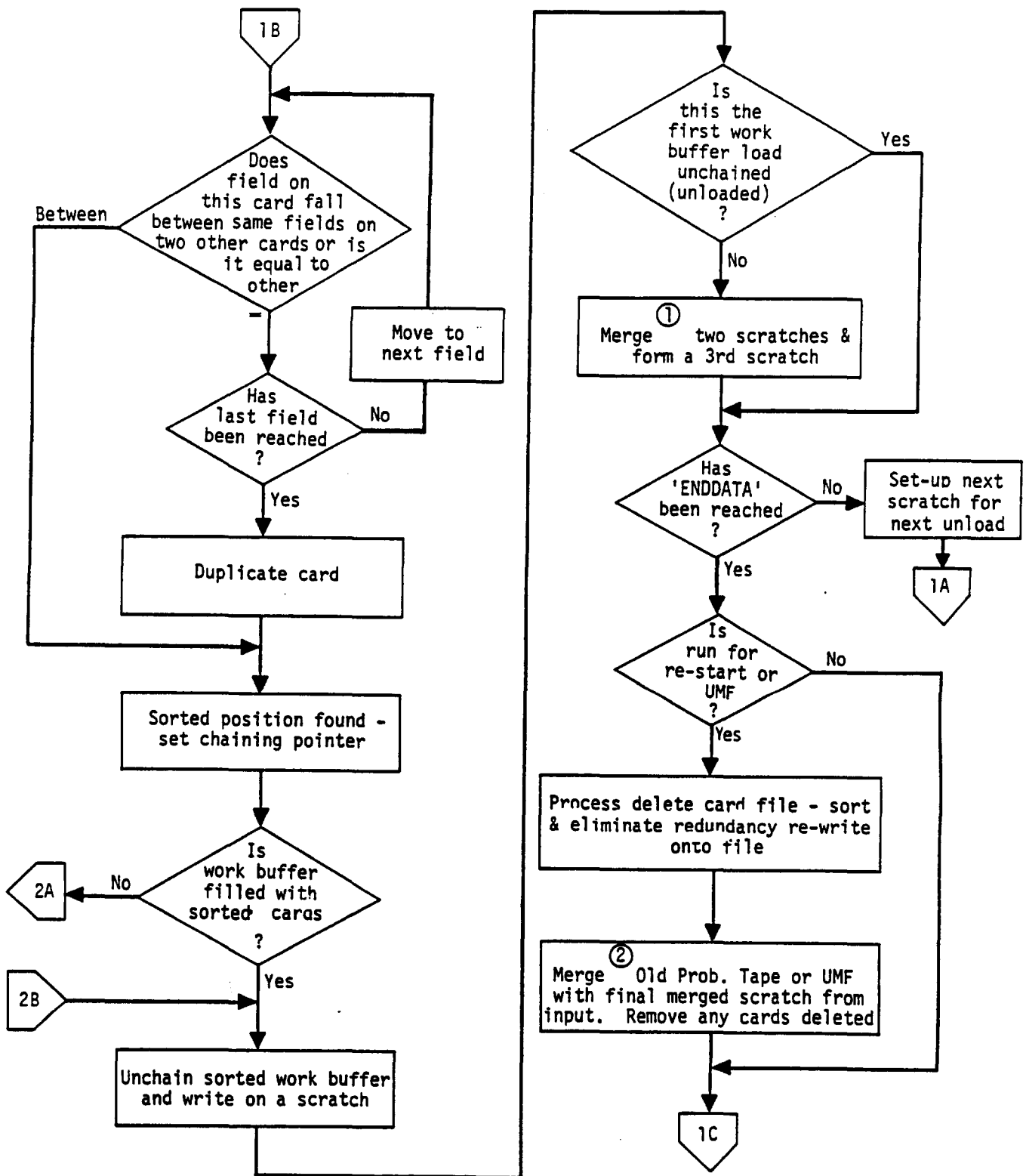


Figure 1.(b) Flowchart for module XSORT.

MODULE FUNCTIONAL DESCRIPTIONS

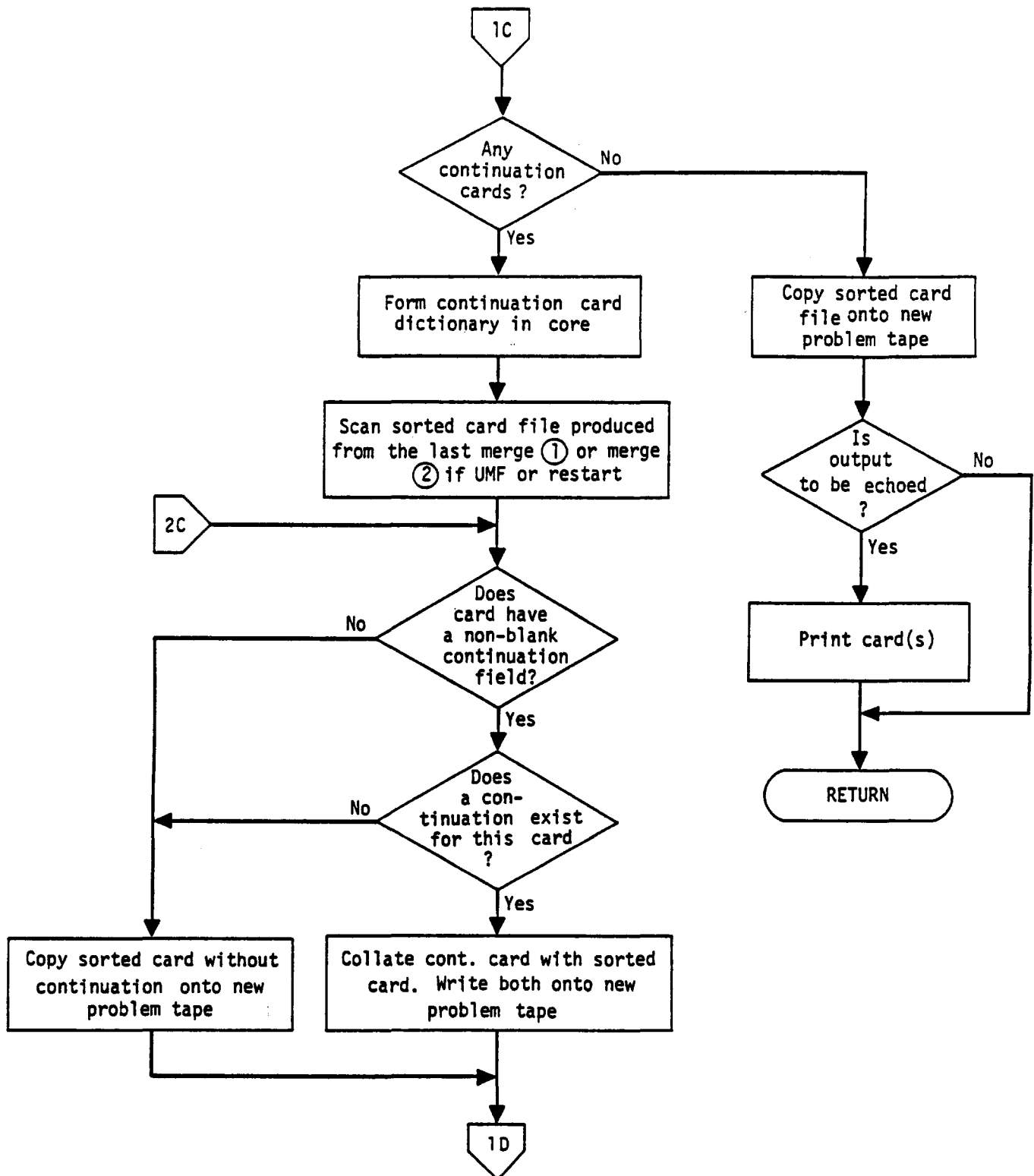


Figure 1.(c) Flowchart for module XSORT.

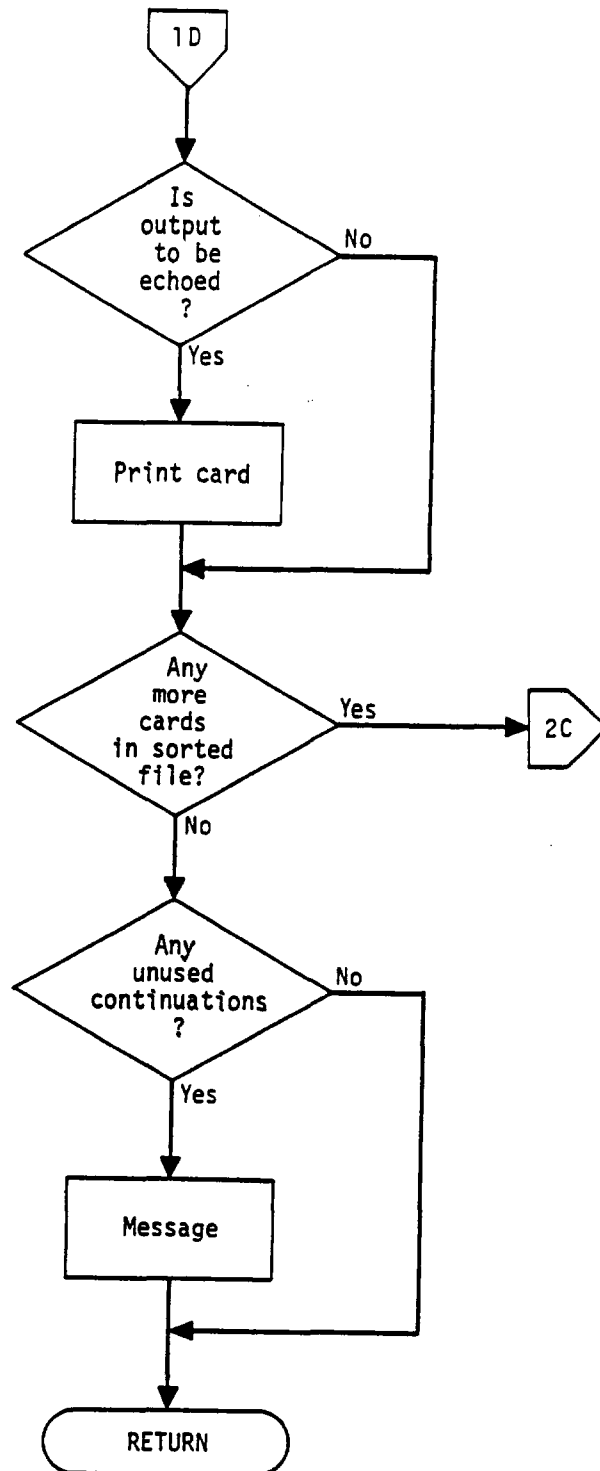


Figure 1.(d) Flowchart for module XSØRT.

EXECUTIVE PREFACE MODULE IFP (INPUT FILE PROCESSOR)

4.5 EXECUTIVE PREFACE MODULE IFP (INPUT FILE PROCESSOR)

4.5.1 Entry Point: IFP

4.5.2 Purpose

To process the Bulk Data Deck sorted by Executive Preface module, XSORT. This task is accomplished as follows: 1) the sorted Bulk Data Deck is read from the New Problem Tape (NPTP) card-by-card; 2) the contents of each field of each card are validated (see section 2.4 of the User's Manual for detailed descriptions of each bulk data card); 3) card images or modified card images are written on data blocks on the NPTP or the Data Pool File (see section 2.3.2 of the Programmer's Manual for details on the formats of these data blocks).

4.5.3 Calling Sequence

CALL IFP. IFP, an Executive Preface module, is called only by the Preface driver, subroutine SEMINT.

4.5.4 Input

The input to IFP consists of the Bulk Data Deck sorted by Executive Preface module XSORT.

4.5.5 Output

The output of IFP consists of: 1) data blocks used in Rigid Formats; 2) the AXIC data block, which is processed by Executive Preface Modules IFP3, IFP4 and IFP5 and is present only if the NASTRAN run is a conical shell (a unique structural element) problem, a hydroelastic problem, or an acoustic cavity problem; 3) the PVT Executive table, which contains the names and values of all DMAP parameters input by means of the PARAM bulk data card, and which is written on a scratch file to be processed by Executive Preface module XGPI; and 4) DMI's (Direct Matrix Inputs) and DTI's (Direct Table Inputs), each of which is written on the Data Pool File as a data block and is indistinguishable from any matrix data block or table data block pooled by the Executive Segment File Allocator (XSFA) module.

MODULE FUNCTIONAL DESCRIPTIONS

4.5.5.1 Output Data Blocks Used in Rigid Formats

- GEØM1 - Grid point, coordinate system, and sequence data.
- GEØM2 - Element connection data.
- GEØM3 - Static loads and temperature data.
- GEØM4 - Displacement set definitions data.
- EPT - Element Property Table.
- MPT - Material Property Table.
- DIT - Direct Input Tables.
- EDT - Element Deformation Table.
- DYNAMICS - Collection of bulk data cards for a dynamics problem.
- MATPØØL - Data block containing matrices input on DMIG bulk data cards.

Note: Do not confuse the DTI (Direct Table Input) bulk data card and the DIT (Direct Input Table) data block.

4.5.6 Method

4.5.6.1 General Comments

The bulk data cards processed by IFP are classified into five categories. Listed below is a brief explanation of each with a few examples.

1. Closed End Cards (Fixed Length Card)

Cards such as CQUAD2 and PRØD go through all the standard bulk data card checks (see 4.5.6.2) before being processed by the card dependent subroutines within IFP, (IFSIP, i = 1, 2, 3, 4, 5). These closed end cards are output to one of the standard GINØ files.

2. Open Ended Cards (Variable Length Cards)

In cards such as SPC1 and PLFACT, since the length and therefore the formats are not known, the bulk data checks using the data initialized in the block data subprograms must be made in the card dependent subroutines. Also, since the length is not known, a flag is placed at the end of the information for that card before being written on the file in order that routines reading an open ended card will be signaled as to an end-of-card condition.

EXECUTIVE PREFACE MODULE IFP (INPUT FILE PROCESSOR)

3. GRDSET and BARØR Cards

Special cards such as GRDSET and BARØR are not output to a GINØ data block, but are stored in local variables, and provide default values for the GRID and CBAR cards.

4. DMI and DTI Cards

DMI and DTI cards are unique in the manner in which they are used by the user and processed by IFP. The DMI card enables the user to define matrix data blocks directly, while the DTI card gives the user the capability to input his own table data blocks directly. The user must write a DMAP sequence or use the ALTER feature - see section 2.2 of the User's Manual - in the Executive Control Deck to alter the Rigid Format chosen in order to use the DMI or DTI feature since he is defining his own data blocks. Both DMI and DTI cards are written directly onto the Data Pool File.

5. PARAM Cards

PARAM bulk data cards are stored in open core by IFP until the entire Bulk Data Deck has been processed. PARAM cards are then written as the PVT (Parameter Value Table) on a scratch file for subsequent processing by the Executive Preface module XGPI.

4.5.6.2 Card Processing

1. IFP searches the NPTP for the Bulk Data Deck and extracts it in 20 word (one physical card) segments. Each card is passed to subroutine RCARD, which takes the BCD card images and converts the fields thereon to values, and identifies the values as to type: blank data field, integer data field, real single precision data field, BCD data field, real double precision data field or a data field which is in error.

IFP always has two physical bulk data cards in internal storage areas: the "current" card and the "next" card. M is the local FØRTRAN array where the values of the current bulk data card are located, and M1 the local FØRTRAN array where the values of the next card are located. After the current card is processed, the data in M1 are transferred into M, and new card values for M1 are input from the NPTP. When the values from M1 are transferred to M, the first two words (the card mnemonic) are stripped off. M1(3) is stored in M(1), M1(4) is stored in M(2), and so on.

MODULE FUNCTIONAL DESCRIPTIONS

2. /IFPX1/ is referenced to verify the admissibility of the name (mnemonic) of the particular card taken from the NPTP.

3. The approach acceptability flag is checked. This flag is defined as follows:

- 0 = OK for any approach;
- 1 = Not used by displacement approach;
- 2 = Illegal for displacement approach.

The approach flag (DISPL, DMAP) is found in /SYSTEM/.

4. The proper output files are established. See Table 1 or Table 2 for the output file on which the various bulk data cards will reside.

5. Uniqueness flags, which reside in COMMON/IFPX5/, are defined for each card type as follows:

- 0 - No check is made;
- 1 - A check is made;
- 2 - A special check is made.

For example, on the bulk data card CØNRØD, field 2 is the EID, and it must be unique with respect to all other CØNRØD EID's.

6. The next physical card is read from the NPTP. This will be the next card to be processed.

7. A check for too many continuation cards is made. This check is made on fixed length cards only.

8. A check is made for the minimum and maximum number of words for a logical bulk data card.

9. A check for the proper types of values for the fields on a card is made by referencing /IFPX7/, which contains format codes for each card type as follows:

- 0 = Blank
- 1 = Integer
- 2 = Real
- 3 = BCD
- 4 = Double Precision
- 5 = Anything.

EXECUTIVE PREFACE MODULE IFP (INPUT FILE PROCESSOR)

When RCARD passes format values to IFP, a format code of 0 will override this check. In the card dependent code check (step 10), the value will be looked at to see if it is in error.

10. An auxiliary subroutine IFSiP, $i = 1, 2, 3, 4, 5$, is called to execute card dependent code.

11. If the input card passes the tests in the card dependent code, the data are written on the appropriate GINØ output file.

4.5.6.3 Module Conclusion

When the sorted Bulk Data Deck has been exhausted, the following steps are carried out.

1. The appropriate trailer codes are written for each data block. For listings of trailer information reference section 2.3.2.

2. The PVT is written on a scratch file.

3. Restart flags are set in /IFPX0/.

4.5.7 Subroutines

IFP uses the utility routine RCARD described in section 3.4.20.

4.5.7.1 Block Data Subprogram: IFX1BD

Purpose: To initialize /IFPX1/, which is used by IFP to validate card names. All bulk data card names must appear in this table.

4.5.7.2 Block Data Subprogram: IFX2BD

Purpose: To initialize /IFPX2/. This table contains two words per entry (two words per card type): the first gives the GINØ output file number, and the second gives the approach acceptability flag.

4.5.7.3 Block Data Subprogram: IFX3BD

Purpose: To initialize /IFPX3/. This table contains two words per entry (two words per card type): the first word is used as the Conical Shell Problem flag, and the second word is used internally to store the number of words to be output to the GINØ output file.

MODULE FUNCTIONAL DESCRIPTIONS

4.5.7.4 Block Data Subprogram: IFX4BD

Purpose: To initialize /IFPX4/. This table contains two words per entry (two words per card type): the first is the minimum number of words allowable, the second is the maximum number of words allowable. The first word of an entry being negative implies the card is open ended.

4.5.7.5 Block Data Subprogram: IFX5BD

Purpose: To initialize /IFPX5/. This table contains two words per entry (two words per card type): the first is a pointer into /IFPX7/, the second is the field 2 uniqueness check flag.

4.5.7.6 Block Data Subprogram: IFX6BD

Purpose: To initialize /IFPX6/. This table contains two words per entry (two words per card type): the first is header word 1 (card type), the second is header word 2 (trailer bit position) of the three word header information of each logical record, which corresponds to all the data of particular bulk data card type. See section 2.3.2 for details.

4.5.7.7 Block Data Subprogram: IFX7BD

Purpose: To initialize /IFPX7/. Each entry contains the admissible sequence of format codes for that card type (see step 9 in section 4.5.6.2 above).

4.5.7.8 Subroutine Name: IFSiP, $i = 1, 2, 3, 4, 5$

1. Entry Point: IFSiP, $i = 1, 2, 3, 4, 5$
2. Purpose: These are the four subroutines that the module driver IFP calls to execute card dependent code.
3. Calling Sequence: CALL IFSiP (n_1, n_2, n_3)
 - n_1 - FORTRAN statement number defining the return taken in the event of a format or data error.
 - n_2 - FORTRAN statement number defining the return taken when local variables are set to provide default values for appropriate cards.
 - n_3 - FORTRAN statement number defining the return taken in the event of a data error.

EXECUTIVE PREFACE MODULE IFP (INPUT FILE PROCESSOR)

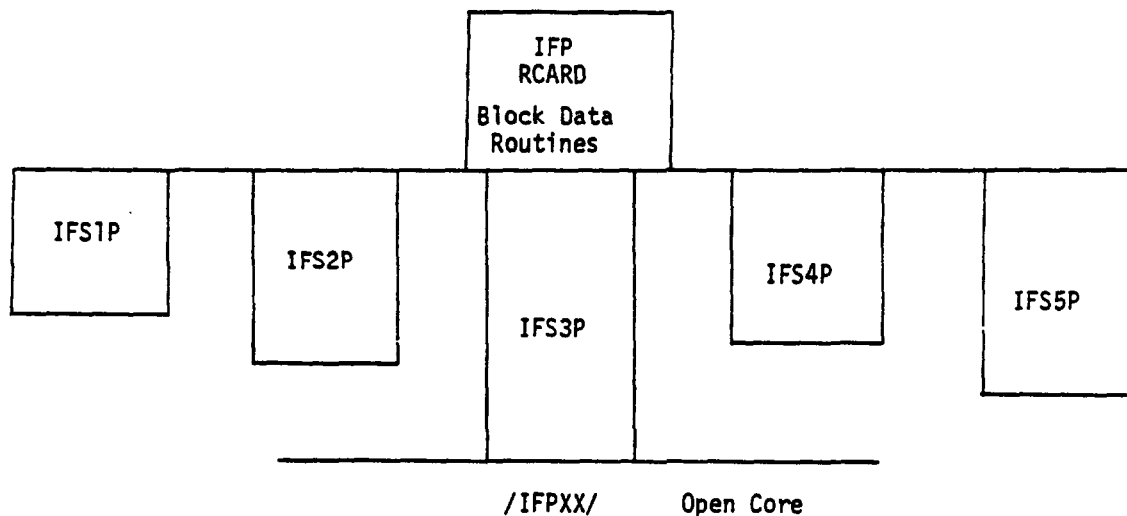
4.5.7.9 Subroutine Name: IFPDCØ

1. Entry Point: IFPDCØ
2. Purpose: LOGICAL FUNCTION IFPDCØ decodes packed component code and returns .FALSE. if no errors are detected and .TRUE. if any errors are detected. The decoded results are stored in labeled common block /IFPDTA/. This subroutine is also used by the Preface Module IFP3.

4.5.8 Design Requirements

Open core is defined in /IFPXX/. Open core is used to store all PARAM cards until the Bulk Data Deck has been exhausted, at which time the PARAM cards are written on a scratch file as the PVT Executive table.

IFP has a compilation-dependent overlay structure as shown in the sketch below.



The open core (common block /IFPXX/ must be located below the longest of the IFSiP segments. This is automatically done on the Univac 1108 but must be done by the programmer on the other machines.

4.5.9 Diagnostic Messages

If a fatal error is detected during any phase of the processing of module IFP, the NØGØ flag will be set, and the error message will be printed out. IFP will continue processing data cards until all are processed.

MODULE FUNCTIONAL DESCRIPTIONS

Table 1(a). Bulk Data Cards Processed by IFP Sorted by Internal Card Number.

The following list gives an explanation of the column headings on the following pages of Table 1.

- A = Internal IFP Bulk Data Card Number
- B = Bulk Data Card Name (an asterisk following a name implies the card is not available)
- C = Internal IFP GINØ Output File Number
- D = Data Block Name
- E = Approach Acceptance Indicator
 - 2 = Illegal for Heat Approach
 - 1 = Ignored for Heat Approach
 - 0 = OK for all approaches
 - 1 = Ignored for Displacement Approach
 - 2 = Illegal for Displacement Approach
- F = Minimum Number of Words Allowed Per Logical Card (F negative implies an open ended card)
- G = Maximum Number of Words Allowed Per Logical Card
- H = Format Check Pointer Into IFX7BD
- I = Field 2 Uniqueness Check Flag
 - 0 = No Check is Made
 - 1 = Check is Made
 - 2 = Special
- J = Subroutine LOCATE Code for Card on Output Data Block
- K = Trailer Bit Position
- L = Pointer to Secondary (Card Dependent) Code
 - S1 = Subroutine IFS1P
 - S2 = Subroutine IFS2P
 - S3 = Subroutine IFS3P
 - S4 = Subroutine IFS4P
 - S5 = Subroutine IFS5P
- M = FØRTRAN Statement Number in the Card Dependent Subroutines
- N = Conical Shell Problem Flag
 - 1 = Illegal for Shell Mode

EXECUTIVE PREFACE MODULE IFP (INPUT FILE PROCESSOR)

Table 1(b). Bulk Data Cards Processed by IFP Sorted by Internal Card Number.

0 = OK for Shell Mode

1 = Puts Card Into Different Data Block

Ø = Users Map for Data Blocks IFX2BD,...,IFX6BD

Values for I = 1,2,3 or 4

J = 1,2 or 3

H = A,B,C,D or E

K = 1,2,3,4,5 or 6

I = Data Statement in the Block Data Program

J = The Group of A Through E Continuation Card Blocks Within the Ith Data Statement

H = Alphabetic Character in Col 6 (Continuation Column) in the Jth Group

K = The Pair Number on Line H Where the Actual Data is located.

MODULE FUNCTIONAL DESCRIPTIONS

Table 1(c). Bulk Data Cards Processed by IFP Sorted by Internal Card Number.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ø IJHK
1	GRID	1	GEØM1	0	4	12	1	1	4501	45	S3	100	-1	11A1
2	GRDSET	1	GEØM1	0	4	12	13	2	0	0	S3	200	-1	11A2
3	ADUM1	1	GEØM1	0	8	8	537	0	0	0	S5	100	0	11A3
4	SEQGP	1	GEØM1	0	-4	9	37	0	5301	53	S1	40	-1	11A4
5	CØRD1R	1	GEØM1	0	4	8	37	0	1801	18	S1	500	-1	11A5
6	CØRD1C	1	GEØM1	0	4	8	37	0	1701	17	S1	600	-1	11A6
7	CØRD1S	1	GEØM1	0	4	8	37	0	1901	19	S1	700	-1	11B1
8	CØRD2R	1	GEØM1	0	12	16	45	1	2101	21	S1	800	-1	11B2
9	CØRD2C	1	GEØM1	0	12	16	45	1	2001	20	S1	900	-1	11B3
10	CØRD2S	1	GEØM1	0	12	16	45	1	2201	22	S1	1000	-1	11B4
11	PLØTEL	8	GEØM2	0	4	8	505	0	5201	52	S1	1111	-1	11B5
12	SPC1	10	GEØM4	0	-4	9	-1	0	5481	58	S3	3980	-1	11B6
13	SPCADD	10	GEØM4	0	4	8	-1	1	5491	59	S3	4020	1	11C1
14	SUPØRT	10	GEØM4	0	4	8	37	0	5601	56	S1	1400	-1	11C2
15	ØMIT	10	GEØM4	0	4	8	37	0	5001	50	S1	1400	-1	11C3
16	SPC	10	GEØM4	0	4	8	101	0	5501	55	S1	1600	-1	11C4
17	MPC	10	GEØM4	0	4	8	-1	0	4901	49	S3	1700	-1	11C5
18	FØRCE	9	GEØM3	0	8	12	109	0	4201	42	S1	1800	1	11C6
19	MØMENT	9	GEØM3	0	8	12	109	0	4801	48	S1	1800	1	11D1
20	FØRCE1	9	GEØM3	0	8	12	121	0	4001	40	S1	2000	-1	11D2
21	MØMENT1	9	GEØM3	0	8	12	121	0	4601	46	S1	2000	-1	11D3
22	FØRCE2	9	GEØM3	0	8	12	133	0	4101	41	S1	2200	-1	11D4
23	MØMENT2	9	GEØM3	0	8	12	133	0	4701	47	S1	2200	-1	11D5
24	PLØAD	9	GEØM3	0	8	12	145	0	5101	51	S1	2400	-1	11D6
25	SLØAD	9	GEØM3	0	4	8	157	0	5401	54	S1	2500	-1	11E1
26	GRAV	9	GEØM3	0	8	12	165	1	4401	44	S1	2600	1	11E2
27	TEMP	9	GEØM3	0	4	8	157	0	5701	57	S1	2500	-1	11E3
28	GENEL	8	GEØM2	0	-4	9	-1	1	4301	43	S3	2800	-1	11E4
29	PRØD	2	EPT	0	4	12	165	1	902	9	S1	2900	-1	11E5
30	PTUBE	2	EPT	0	4	12	177	1	1602	16	S1	2920	-1	11E6
31	PVISC	2	EPT	0	4	8	189	0	1802	18	S1	310	-1	12A1
32	ADUM2	1	GEØM1	0	-4	9	537	0	0	0	S5	200	0	12A2
33	PTRIA1	2	EPT	0	4	16	221	1	1202	12	S1	2980	-1	12A3
34	PTRIA2	2	EPT	0	4	8	237	0	1302	13	S1	3000	-1	12A4
35	PTRBSC	2	EPT	0	4	12	257	1	1102	11	S1	3020	-1	12A5
36	PTRPLT	2	EPT	0	4	12	257	1	1502	15	S1	3020	-1	12A6
37	PTRMEM	2	EPT	0	4	8	237	0	1402	14	S1	3000	-1	12B1
38	PQUAD1	2	EPT	0	4	16	221	1	702	7	S1	2980	-1	12B2
39	PQUAD2	2	EPT	0	4	8	237	0	802	8	S1	3000	-1	12B3
40	PQDPLT	2	EPT	0	4	12	257	0	602	6	S1	3020	-1	12B4
41	PQDMEM	2	EPT	0	4	8	237	0	502	5	S1	3000	-1	12B5
42	PSHEAR	2	EPT	0	4	8	237	0	1002	10	S1	3000	-1	12B6
43	PTWIST	2	EPT	0	4	8	237	0	1702	17	S1	3000	-1	12C1
44	PMASS	2	EPT	0	4	8	269	0	402	4	S1	3200	-1	12C2
45	PDAMP	2	EPT	0	4	8	269	0	202	2	S1	3200	-1	12C3
46	PELAS	2	EPT	0	4	8	497	0	302	3	S1	3240	-1	12C4
47	CØNRØD	8	GEØM2	0	8	12	277	1	1601	16	S1	3260	-1	12C5
48	CRØD	8	GEØM2	0	4	8	37	0	3001	30	S1	3281	-1	12C6
49	CTUBE	8	GEØM2	0	4	8	37	0	3701	37	S1	3282	-1	12D1
50	CVISC	8	GEØM2	0	4	8	37	0	3901	39	S1	3283	-1	12D2
51	ADUM3	1	GEØM1	0	-4	9	537	0	0	0	S5	300	0	12D3
52	CTRIA1	8	GEØM2	0	8	12	313	1	3301	33	S1	3360	-1	12D4
53	CTRIA2	8	GEØM2	0	8	12	313	1	3401	34	S1	3360	-1	12D5
54	CTRBSC	8	GEØM2	0	8	12	313	1	3201	32	S1	3360	-1	12D6
55	CTRPLT	8	GEØM2	0	8	12	313	1	3601	36	S1	3360	-1	12E1
56	CTRMEM	8	GEØM2	0	8	12	313	1	3501	35	S1	3360	-1	12E2
57	CQUAD1	8	GEØM2	0	8	12	325	1	2801	28	S1	3460	-1	12E3
58	CQUAD2	8	GEØM2	0	8	12	325	1	2901	29	S1	3460	-1	12E4

EXECUTIVE PREFACE MODULE IFP (INPUT FILE PROCESSOR)

Table 1(d). Bulk Data Cards Processed by IFP Sorted by Internal Code Number.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ø IJHK
59	CQDPLT	8	GEØM2	0	8	12	325	1	2701	27	S1	3460	-1	12E5
60	CQDMEM	8	GEØM2	0	8	12	325	1	2601	26	S1	3460	-1	12E6
61	CSHEAR	8	GEØM2	0	8	12	337	1	3101	31	S1	3540	-1	13A1
62	CTWIST	8	GEØM2	0	8	12	337	1	3801	38	S1	3540	-1	13A2
63	CØNM1	8	GEØM2	0	8	28	349	1	1401	14	S1	3580	-1	13A3
64	CØNM2	8	GEØM2	0	8	20	377	1	1501	15	S1	3600	-1	13A4
65	CMASS1	8	GEØM2	0	4	12	337	1	1001	10	S1	3620	-1	13A5
66	CMASS2	8	GEØM2	0	4	12	397	1	1101	11	S1	3623	-1	13A6
67	CMASS3	8	GEØM2	0	4	8	37	0	1201	12	S1	3674	-1	13B1
68	CMASS4	8	GEØM2	0	4	8	409	0	1301	13	S1	3697	-1	13B2
69	CDAMP1	8	GEØM2	0	4	12	337	1	201	2	S1	3620	-1	13B3
70	CDAMP2	8	GEØM2	0	4	12	397	1	301	3	S1	3623	-1	13B4
71	CDAMP3	8	GEØM2	0	4	8	37	0	401	4	S1	3675	-1	13B5
72	CDAMP4	8	GEØM2	0	4	8	409	0	501	5	S1	3698	-1	13B6
73	CELAS1	8	GEØM2	0	4	12	337	1	601	6	S1	3620	-1	13C1
74	CELAS2	8	GEØM2	0	4	12	417	1	701	7	S1	3800	-1	13C2
75	CELAS3	8	GEØM2	0	4	8	37	0	801	8	S1	3676	-1	13C3
76	CELAS4	8	GEØM2	0	4	8	409	0	901	9	S1	3699	-1	13C4
77	MAT1	3	MPT	0	4	20	429	1	103	1	S1	3860	0	13C5
78	MAT2	3	MPT	0	8	20	449	1	203	2	S1	3880	0	13C6
79	CTRIARG	8	GEØM2	0	8	12	738	1	1708	17	S4	790	-1	13D1
80	CTRAPRG	8	GEØM2	0	8	12	737	1	1808	18	S4	800	-1	13D2
81	DEFØRM	4	EDT	0	4	8	157	0	104	1	S1	2500	-1	13D3
82	PARAM	6	PVT	0	-5	16	-1	2	0	0	S3	3960	0	13D4
83	MPCADD	10	GEØM4	0	4	8	-1	1	4891	60	S3	4020	1	13D5
84	LØAD	9	GEØM3	0	4	8	-1	1	4551	61	S3	4060	1	13D6
85	EIGR	7	DYNAMICS	0	14	18	469	1	307	3	S2	850	0	13E1
86	EIGB	7	DYNAMICS	0	14	18	469	1	107	1	S2	850	0	13E2
87	EIGC	7	DYNAMICS	0	-4	10	-1	1	207	2	S2	870	0	13E3
88	ADUM4	1	GEØM1	0	-4	9	537	0	0	0	S5	400	0	13E4
89		1	GEØM1	0	8	8	-1	2	0	0	S2	890	0	13E5
90	MATS1	3	MPT	0	4	16	545	1	503	5	S4	900	-1	13E6
91	MATT1	3	MPT	0	4	16	545	1	703	7	S4	900	0	21A1
92	ØMIT1	10	GEØM4	0	-4	9	-1	0	4951	63	S3	3981	-1	21A2
93	TABLEM1	5	DIT	0	-4	16	-1	1	105	1	S2	930	0	21A3
94	TABLEM2	5	DIT	0	-4	16	-1	1	205	2	S2	930	0	21A4
95	TABLEM3	5	DIT	0	-4	16	-1	1	305	3	S2	930	0	21A5
96	TABLEM4	5	DIT	0	-4	16	-1	1	405	4	S2	960	0	21A6
97	TABLES1	5	DIT	0	-4	16	-1	1	3105	31	S2	930	-1	21B1
98	TEMPD	9	GEØM3	0	4	12	269	0	5641	65	S4	980	1	21B2
99	ADUM5	1	GEØM1	0	-4	9	537	0	0	0	S5	500	0	21B3
100	ADUM6	1	GEØM1	0	-4	9	537	0	0	0	S5	600	0	21B4
101	ADUM7	1	GEØM1	0	-4	9	537	0	0	0	S5	700	0	21B5
102	MATT2	3	MPT	0	4	16	525	1	803	8	S4	1020	-1	21B6
103	ADUM8	1	GEØM1	0	-4	9	537	0	0	0	S5	800	0	21C1
104	CTØRDRG	8	GEØM2	0	4	12	750	1	1908	19	S4	1040	-1	21C2
105	SPØINT	8	GEØM2	0	-4	9	794	0	5551	49	S4	1050	-1	21C3
106	ADUM9	1	GEØM1	0	-4	9	537	0	0	0	S5	900	0	21C4
107	CDUM1	8	GEØM2	0	8	24	925	1	6108	61	S5	1000	0	21C5
108	CDUM2	8	GEØM2	0	8	24	925	1	6208	62	S5	1100	0	21C6
109	CDUM3	8	GEØM2	0	8	24	925	1	6308	63	S5	1200	0	21D1
110	CDUM4	8	GEØM2	0	8	24	925	1	6408	64	S5	1300	0	21D2
111	CDUM5	8	GEØM2	0	8	24	925	1	6508	65	S5	1400	0	21D3
112	CDUM6	8	GEØM2	0	8	24	925	1	6608	66	S5	1500	0	21D4
113	CDUM7	8	GEØM2	0	8	24	925	1	6708	67	S5	1600	0	21D5
114	CDUM8	8	GEØM2	0	8	24	925	1	6808	68	S5	1700	0	21D6
115	CDUM9	8	GEØM2	0	8	24	925	1	6908	69	S5	1800	0	21E1
116	PDUM1	2	EPT	0	4	24	925	1	6102	61	S5	1900	0	21E2

MODULE FUNCTIONAL DESCRIPTIONS

Table 1(e). Bulk Data Cards Processed by IFP Sorted by Internal Card Number.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ø IJHK
117	PDUM2	2	EPT	0	4	24	925	1	6202	62	S5	2000	0	21E3
118	PDUM3	2	EPT	0	4	24	925	1	6302	63	S5	2100	0	21E4
119	DMI	12	PØØL	0	-4	16	-1	0	0	0	S2	1190	0	21E5
120	DMIG	14	MATPØØL	0	-4	12	-1	0	114	1	S2	1200	0	21E6
121	PTØRDRG	2	EPT	0	4	8	237	0	2102	21	S1	3000	-1	22A1
122	MAT3	3	MPT	0	4	20	449	1	1403	14	S4	1220	-1	22A2
123	DLØAD	7	DYNAMICS	0	4	8	-1	1	57	5	S3	4060	0	22A3
124	EPØINT	7	DYNAMICS	0	-4	9	794	0	707	7	S4	1050	0	22A4
125	FREQ1	7	DYNAMICS	0	4	8	705	0	1007	10	S1	1250	0	22A5
126	FREQ	7	DYNAMICS	0	4	8	-1	1	1307	13	S3	1260	0	22A6
127	NØLIN1	7	DYNAMICS	0	8	12	725	0	3107	31	S1	1270	0	22B1
128	NØLIN2	7	DYNAMICS	0	8	12	725	0	3207	32	S1	1280	0	22B2
129	NØLIN3	7	DYNAMICS	0	8	12	725	0	3307	33	S1	1290	0	22B3
130	NØLIN4	7	DYNAMICS	0	8	12	725	0	3407	34	S1	1290	0	22B4
131	RLØAD1	7	DYNAMICS	0	8	8	337	1	5107	51	S3	1310	0	22B5
132	RLØAD2	7	DYNAMICS	0	8	8	337	1	5207	52	S3	1310	0	22B6
133	TABLED1	5	DIT	0	-4	16	-1	1	1105	11	S2	930	0	22C1
134	TABLED2	5	DIT	0	-4	16	-1	1	1205	12	S2	930	0	22C2
135	SEQEP	7	DYNAMICS	0	4	8	37	0	5707	57	S1	40	-1	22C3
136	TF	7	DYNAMICS	0	8	12	-1	0	6207	62	S1	1360	0	22C4
137	TIC	7	DYNAMICS	0	4	12	713	0	6607	66	S1	1370	0	22C5
138	TLØAD1	7	DYNAMICS	0	8	8	681	1	7107	71	S3	1380	0	22C6
139	TLØAD2	7	DYNAMICS	0	8	16	689	1	7207	72	S3	1390	0	22D1
140	TABLED3	5	DIT	0	-4	16	-1	1	1305	13	S2	930	0	22D2
141	TABLED4	5	DIT	0	-4	16	-1	1	1405	14	S2	960	0	22D3
142	TSTEP	7	DYNAMICS	0	4	8	-1	1	8307	83	S1	1420	0	22D4
143	DSFACT	3	MPT	0	4	8	-1	1	53	10	S3	1430	0	22D5
144	AXIC	15	AXIC	0	4	8	93	0	515	5	S3	1440	0	22D6
145	RINGAX	15	AXIC	0	4	12	245	1	5615	56	S3	1450	0	22E1
146	CCØNEAX	15	AXIC	0	4	8	645	1	2315	23	S3	1460	0	22E2
147	PCØNEAX	2	EPT	0	4	28	653	1	152	19	S3	1470	0	22E3
148	SPCAX	15	AXIC	0	4	12	485	0	6215	62	S3	1480	0	22E4
149	MPCAX	15	AXIC	0	4	8	-1	0	4015	40	S3	1490	0	22E5
150	ØMITAX	15	AXIC	0	4	8	337	0	4315	43	S3	1500	0	22E6
151	SUPAX	15	AXIC	0	4	8	337	0	6415	64	S3	1500	0	23A1
152	PØINTAX	15	AXIC	0	4	8	517	1	4915	49	S3	1520	0	23A2
153	SECTAX	15	AXIC	0	4	12	177	1	6015	60	S3	1530	0	23A3
154	PRESAX	15	AXIC	0	4	12	61	0	5215	52	S3	1540	0	23A4
155	TEMPAX	15	AXIC	0	4	8	237	0	6815	68	S3	1550	0	23A5
156	FØRCEAX	15	AXIC	0	-4	13	109	0	2115	21	S3	1560	0	23A6
157	MØMAX	15	AXIC	0	-4	13	109	0	3815	38	S3	1560	0	23B1
158	EIGP	7	DYNAMICS	0	4	8	561	0	257	4	S1	1580	0	23B2
159	PDUM4	2	EPT	0	4	24	925	1	6402	64	S5	2200	0	23B3
160	PDUM5	2	EPT	0	4	24	925	1	6502	65	S5	2300	0	23B4
161	PDUM6	2	EPT	0	4	24	925	1	6602	66	S5	2400	0	23B5
162	TABDMP1	5	DIT	0	-4	16	-1	1	15	21	S2	930	0	23B6
163	PDUM7	2	EPT	0	4	24	925	1	6702	67	S5	2500	0	23C1
164	PDUM8	2	EPT	0	4	24	925	1	6802	68	S5	2600	0	23C2
165	PDUM9	2	EPT	0	4	24	925	1	6902	69	S5	2700	0	23C3
166	FREQ2	7	DYNAMICS	0	4	8	705	0	1107	11	S1	1660	0	23C4
167	CØNCT1	10	GEØM4	0	-4	20	-1	0	110	41	S5	2800	-1	23C5
168	CØNCT	10	GEØM4	0	-4	12	-1	0	210	2	S5	2900	-1	23C6
169	TRANS	10	GEØM4	0	12	16	45	1	310	3	S5	3000	-1	23D1
170	RELES	10	GEØM4	0	-4	12	-1	0	410	4	S5	3100	-1	23D2
171	LØADC	10	GEØM4	0	-8	16	-1	0	500	5	S5	3200	-1	23D3
172	SPCSD	10	GEØM4	0	6	9	1000	0	610	6	S5	3300	-1	23D4
173	SPCS1	10	GEØM4	0	-4	12	-1	0	710	7	S5	3400	-1	23D5
174	SPCS	10	GEØM4	0	-4	12	-1	0	810	8	S5	3500	-1	23D6

EXECUTIVE PREFACE MODULE IFP (INPUT FILE PROCESSOR)

Table 1(f). Bulk Data Cards Processed by IFP. Sorted by Internal Card Number.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ø IJHK
175	BDYC	10	GEØM4	0	-8	16	-1	1	910	9	S5	3600	-1	23E1
176	WPCS	10	GEØM4	0	-8	12	-1	0	1110	11	S5	3700	-1	23E2
177	BDYS	10	GEØM4	0	4	12	-1	0	1210	12	S5	3800	-1	23E3
178	BDYS1	10	GEØM4	0	-4	12	-1	0	1310	13	S5	3900	-1	23E4
179	BARØR	8	GEØM2	0	4	12	25	2	0	0	S1	100	-1	23E5
180	CBAR	8	GEØM2	0	8	20	73	1	2408	24	S1	200	-1	23E6
181	PBAR	2	EPT	0	4	24	621	1	52	20	S1	300	-1	31A1
182	DAREA	7	DYNAMICS	0	4	8	101	0	27	17	S3	1820	0	31A2
183	DELAY	7	DYNAMICS	0	4	8	101	0	37	18	S3	1820	0	31A3
184	DPHASE	7	DYNAMICS	0	4	8	101	0	77	19	S3	1820	0	31A4
185	PI FACT	3	MPT	0	4	8	-1	1	1103	11	S3	1420	-1	31A5
186	GNEW*	10	GEØM4	0	9	9	993	0	1410	14	S5	4000	-1	31A6
187	GTRAN	10	GEØM4	0	5	9	993	0	1510	15	S5	4100	-1	31B1
188	TABRNDG	5	DIT	0	4	8	1	0	56	26	S2	1000	0	31C2
189	MATT3	3	MPT	0	4	16	525	1	1503	15	S4	1020	-1	31B3
190	RFØRCE	9	GEØM3	0	8	12	109	0	5509	55	S1	1900	1	31B4
191	TABRND1	5	DIT	0	-4	16	-1	1	55	25	S2	930	0	31B5
192	UDEF	10	GEØM4	0	-4	16	-1	0	0	0	S5	4300	0	31B6
193	USET	10	GEØM4	0	-4	9	1065	0	110	1	S5	4400	0	31C1
194	USET1	10	GEØM4	0	-4	10	-1	0	210	2	S5	4500	0	31C2
195	RANDPS	7	DYNAMICS	0	4	12	782	0	2107	21	S4	1950	-1	31C3
196	RANDT1	7	DYNAMICS	0	4	8	752	0	2207	22	S4	1960	-1	31C4
197	RANDT2*	7	DYNAMICS	0	-4	8	-1	1	2307	23	S9		-1	31C5
198	PLØAD1*	9	GEØM3	0	0	0	0	0	6909	69	S9		-1	31C6
199	PLØAD2	9	GEØM3	0	-4	9	774	0	6802	68	S4	1990	-1	31D1
200	DTI	12	PØØL	0	-4	16	-1	0	0	0	S2	2000	0	31D2
201	TEMPP1	9	GEØM3	0	-4	10	-1	0	8109	81	S4	2100	-1	31D3
202	TEMPP2	9	GEØM3	0	-4	10	-1	0	8209	82	S4	2200	-1	31D4
203	TEMPP3	9	GEØM3	0	-4	10	-1	0	8309	83	S4	2300	-1	31D5
204	TEMPRB	9	GEØM3	0	-4	10	-1	0	8409	84	S4	2400	-1	31D6
205	GRIDB	15	AXIC	0	8	12	1	1	8115	81	S4	3100	-1	31E1
206	FSLIST	15	AXIC	0	-4	10	-1	0	8215	82	S4	3200	-1	31E2
207	RINGFL	15	AXIC	0	4	8	497	1	8315	83	S4	3300	-1	31E3
208	PRESPT	15	AXIC	0	4	8	834	0	8415	84	S4	3400	-1	31E4
209	CFLUID2	15	AXIC	0	8	8	845	1	8515	85	S4	3500	-1	31E5
210	CFLUID3	15	AXIC	0	8	8	853	1	8615	86	S4	3600	-1	31E6
211	CFLUID4	15	AXIC	0	8	8	861	1	8715	87	S4	3700	-1	32A1
212	AXIF	15	AXIC	0	-8	10	-1	0	8815	88	S4	3800	-1	32A2
213	BDYLIST	15	AXIC	0	-4	10	-1	0	8915	89	S4	3900	-1	32A3
214	FREETPT	15	AXIC	0	4	8	834	0	9015	90	S4	4000	-1	32A4
215	ASET	10	GEØM4	0	4	8	37	0	5561	76	S1	1400	-1	32A5
216	ASET1	10	GEØM4	0	-4	9	-1	0	5571	77	S3	3931	-1	32A6
217	CTETRA	8	GEØM2	0	8	8	337	1	5508	55	S4	4100	-1	32B1
218	CWEDGE	8	GEØM2	0	8	8	525	1	5608	56	S4	4200	-1	32B2
219	CHEXA1	8	GEØM2	0	16	16	531	1	5708	57	S4	4300	-1	32B3
220	CHEXA2	8	GEØM2	0	16	16	531	1	5808	58	S4	4400	-1	32B4
221	DMIAx	14	MATPØØL	0	-4	9	-1	0	214	2	S4	4500	-1	32B5
222	FLSYM	15	AXIC	0	4	10	826	0	9115	91	S4	4600	-1	32B6
223	AXSLØT	15	AXIC	0	8	8	869	1	1115	11	S1	4100	0	32C1
224	CAXIF2	8	GEØM2	0	8	8	877	1	2108	21	S1	4200	0	32C2
225	CAXIF3	8	GEØM2	0	8	8	877	1	2208	22	S1	4300	0	32C3
226	CAXIF4	8	GEØM2	0	8	8	877	1	2308	23	S1	4400	0	32C4
227	CSLØT3	8	GEØM2	0	8	8	877	1	4408	44	S1	4500	0	32C5
228	CSLØT4	8	GEØM2	0	8	16	877	1	4508	45	S1	4600	0	32C6
229	GRIDF	15	AXIC	0	4	8	885	1	1215	12	S1	4700	0	32D1
230	GRIDS	15	AXIC	0	4	8	893	1	1315	13	S1	4800	0	32D2
231	SLBDY	15	AXIC	0	-4	8	-1	0	1415	14	S1	4900	0	32D3
232	CHBDY	8	GEØM2	0	9	17	909	1	4208	42	S1	5000	-1	32D4

MODULE FUNCTIONAL DESCRIPTIONS

Table 1(g). Bulk Data Cards Processed by IFP Sorted by Internal Card Number.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ø IJHK
233	QHBDY	9	GEØM3	0	9	9	901	0	4309	43	S1	5050	-1	32D5
234	MAT4	3	MPT	0	4	8	198	1	2103	21	S1	3900	-1	32D6
235	MAT5	3	MPT	0	4	8	198	1	2203	22	S1	4000	-1	32E1
236	PHBDY	2	EPT	0	4	8	350	1	2502	25	S1	5100	-1	32E2
237	MATT4	3	MPT	0	4	8	37	1	2303	23	S1	3950	-1	32E3
238	MATT5	3	MPT	0	4	8	37	1	2403	24	S1	4050	-1	32E4
239	QBDY1	9	GEØM3	0	-4	9	774	0	4509	45	S4	1990	-1	32E5
240	QBDY2	9	GEØM3	0	4	8	350	0	4909	49	S1	5150	-1	32E6
241	QVECT	9	GEØM3	0	-4	16	-1	0	5009	50	S1	5200	-1	33A1
242	QVØL	9	GEØM3	0	-4	9	774	0	5209	52	S1	1990	-1	33A2
243	RADLST	14	MATPØØL	0	-4	16	-1	0	2014	20	S1	5250	-1	33A3
244	RADMTX	14	MATPØØL	0	-4	8	-1	0	3014	30	S3	1400	-1	33A4
245	SAME*	10	GEØM4	0	-4	10	-1	0	7810	78	S5	4600	-1	33A5
246	SAME1*	10	GEØM4	0	-8	9	-1	0	7910	79	S5	5	-1	33A6
247	INPUT	11	GEØM5	0	17	17	-1	0	1310	13	S5	5	-1	33B1
248	ØUTPUT	11	GEØM5	0	17	17	-1	0	1410	14	S5	5	-1	33B2
249	CQDMEM1	8	GEØM2	0	8	12	325	0	2008	20	S1	3460	-1	33B3
250	PQDMEM1	2	EPT	0	4	8	237	0	2202	22	S1	3000	-1	33B4
251	CIHEX1	8	GEØM2	0	12	16	951	1	7108	71	S5	5000	-1	33B5
252	CIHEX2	8	GEØM2	0	24	28	951	1	7208	72	S5	5100	-1	33B6
253	CIHEX3	8	GEØM2	0	36	40	951	1	7308	73	S5	5200	-1	33C1
254	PIHEX	2	EPT	0	4	12	981	1	7002	70	S5	5300	-1	33C2
255	PLØAD3	9	GEØM3	0	8	12	949	0	7109	71	S5	5400	-1	33C3
256	SPCD	10	GEØM4	0	4	8	101	0	5110	51	S1	1600	-1	33C4
257	CYJØIN	10	GEØM4	0	-4	16	-1	0	5210	52	S1	5240	-1	33C5
258	CNGRNT	8	GEØM2	0	-4	16	-1	0	5008	50	S1	5245	-1	33C6
259	CQDMEM2	8	GEØM2	0	8	12	325	0	5308	53	S1	3460	-1	33D1
260	PQDMEM2	2	EPT	0	4	8	237	0	5302	53	S1	3000	-1	33D2
261	CQDMEM3	8	GEØM2	0	8	12	325	0	5408	54	S1	3460	-1	33D3
262	PQDMEM3	2	EPT	0	4	8	237	0	5402	54	S1	3000	-1	33D4
263	CAERO1	4	EDT	0	16	16	39	1	3002	30	S5	6400	-1	33D5
264	PAERO1	4	EDT	0	4	8	803	1	3102	31	S5	6500	-1	33D6
265	AERØ	4	EDT	0	8	12	2	0	3202	32	S5	6600	-1	33E1
266	SPLINE1	4	EDT	0	8	12	42	1	3302	33	S5	6700	-1	33E2
267	SPLINE2	4	EDT	0	12	16	1025	1	3402	34	S5	6800	-1	33E3
268	SET1	4	EDT	0	-4	16	-1	0	3502	35	S1	5300	-1	33E4
269	SET2	4	EDT	0	4	8	197	0	3602	36	S5	5600	-1	33E5
270	MKAERO2	4	EDT	0	4	8	805	0	3702	37	S5	5700	-1	33E6
271	MKAERO1	4	EDT	0	16	16	805	0	3802	38	S5	5800	-1	41A1
272	FLUTTER	4	EDT	0	10	14	1005	1	3902	39	S5	5900	-1	41A2
273	AEFACT	4	EDT	0	-4	16	-1	1	4002	40	S3	1415	-1	41A3
274	FLFACT	4	EDT	0	-4	16	-1	1	4102	41	S3	1415	-1	41A4
275	CBARAO*	8	GEØM2	0	9	13	-1	1	4001	40	S5	6100	-1	41A5
276	PLIMIT	3	MPT	0	-9	14	-1	0	304	3	S5	6200	-1	41A6
277	PØPT	3	MPT	0	9	13	1017	0	404	4	S5	6300	-1	41B1
278	PLØADX*	9	GEØM3	0	8	12	1037	0	7001	70	S5	6900	-1	41B2
279	CRIGD1	10	GEØM4	-2	-3	48	-1	1	5310	53	S3	2010	-1	41B3
280	CQUADTS*	8	GEØM2	0	8	20	1045	1	4108	41	S4	2020	-1	41B4
281	PQUADTS*	2	EPT	0	8	12	277	1	2402	24	S4	2030	-1	41B5
282	CTRIATS*	8	GEØM2	0	8	20	1047	1	5908	59	S4	2021	-1	41B6
283	PTRIATS*	2	EPT	0	8	12	277	1	2302	23	S4	2030	-1	41C1
284	CRIGD2	10	GEØM4	-2	-4	48	-1	1	5410	54	S3	2060	-1	41C2
285	CTRIAAX	15	AXIC	-2	4	8	313	1	7012	70	S3	2111	0	41C3
286	PTRIAAX	2	EPT	-2	4	24	349	1	7032	85	S3	2030	0	41C4
287	CTRAPAX	15	AXIC	-2	4	8	325	1	7042	74	S3	2040	0	41C5
288	PTRAPAX	2	EPT	-2	4	24	349	1	7052	95	S3	2030	0	41C6
289	VIEW	2	EPT	0	4	8	326	1	2606	26	S1	5175	0	41D1
290	VARIAN	4	EDT	0	-4	16	-1	0	4202	42	S3	1410	0	41D2

EXECUTIVE PREFACE MODULE IFP (INPUT FILE PROCESSOR)

Table 1(h). Bulk Data Cards Processed by IFP Sorted by Internal Card Number.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ø IJHK
291	CTRIM6	8	GEØM2	-2	12	16	913	1	6101	81	S1	6101	-1	41D3
292	PTRIM6	8	GEØM2	-2	8	12	802	1	6201	82	S1	6201	-1	41D4
293	CTRPLT1	8	GEØM2	-2	12	16	913	1	6301	83	S1	6301	-1	41D5
294	PTRPLT1	2	EPT	-2	8	20	1089	1	6401	84	S1	6401	-1	41D6
295	TEMPG	9	GEØM3	-2	-8	20	-1	0	8509	85	S4	6501	-1	41E1
296	TEMPP4	9	GEØM3	-2	-8	20	-1	0	8609	86	S4	6601	-1	41E2
297	CRIGDR	10	GEØM4	-2	4	8	37	1	8210	82	S3	6000	-1	41E3
298	CRIGD3	10	GEØM4	-2	-3	48	-1	1	8301	83	S3	7000	-1	41E4
299	CTRSHL	8	GEØM2	-2	12	16	913	1	7501	75	S1	7501	-1	41E5
300	PTRSHL	2	EPT	-2	20	24	1005	1	7601	76	S1	7601	-1	41E6
301	CAERØ2	4	EDT	0	12	16	39	1	4301	43	S5	6400	-1	42A1
302	CAERØ3	4	EDT	0	16	16	39	1	4401	44	S5	6400	-1	42A2
303	CAERØ4	4	EDT	0	16	16	39	1	4501	45	S5	6400	-1	42A3
304	PAERØ2	4	EDT	0	16	16	1162	1	4601	46	S5	6510	-1	42A4
305	PAERØ3	4	EDT	0	4	24	801	1	4701	47	S5	6520	-1	42A5
306	PAERØ4	4	EDT	0	-4	8	-1	1	4801	48	S5	6530	-1	42A6
307	SPLINE3	4	EDT	0	-4	16	-1	0	4901	49	S5	6850	-1	42B1
308	GUST	5	DIT	0	4	8	165	0	1005	10	S5	7600	-1	42B2
309	CAERØ5	4	EDT	0	16	16	39	1	5001	50	S5	6400	-1	42B3
310	PAERØ5	4	EDT	0	-4	8	-1	1	5101	51	S5	7700	-1	42B4
311	DAREAS	7	DYNAMICS	-2	-4	9	1080	0	9027	90	S5	3300	-1	42B5
312	DELAYS	7	DYNAMICS	-2	-4	9	1080	0	9137	91	S5	3300	-1	42B6
313	DPHASES	7	DYNAMICS	-2	-4	9	1080	0	9277	92	S5	3300	-1	42C1
314	TICS	7	DYNAMICS	-2	-4	9	1080	0	9307	93	S5	3350	-1	42C2

EXECUTIVE PREFACE MODULE IFP (INPUT FILE PROCESSOR)

(THIS PAGE LEFT BLANK FOR FUTURE USE)

EXECUTIVE PREFACE MODULE IFP (INPUT FILE PROCESSOR)

THIS PAGE HAS BEEN LEFT BLANK INTENTIONALLY.

MODULE FUNCTIONAL DESCRIPTIONS

Table 2(a). Bulk Data Cards Processed by IFP Sorted Alphabetically by Card Name.

The following list gives an explanation of the column headings on the following pages of Table 2.

- A = Internal IFP Bulk Data Card Number
- B = Bulk Data Card Name (an asterisk following a name implies the card is not available)
- C = Internal IFP GINØ Output File Number
- D = Data Block Name
- E = Approach Acceptance Indicator
 - 2 = Illegal for Heat Approach
 - 1 = Ignored for Heat Approach
 - 0 = OK for all approaches
 - 1 = Ignored for Displacement Approach
 - 2 = Illegal for Displacement Approach
- F = Minimum Number of Words Allowed Per Logical Card (F negative implies an open ended card)
- G = Maximum Number of Words Allowed Per Logical Card
- H = Format Check Pointer Into IFX7BD
- I = Field 2 Uniqueness Check Flag
 - 0 = No Check is Made
 - 1 = Check is Made
 - 2 = Special
- J = Subroutine LOCATE Code for Card on Output Data Block
- K = Trailer Bit Position
- L = Pointer to Secondary (Card Dependent) Code
 - S1 = Subroutine IFS1P
 - S2 = Subroutine IFS2P
 - S3 = Subroutine IFS3P
 - S4 = Subroutine IFS4P
 - S5 = Subroutine IFS5P
- M = FØRTRAN Statement Number in the Card Dependent Subroutines
- N = Conical Shell Problem Flag
 - 1 = Illegal for Shell Mode

EXECUTIVE PREFACE MODULE IFP (INPUT FILE PROCESSOR)

Table 2(b). Bulk Data Cards Processed by IFP Sorted Alphabetically by Card Name.

0 = OK for Shell Mode

1 = Puts Card Into Different Data Block

Ø = Users Map for Data Blocks IFX2BD,...,IFX6BD

Values for I = 1,2,3 or 4

J = 1,2 or 3

H = A,B,C,D or E

K = 1,2,3,4,5 or 6

I = Data Statement in the Block Data Program

J = The Group of A Through E Continuation Card Blocks Within the Ith Data Statement

H = Alphabetic Character in Col 6 (Continuation Column) in the Jth Group

K = The Pair Number on Line H Where the Actual Data is located.

MODULE FUNCTIONAL DESCRIPTIONS

Table 2(c). Bulk Data Cards Processed by IFP, Sorted Alphabetically by Card Name.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ø IJHK
89		1	GEØM1	0	8	8	-1	2	0	0	S2	890	0	13E5
3	ADUM1	1	GEØM1	0	-4	9	537	0	0	0	S5	100	0	11A3
32	ADUM2	1	GEØM1	0	-4	9	537	0	0	0	S5	200	0	12A2
51	ADUM3	1	GEØM1	0	-4	9	537	0	0	0	S5	300	0	12D3
88	ADUM4	1	GEØM1	0	-4	9	537	0	0	0	S5	400	0	13E4
99	ADUM5	1	GEØM1	0	-4	9	537	0	0	0	S5	500	0	21B3
100	ADUM6	1	GEØM1	0	-4	9	537	0	0	0	S5	600	0	21B4
101	ADUM7	1	GEØM1	0	-4	9	537	0	0	0	S5	700	0	21B5
103	ADUM8	1	GEØM1	0	-4	9	537	0	0	0	S5	800	0	21C1
106	ADUM9	1	GEØM1	0	-4	9	537	0	0	0	S5	900	0	21C4
273	AEFACT	4	EDT	0	-4	16	-1	1	4002	40	S3	1415	-1	41A3
265	AERØ	4	EDT	0	8	12	2	0	3202	32	S5	6600	-1	33E1
215	ASET	10	GEØM4	0	4	8	37	0	5561	76	S1	1400	-1	32A5
216	ASET1	10	GEØM4	0	-4	9	-1	0	5571	77	S3	3981	-1	32A6
144	AXIC	15	AXIC	0	4	8	93	0	515	5	S3	1440	0	22D6
212	AXIF	15	AXIC	0	-8	10	-1	0	8815	88	S4	3800	-1	32A2
223	AXSLØT	15	AXIC	0	8	8	869	1	1115	11	S1	4100	0	32C1
179	BARØR	9	GEØM2	0	4	12	25	2	0	0	S1	100	-1	23E5
175	BDYC	10	GEØM4	0	-8	16	-1	1	910	9	S5	3600	-1	23E1
213	BDYLIST	15	AXIC	0	-4	10	-1	0	8915	89	S4	3900	-1	32A3
177	BDYS	10	GEØM4	0	4	12	-1	0	1210	12	S5	3800	-1	23E3
178	BDYS1	10	GEØM4	0	-4	12	-1	0	1310	13	S5	3900	-1	23E4
263	CAERØ1	4	EDT	0	16	16	39	1	3002	30	S5	6400	-1	33D5
301	CAERØ2	4	EDT	0	12	16	39	1	4301	43	S5	6400	-1	42A1
302	CAERØ3	4	EDT	0	16	16	39	1	4401	44	S5	6400	-1	42A2
303	CAERØ4	4	EDT	0	16	16	39	1	4501	45	S5	6400	-1	42A3
309	CAERØ5	4	EDT	0	16	16	39	1	5001	50	S5	6400	-1	42B3
224	CAXIF2	8	GEØM2	0	8	8	877	1	2108	21	S1	4200	0	32C2
225	CAXIF3	8	GEØM2	0	8	8	877	1	2208	22	S1	4300	0	32C3
226	CAXIF4	8	GEØM2	0	8	8	877	1	2308	23	S1	4400	0	32C4
180	CBAR	8	GEØM2	0	8	20	73	1	2408	24	S1	200	-1	23E6
275	CBARAØ*	8	GEØM2	0	9	13	-1	1	4001	40	S5	6100	-1	41A5
146	CCØNEAX	15	AXIC	0	4	8	645	1	2315	23	S3	1460	0	22E2
69	CDAMP1	8	GEØM2	0	4	12	337	1	201	2	S1	3620	-1	13B3
70	CDAMP2	8	GEØM2	0	4	12	397	1	301	3	S1	3623	-1	13B4
71	CDAMP3	8	GEØM2	0	4	8	37	0	401	4	S1	3675	-1	13B5
72	CDAMP4	8	GEØM2	0	4	8	409	0	501	5	S1	3698	-1	13B6
107	CDUM1	8	GEØM2	0	8	24	925	1	6108	61	S5	1000	0	21C5
108	CDUM2	8	GEØM2	0	8	24	925	1	6208	62	S5	1100	0	21C6
109	CDUM3	8	GEØM2	0	8	24	925	1	6308	63	S5	1200	0	21D1
110	CDUM4	8	GEØM2	0	8	24	925	1	6408	64	S5	1300	0	21D2
111	CDUM5	8	GEØM2	0	8	24	925	1	6508	65	S5	1400	0	21D3
112	CDUM6	8	GEØM2	0	8	24	925	1	6608	66	S5	1500	0	21D4
113	CDUM7	8	GEØM2	0	8	24	925	1	6708	67	S5	1600	0	21D5
114	CDUM8	8	GEØM2	0	8	24	925	1	6808	68	S5	1700	0	21D6
115	CDUM9	8	GEØM2	0	8	24	925	1	6908	69	S5	1800	0	21E1
73	CELAS1	8	GEØM2	0	4	12	337	1	601	6	S1	3620	-1	13C1
74	CELAS2	8	GEØM2	0	4	12	417	1	701	7	S1	3800	-1	13C2
75	CELAS3	8	GEØM2	0	4	8	37	0	801	8	S1	3676	-1	13C3
76	CELAS4	8	GEØM2	0	4	8	409	0	901	9	S1	3699	-1	13C4
209	CFLUID2	15	AXIC	0	8	8	845	1	8515	85	S4	3500	-1	31E5
210	CFLUID3	15	AXIC	0	8	8	853	1	8615	86	S4	3600	-1	31E6
211	CFLUID4	15	AXIC	0	8	8	861	1	8715	87	S4	3700	-1	32A1
232	CHBDY	8	GEØM2	0	9	17	909	1	4208	42	S1	5000	-1	32D4
219	CHEXA1	8	GEØM2	0	16	16	531	1	5708	57	S4	4300	-1	32B3
220	CHEXA2	8	GEØM2	0	16	16	531	1	5808	58	S4	4400	-1	32B4
251	CIHEX1	8	GEØM2	0	12	16	951	1	7108	71	S5	5000	-1	33B5
252	CIHEX2	8	GEØM2	0	24	28	951	1	7208	72	S5	5100	-1	33B6
253	CIHEX3	8	GEØM2	0	36	40	951	1	7308	73	S5	5200	-1	33C1

EXECUTIVE PREFACE MODULE IFP (INPUT FILE PROCESSOR)

Table 2(d). Bulk Data Cards Processed by IFP, Sorted Alphabetically by Card Name.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ø IJHK
65	CMASS1	8	GEØM2	0	4	12	337	1	1001	10	S1	3620	-1	13A5
66	CMASS2	8	GEØM2	0	4	12	397	1	1101	11	S1	3623	-1	13A6
67	CMASS3	8	GEØM2	0	4	8	37	0	1201	12	S1	3674	-1	13B1
68	CMASS4	8	GEØM2	0	4	8	409	0	1301	13	S1	3697	-1	13B2
258	CNGRNT	8	GEØM2	0	-4	16	-1	0	5008	50	S1	5245	-1	33C6
168	CØNCT	10	GEØM4	0	-4	12	-1	0	210	2	S5	2900	-1	23C6
167	CØNCT1	10	GEØM4	0	-4	20	-1	0	110	41	S5	2800	-1	23C5
63	CØNM1	8	GEØM2	0	8	28	349	1	1401	14	S1	3580	-1	13A3
64	CØNM2	8	GEØM2	0	8	20	377	1	1501	15	S1	3600	-1	13A4
47	CØNRØD	8	GEØM2	0	8	12	277	1	1601	16	S1	3260	-1	12C5
6	CØRD1C	1	GEØM1	0	4	8	37	0	1701	17	S1	600	-1	11A6
5	CØRD1R	1	GEØM1	0	4	8	37	0	1801	18	S1	500	-1	11A5
7	CØRD1S	1	GEØM1	0	4	8	37	0	1901	19	S1	700	-1	11B1
9	CØRD2C	1	GEØM1	0	12	16	45	1	2001	20	S1	900	-1	11B3
8	CØRD2R	1	GEØM1	0	12	16	45	1	2101	21	S1	800	-1	11B2
10	CØRD2S	1	GEØM1	0	12	16	45	1	2201	22	S1	1000	-1	11B4
60	CQDMEM	8	GEØM2	0	8	12	325	1	2601	26	S1	3460	-1	12E6
249	CQDMEM1	8	GEØM2	0	8	12	325	0	2008	20	S1	3460	-1	33B3
259	CQDMEM2	8	GEØM2	0	8	12	325	0	5308	53	S1	3460	-1	33D1
261	CQDMEM3	8	GEØM2	0	8	12	325	0	5408	54	S1	3460	-1	33D3
59	CODPLT	8	GEØM2	0	8	12	325	1	2701	27	S1	3460	-1	12E5
280	CQUADTS*	8	GEØM2	0	8	20	1045	1	4108	41	S4	2020	-1	41B4
57	CQUAD1	8	GEØM2	0	8	12	325	1	2801	28	S1	3460	-1	12E3
58	CQUAD2	8	GEØM2	0	8	12	325	1	2901	29	S1	3460	-1	12E4
297	CRIGDR	10	GEØM4	-2	4	8	37	1	8210	82	S3	6000	-1	41E3
279	CRIGD1	10	GEØM4	-2	-3	48	-1	1	5310	53	S3	2010	-1	41B3
284	CRIGD2	10	GEØM4	-2	-4	48	-1	1	5410	54	S3	2060	-1	41C2
298	CRIGD3	10	GEØM4	-2	-3	48	-1	1	8310	83	S3	7000	-1	41E4
48	CRØD	8	GEØM2	0	4	8	37	0	3001	30	S1	3281	-1	12C6
61	CSHEAR	8	GEØM2	0	8	12	337	1	3101	31	S1	3540	-1	13A1
227	CSLØT3	8	GEØM2	0	8	8	877	1	4408	44	S1	4500	0	32C5
228	CSLØT4	8	GEØM2	0	8	16	877	1	4508	45	S1	4600	0	32C6
217	CTETRA	8	GEØM2	0	8	8	337	1	5508	55	S4	4100	-1	32B1
104	CTØRDRG	8	GEØM2	0	4	12	750	1	1908	19	S4	1040	-1	21C2
287	CTRAPAX	15	AXIC	-2	4	8	325	1	7042	74	S3	2040	0	41C5
80	CTRAPRG	8	GEØM2	0	8	12	737	1	1808	18	S4	800	-1	13D2
54	CTRBSC	8	GEØM2	0	8	12	313	1	3201	32	S1	3360	-1	12D6
285	CTRIAAX	15	AXIC	-2	4	8	313	1	7012	70	S3	2111	0	41C3
52	CTRIA1	8	GEØM2	0	8	12	313	1	3301	33	S1	3360	-1	12D4
53	CTRIA2	8	GEØM2	0	8	12	313	1	3401	34	S1	3360	-1	12D5
79	CTRIARG	8	GEØM2	0	8	12	738	1	1708	17	S4	790	-1	13D1
282	CTRIATS*	8	GEØM2	0	8	20	1047	1	5908	59	S4	2021	-1	41B6
291	CTRIM6	8	GEØM2	-2	12	16	913	1	6101	81	S1	6101	-1	41D3
56	CTRMEM	8	GEØM2	0	8	12	313	1	3501	35	S1	3360	-1	12E2
55	CTRPLT	8	GEØM2	0	8	12	313	1	3601	36	S1	3360	-1	12E1
293	CTRPLT1	8	GEØM2	-2	12	16	913	1	6301	83	S1	6301	-1	41D5
299	CTRSHL	8	GEØM2	-2	12	16	913	1	7501	75	S1	7501	-1	41E5
49	CTUBE	8	GEØM2	0	4	8	37	0	3701	37	S1	3282	-1	12D1
62	CTWIST	8	GEØM2	0	8	12	337	1	3801	38	S1	3540	-1	13A2
50	CVISC	8	GEØM2	0	4	8	37	0	3901	39	S1	3283	-1	12D2
218	CWEDGE	8	GEØM2	0	8	8	525	1	5608	56	S4	4200	-1	32B2
257	CYJØIN	10	GEØM4	0	-4	16	-1	0	5210	52	S1	5240	-1	33C5
182	DAREA	7	DYNAMICS	0	4	8	101	0	27	17	S3	1820	0	31A2
311	DAREAS	7	DYNAMICS	0	-4	9	1080	0	9027	90	S5	3300	-1	42B5
81	DEFØRM	4	EDT	0	4	8	157	0	104	1	S1	2500	-1	13D3
183	DELAY	7	DYNAMICS	0	4	8	101	0	37	18	S3	1820	0	31A3
312	DELAYS	7	DYNAMICS	0	-4	9	1080	0	9137	91	S5	3300	-1	42B6
123	DLØAD	7	DYNAMICS	0	4	8	-1	1	57	5	S3	4060	0	22A3
119	DMI	12	PØØL	0	-4	16	-1	0	0	0	S2	1190	0	21E5
221	DMIAX	14	MATPØØL	0	-4	9	-1	0	214	2	S4	4500	-1	32B5

MODULE FUNCTIONAL DESCRIPTIONS

Table 2(e). Bulk Data Cards Processed by IFP Sorted Alphabetically by Card Name.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ø IJHK
120	DMIG	14	MATPØØL	0	-4	12	-1	0	114	1	S2	1200	0	21E6
184	DPHASE	7	MATPØØL	0	4	8	101	0	77	19	S3	1820	0	31A4
313	DPHASES	7	DYNAMICS	0	-4	9	1080	0	9277	92	S5	3300	-1	42C1
143	DSFACT	3	MPT	0	4	8	-1	1	53	10	S3	1430	0	22D5
200	DTI	12	PØØL	0	-4	16	-1	0	0	0	S2	2000	0	31D2
86	EIGB	7	DYNAMICS	0	14	18	469	1	107	1	S2	850	0	13E2
87	EIGC	7	DYNAMICS	0	-4	10	-1	1	207	2	S2	870	0	13E3
158	EIGP	7	DYNAMICS	0	4	8	561	0	257	4	S1	1580	0	23B2
85	EIGR	7	DYNAMICS	0	14	18	469	1	307	3	S2	850	0	13E1
124	EPØINT	7	DYNAMICS	0	-4	9	794	0	707	7	S4	1050	0	22A4
274	FLFACT	4	EDT	0	-4	16	-1	1	4102	41	S3	1415	-1	41A4
222	FLSYM	15	AXIC	0	4	10	826	0	9115	91	S4	4600	-1	32B6
272	FLUTTER	4	EDT	0	10	14	1005	1	3902	39	S5	5900	-1	41A2
18	FØRCE	9	GEØM3	0	8	12	109	0	4201	42	S1	1800	1	11C6
20	FØRCE1	9	GEØM3	0	8	12	121	0	4001	40	S1	2000	-1	11D2
22	FØRCE2	9	GEØM3	0	8	12	133	0	4101	41	S1	2200	-1	11D4
156	FØRCEAX	15	AXIC	0	-4	13	109	0	2115	21	S3	1560	0	23A6
214	FREETPT	15	AXIC	0	4	8	834	0	9015	90	S4	4000	-1	32A4

MODULE FUNCTIONAL DESCRIPTIONS

Table 2(f). Bulk Data Cards Processed by IFP, Sorted Alphabetically by Card Name.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ø IJHK
126	FREQ	7	DYNAMICS	0	4	8	-1	1	1307	13	S3	1260	0	22A6
125	FREQ1	7	DYNAMICS	0	4	8	705	0	1007	10	S1	1250	0	22A5
166	FREQ2	7	DYNAMICS	0	4	8	705	0	1107	11	S1	1660	0	23C4
206	FSLIST	15	AXIC	0	-4	10	-1	0	8215	82	S4	3200	-1	31E2
28	GENEL	8	GEØM2	0	-4	9	-1	1	4301	43	S3	2800	-1	11E4
186	GNEW*	10	GEØM4	0	9	9	993	0	1410	14	S5	4000	-1	31A6
26	GRAV	9	GEØM3	0	8	12	165	1	4401	44	S1	2600	1	11E2
2	GRDSET	1	GEØM1	0	4	12	13	2	0	0	S3	200	-1	11A2
1	GRID	1	GEØM1	0	4	12	1	1	4501	45	S3	100	-1	11A1
205	GRIDB	15	AXIC	0	8	12	1	1	8115	81	S4	3100	-1	31E1
229	GRIDF	15	AXIC	0	4	8	885	1	1215	12	S1	4700	0	32D1
230	GRIDS	15	AXIC	0	4	8	893	1	1315	13	S1	4800	0	32D2
187	GTRAN	10	GEØM4	0	5	9	993	0	1510	15	S5	4100	-1	31B1
308	GUST	5	DIT	0	4	8	165	0	1005	10	S5	7600	-1	42B2
247	INPUT	11	GEØM5	0	17	17	-1	0	1310	13	S5	5	-1	33B1
84	LØAD	9	GEØM3	0	4	8	-1	1	4551	61	S3	4060	1	13D6
171	LØADC	10	GEØM4	0	-8	16	-1	0	500	5	S5	3200	-1	23D3
90	MATS1	3	MPT	0	4	16	545	1	503	5	S4	900	-1	13E6
91	MATT1	3	MPT	0	4	16	545	1	703	7	S4	900	0	21A1
102	MATT2	3	MPT	0	4	16	525	1	803	8	S4	1020	-1	21B6
189	MATT3	3	MPT	0	4	16	525	1	1503	15	S4	1020	-1	31B3
237	MATT4	3	MPT	0	4	8	37	1	2303	23	S1	3950	-1	32E3
238	MATT5	3	MPT	0	4	8	37	1	2404	24	S1	4050	-1	32E4
77	MAT1	3	MPT	0	4	20	429	1	103	1	S1	3860	0	13C5
78	MAT2	3	MPT	0	8	20	449	1	203	2	S1	3880	0	13C6
122	MAT3	3	MPT	0	4	20	449	1	1403	14	S4	1220	-1	22A2
234	MAT4	3	MPT	0	4	8	198	1	2103	21	S1	3900	-1	32D6
235	MAT5	3	MPT	0	4	8	198	1	2203	22	S1	4000	-1	32E1
271	MKAERO1	4	EDT	0	16	16	805	0	3802	38	S5	5800	-1	41A1
270	MKAERO2	4	EDT	0	4	8	805	0	3702	37	S5	5700	-1	33E6
157	MØMAX	15	AXIC	0	-4	13	109	0	3815	38	S3	1560	0	23B1
19	MØMENT	9	GEØM3	0	8	12	109	0	4801	48	S1	1800	1	11D1
21	MØMENT1	9	GEØM3	0	8	12	121	0	4601	46	S1	2000	-1	11D3
23	MØMENT2	9	GEØM3	0	8	12	133	0	4701	47	S1	2200	-1	11D5
17	MPC	10	GEØM4	0	4	8	-1	0	4901	49	S3	1700	-1	11C5
83	MPCADD	10	GEØM4	0	4	8	-1	1	4891	60	S3	4020	1	13D5
149	MPCAX	15	AXIC	0	4	8	-1	0	4015	40	S3	1490	0	22E5
176	MPCS	10	GEØM4	0	-8	12	-1	0	1110	11	S5	3700	-1	23E2
127	NØLIN1	7	DYNAMICS	0	8	12	725	0	3107	31	S1	1270	0	22B1
128	NØLIN2	7	DYNAMICS	0	8	12	725	0	3207	32	S1	1280	0	22B2
129	NØLIN3	7	DYNAMICS	0	8	12	725	0	3307	33	S1	1290	0	22B3
130	NØLIN4	7	DYNAMICS	0	8	12	725	0	3407	34	S1	1290	0	22B4
15	ØMIT	10	GEØM4	0	4	8	37	0	5001	50	S1	1400	-1	11C3
150	ØMITAX	15	AXIC	0	4	8	337	0	4315	43	S3	1500	0	22E6
92	ØMIT1	10	GEØM4	0	-4	9	-1	0	4951	63	S3	3981	-1	21A2
248	ØUTPUT	11	GEØM5	0	17	17	-1	0	1410	14	S5	5	-1	33B2
264	PAERO1	4	EDT	0	4	8	803	1	3102	31	S5	6500	-1	33D6
304	PAERO2	4	EDT	0	16	16	1162	1	4601	46	S5	6510	-1	42A4
305	PAERO3	4	EDT	0	4	24	801	1	4701	47	S5	6520	-1	42A5
306	PAERO4	4	EDT	0	-4	8	-1	1	4801	48	S5	6530	-1	42A6
310	PAERO5	4	EDT	0	-4	8	-1	1	5101	51	S5	7700	-1	42B4
82	PARAM	6	PVT	0	-5	16	-1	2	0	0	S3	3960	0	13D4
181	PBAR	2	EPT	0	4	24	621	1	52	20	S1	300	-1	31A1
147	PCØNEAX	2	EPT	0	4	28	653	1	152	19	S3	1470	0	22E3
45	PDAMP	2	EPT	0	4	8	269	0	202	2	S1	3200	-1	12C3
116	PDUM1	2	EPT	0	4	24	925	1	6102	61	S5	1900	0	21E2
117	PDUM2	2	EPT	0	4	24	925	1	6202	62	S5	2000	0	21E3
118	PDUM3	2	EPT	0	4	24	925	1	6302	63	S5	2100	0	21E4
159	PDUM4	2	EPT	0	4	24	925	1	6402	64	S5	2200	0	23B3

EXECUTIVE PREFACE MODULE IFP (INPUT FILE PROCESSOR)

Table 2(n). Bulk Data Cards Processed by IFP Sorted Alphabetically by Card Name.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ø I J H K
160	PDUM5	2	EPT	0	4	24	925	1	6502	65	S5	2300	0	23B4
161	PDUM6	2	EPT	0	4	24	925	1	6602	66	S5	2400	0	23B5
163	PDUM7	2	EPT	0	4	24	925	1	6702	67	S5	2500	0	23C1
164	PDUM8	2	EPT	0	4	24	925	1	6802	68	S5	2600	0	23C2
165	PDUM9	2	EPT	0	4	24	925	1	6902	69	S5	2700	0	23C3
46	PELAS	2	EPT	0	4	8	497	0	302	3	S1	3240	-1	12C4
236	PHBDY	2	EPT	0	4	8	350	1	2502	25	S1	5100	-1	32E2
254	PIHEX	2	EPT	0	4	12	981	1	7002	70	S5	5300	-1	33C2
185	PLFACT	3	MPT	0	4	8	-1	1	1103	11	S3	1420	-1	31A5
276	PLIMIT	3	MPT	0	-9	14	-1	0	304	3	S5	6200	-1	41A6
24	PLØAD	9	GEØM3	0	8	12	145	0	5101	51	S1	2400	-1	11D6
278	PLØADX	9	GEØM3	0	8	12	1037	0	7001	70	S5	6900	-1	41B2
198	PLØAD1*	9	GEØM3	0	0	0	0	0	6909	69	S9		-1	31C6
199	PLØAD2	9	GEØM3	0	-4	9	774	0	6802	68	S4	1990	-1	31D1
255	PLØAD3	9	GEØM3	0	8	12	949	0	7109	71	S5	5400	-1	33C3
11	PLØTEL	8	GEØM2	0	4	8	505	0	5201	52	S1	1111	-1	11B5
44	PMASS	2	EPT	0	4	8	269	0	402	4	S1	3200	-1	12C2
152	PØINTAX	15	AXIC	0	4	8	517	1	4915	49	S3	1520	0	23A2
277	PØPT	3	MPT	0	9	13	1017	0	404	4	S5	6300	-1	41B1
41	PQDMEM	2	EPT	0	4	8	237	0	502	5	S1	3000	-1	12B5
250	PQDMEM1	2	EPT	0	4	8	237	0	2202	22	S1	3000	-1	33B4
260	PQDMEM2	2	EPT	0	4	8	237	0	5302	53	S1	3000	-1	33D2
262	PQDMEM3	2	EPT	0	4	8	237	0	5402	54	S1	3000	-1	33D4
40	PODPLT	2	EPT	0	4	12	257	0	602	6	S1	3020	-1	12B4
281	PQUADTS	2	EPT	0	8	12	277	1	2402	24	S4	2030	-1	41B5
38	PQUAD1	2	EPT	0	4	16	221	1	702	7	S1	2980	-1	12B2
39	PQUAD2	2	EPT	0	4	8	237	0	802	8	S1	3000	-1	12B3
154	PRESAX	15	AXIC	0	4	12	61	0	5215	52	S3	1540	0	23A4
208	PRESPT	15	AXIC	0	4	8	834	0	8415	84	S4	3400	-1	31E4
29	PRØD	2	EPT	0	4	12	165	1	902	9	S1	2900	-1	11E5
42	PSHEAR	2	EPT	0	4	8	237	0	1002	10	S1	3000	-1	12B6
121	PTØDRG	2	EPT	0	4	8	237	0	2102	21	S1	3000	-1	22A1
288	PTRAPAX	2	EPT	-2	4	24	349	1	7052	95	S3	2030	0	41C6
35	PTRBSC	2	EPT	0	4	12	257	1	1102	11	S1	3020	-1	12A5
286	PTRIAAX	2	EPT	-2	4	24	349	1	7032	85	S3	2030	0	41C4
283	PTRIA TS	2	EPT	0	8	12	277	1	2302	23	S4	2030	-1	41C1
33	PTRIA1	2	EPT	0	4	16	221	1	1202	12	S1	2980	-1	12A3
34	PTRIA2	2	EPT	0	4	8	237	0	1302	13	S1	3000	-1	12A4
292	PTRIM6	2	EPT	-2	8	12	802	1	6201	82	S1	6201	-1	41D4
37	PTRMEM	2	EPT	0	4	8	237	0	1402	14	S1	3000	-1	12B1
36	PTRPLT	2	EPT	0	4	12	257	1	1502	15	S1	3020	-1	12A6
294	PTRPLT1	2	EPT	-2	8	20	1089	1	6401	84	S1	6401	-1	41D6
300	PTRSHL	2	EPT	-2	20	24	1105	1	7601	76	S1	7601	-1	41E6
37	PTRMEM	2	EPT	0	4	8	237	0	1402	14	S1	3000	-1	12B1
36	PTRPLT	2	EPT	0	4	12	257	1	1502	15	S1	3020	-1	12A6
30	PTUBE	2	EPT	0	4	12	177	1	1602	16	S1	2920	-1	11E6
43	PTWIST	2	EPT	0	4	8	237	0	1702	17	S1	3000	-1	12C1
31	PVISC	2	EPT	0	4	8	189	0	1802	18	S1	310	-1	12A1
239	QBDY1	9	GEØM3	0	-4	9	774	0	4509	45	S4	1990	-1	32E5
240	QBDY2	9	GEØM3	0	4	8	350	0	4909	49	S1	5150	-1	32E6
233	QHBDY	9	GEØM3	0	9	9	901	0	4309	43	S1	5050	-1	32D5
241	QVECT	9	GEØM3	0	-4	16	-1	0	5009	50	S1	5200	-1	33A1
242	QVØL	9	GEØM3	0	-4	9	774	0	5209	52	S1	1990	-1	33A2
243	RADLST	14	MATPØØL	0	-4	16	-1	0	2014	20	S1	5250	-1	33A3
244	RADMTX	14	MATPØØL	0	-4	8	-1	0	3014	30	S3	1400	-1	33A4
195	RANDPS	7	DYNAMICS	0	4	12	782	0	2107	21	S4	1950	-1	31C3

MODULE FUNCTIONAL DESCRIPTIONS

Table 2(h). Bulk Data Cards Processed by IFP Sorted Alphabetically by Card Name.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
														I J H K
196	RANDT1	7	DYNAMICS	0	4	8	752	0	2207	22	S4	1960	-1	31C4
197	RANDT2*	7	DYNAMICS	0	-4	8	-1	1	2307	23	S9		-1	31C5
170	RELES	10	GEOM4	0	-4	12	-1	0	410	4	S5	3100	-1	23D2
190	RFORCE	9	GEOM3	0	8	12	109	0	5509	55	S1	1900	1	31B4
145	RINGAX	15	AXIC	0	4	12	245	1	5615	56	S3	1450	0	22E1

EXECUTIVE PREFACE MODULE IFP (INPUT FILE PROCESSOR)

(THIS PAGE LEFT BLANK FOR FUTURE USE)

MODULE FUNCTIONAL DESCRIPTIONS

Table 2(f). Bulk Data Cards Processed by IFP, Sorted Alphabetically by Card Name.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ø IJHK
207	RINGFL	15	AXIC	0	4	8	497	1	8315	83	S4	3300	-1	31E3
131	RLØAD1	7	DYNAMICS	0	8	8	337	1	5107	51	S3	1310	0	22B5
132	RLØAD2	7	DYNAMICS	0	8	8	337	1	5207	52	S3	1310	0	22B6
245	SAME*	10	GEØM4	0	-4	10	-1	0	7810	78	S5	4600	-1	33A5
246	SAME1*	10	GEØM4	0	-8	9	-1	0	7910	79	S5	5	-1	33A6
153	SECTAX	15	AXIC	0	4	12	177	1	6015	60	S3	1530	0	23A3
135	SEQEP	7	DYNAMICS	0	4	8	37	0	5707	57	S1	40	-1	22C3
4	SEQGP	1	GEØM1	0	4	8	37	0	5301	53	S1	40	-1	11A4
268	SET1	4	EDT	0	-4	16	-1	0	3502	35	S1	5300	-1	33E4
269	SET2	4	EDT	0	4	8	197	0	3602	36	S5	5600	-1	33E5
231	SLBDY	15	AXIC	0	-4	8	-1	0	1415	14	S1	4900	0	32D3
25	SLØAD	9	GEØM3	0	4	8	157	0	5401	54	S1	2500	-1	11E1
16	SPC	10	GEØM4	0	4	8	101	0	5501	55	S1	1600	-1	11C4
13	SPCADD	10	GEØM4	0	4	8	-1	1	5491	59	S3	4020	-1	11C1
148	SPCAX	15	AXIC	0	4	12	485	0	6215	62	S3	1480	0	22E4
256	SPCD	10	GEØM4	0	4	8	101	0	5110	51	S1	1600	-1	33C4
174	SPCS	10	GEØM4	0	-4	12	-1	0	810	8	S5	3500	-1	23D6
172	SPCSD	10	GEØM4	0	6	9	1000	0	610	6	S5	3300	-1	23D4
173	SPCS1	10	GEØM4	0	-4	12	-1	0	710	7	S5	3400	-1	23D5
12	SPC1	10	GEØM4	0	-4	9	-1	0	5481	58	S3	3980	-1	11B6
266	SPLINE1	4	EDT	0	8	12	42	1	3302	33	S5	6700	-1	33E2
267	SPLINE2	4	EDT	0	12	16	1025	1	3402	34	S5	6800	-1	33E3
307	SPLINE3	4	EDT	0	-4	16	-1	0	4901	49	S5	6850	-1	42B1
105	SPØINT	8	GEØM2	0	-4	9	794	0	5551	49	S4	1050	-1	21C3
175	SS15***								4710	47				23E1
151	SUPAX	15	AXIC	0	4	8	337	0	6415	64	S3	1500	0	23A1
14	SUPØRT	10	GEØM4	0	4	8	37	0	5601	56	S1	1400	-1	11C2
162	TABDMP1	5	DIT	0	-4	16	-1	1	15	21	S2	930	0	23B6
133	TABLED1	5	DIT	0	-4	16	-1	1	1105	11	S2	930	0	22C1
134	TABLED2	5	DIT	0	-4	16	-1	1	1205	12	S2	930	0	22C2
140	TABLED3	5	DIT	0	-4	16	-1	1	1305	13	S2	930	0	22D2
141	TABLED4	5	DIT	0	-4	16	-1	1	1405	14	S2	960	0	22D3
93	TABLEM1	5	DIT	0	-4	16	-1	1	105	1	S2	930	0	21A3
94	TABLEM2	5	DIT	0	-4	16	-1	1	205	2	S2	930	0	21A4
95	TABLEM3	5	DIT	0	-4	16	-1	1	305	3	S2	930	0	21A5
96	TABLEM4	5	DIT	0	-4	16	-1	1	405	4	S2	960	0	21A6
97	TABLES1	5	DIT	0	-4	16	-1	1	3105	31	S2	930	-1	21B1
191	TABRND1	5	DIT	0	-4	16	-1	1	55	25	S2	930	0	31B5
188	TABRNDG	5	DIT	0	4	8	1	0	56	26	S2	1000	0	31C2
27	TEMP	9	GEØM3	0	4	8	157	0	5701	57	S1	2500	-1	11E3
155	TEMPAX	15	AXIC	0	4	8	237	0	6815	68	S3	1550	0	23A5
98	TEMPO	9	GEØM3	0	4	12	269	0	5641	65	S4	980	-1	21B2
295	TEMPO	9	GEØM3	-2	-8	20	-1	0	8509	85	S4	6501	-1	41E1
201	TEMPP1	9	GEØM3	0	-4	10	-1	0	8109	81	S4	2100	-1	31D3
202	TEMPP2	9	GEØM3	0	-4	10	-1	0	8209	82	S4	2200	-1	31D4
203	TEMPP3	9	GEØM3	0	-4	10	-1	0	8309	83	S4	2300	-1	31D5
296	TEMPP4	9	GEØM3	-2	-8	20	-1	0	8609	86	S4	6601	-1	41E2
204	TEMPRB	9	GEØM3	0	-4	10	-1	0	8409	84	S4	2400	-1	31D6
136	TF	7	DYNAMICS	0	8	12	-1	0	6207	62	S1	1360	0	22C4
137	TIC	7	DYNAMICS	0	4	12	713	0	6207	66	S1	1370	0	22C5
314	TICS	7	DYNAMICS	0	-4	9	1153	0	9307	93	S5	3350	-1	42C2
138	TLØAD1	7	DYNAMICS	0	8	8	681	1	7107	71	S3	1380	0	22C6
139	TLØAD2	7	DYNAMICS	0	8	16	689	1	7207	72	S3	1390	0	22D1
169	TRANS	10	GEØM4	0	12	16	45	1	310	3	S5	3000	-1	23D1
142	TSTEP	7	DYNAMICS	0	4	8	-1	1	8307	83	S1	1420	0	22D4
192	UDEF	10	GEØM4	0	-4	16	-1	0	0	0	S5	4300	0	31B6
193	USET	10	GEØM4	0	-4	0	1065	0	110	1	S5	4400	0	31C1
194	USET1	10	GEØM4	0	-4	10	-1	0	210	2	S5	4500	0	31C2
290	VARIAN	4	EPT	0	-4	16	-1	0	4202	42	S3	1410	0	41D2
289	VIEW	2	EPT	0	4	8	326	1	2606	26	S1	5175	0	41D1

4.6 EXECUTIVE PREFACE MODULE IFP3 (INPUT FILE PROCESSOR 3)

4.6.1 Entry Point: IFP34.6.2 Purpose:

1. To interpret Bulk Data cards unique to an axisymmetric conical shell problem.
2. To generate and distribute to data blocks, GEØM1, GEØM2, GEØM3, and GEØM4 data cards for all harmonics specified in the problem.
3. To convert the following input Bulk Data cards to the following output Bulk Data cards.

<u>Input Bulk Data Card Type</u>	<u>Input Data Block</u>	<u>Output Bulk Data Card Type</u>	<u>Output Data Block</u>
AXIC	AXIC	None	
CCØNEAX	AXIC	CCØNE	GEØM2
CTRAPAX	AXIC	CTRAPA	GEØM2
CTRIAAX	AXIC	CTRIAA	GEØM2
FØRCEAX	AXIC	FORCE	GEØM3
FØRCE	AXIC	FORCE	GEØM3
GRAV	AXIC	GRAV	GEØM3
LØAD	AXIC	LOAD	GEØM3
MØMAX	AXIC	MOMENT	GEØM3
MØMENT	AXIC	MOMENT	GEØM3
MPCADD	AXIC	MPCADD	GEØM4
MPCAX	AXIC	MPC	GEØM4
ØMITAX	AXIC	OMIT	GEØM4
PØINTAX	AXIC	MPC GRID	GEØM4
PRESAX	AXIC	PRESAX	GEØM3
RFØRCE	AXIC	RFØRCE	GEØM3
RINGAX	AXIC	SPC GRID	GEØM4 GEØM1
SECTAX	AXIC	MPC GRID	GEØM4 GEØM1
SEQGP	AXIC	SEQGP	GEØM1
SPCADD	AXIC	SPCADD	GEØM4
SPCAX	AXIC	SPC	GEØM4
SUPAX	AXIC	SUPORT	GEØM4
TEMPAX	AXIC	TEMP	GEØM3
TEMPD	AXIC	TEMPD	GEØM3

4.6.3 Calling Sequence

CALL IFP3. IFP3, a Preface module, is called only from the Preface driver SEMINT.

4.6.4 Input Data Blocks

AXIC - Bulk Data Deck cards as output from IFP.

4.6.5 Output Data Blocks

GEØM1 - Grid Point Data.

GEØM2 - Element Connection Data.

GEØM3 - Loading Data.

GEØM4 - Constraint Data.

4.6.6 Parameters

Not applicable to IFP3.

4.6.7 Method

In the following a "card" is defined as the type of card image output by IFP, i.e., with the mnemonic stripped off and a card image, in some cases a modified card image, written on the output data block.

4.6.7.1 Initialization and Overall Method

IFP3 first determines the amount of core available, and then allocates three buffers for the SCRATCH, AXIC, and GEØM data blocks. The AXIC data block is opened for input using PRELØC. At this point the 21 types of input Bulk Data cards are found, one type at a time, on the AXIC data block, using the LØCATE routine. The cards of each type are converted and output to a GEØM data block.

To facilitate the operation, all Bulk Data cards causing output to a particular GEØM data block are handled together. Thus the processing is such that all cards affecting GEØM2 are first converted, and following these all those for GEØM3, then GEØM4, and finally GEØM1.

The superscripts s and c in the following refer to the sine and cosine sets respectively.

4.6.7.2 Conversion of Input Bulk Data Cards to Output Bulk Data Cards for GEOM2.

1. AXIC card

-Input Card-

AXIC	Harm	0
------	------	---

-Output Card-

(None)

The AXIC card supplies Harm which is used to compute the number of harmonics, N.

$$N = \text{Harm} + 1 \quad (1)$$

2. CCØNEAX card

-Input Card-

CCØNEAX	EL-ID	Prop-ID	Ring _A ID	Ring _B ID
---------	-------	---------	----------------------	----------------------

-N Output Cards-

CCØNE _n	EL-ID _n	Prop-ID	Ring _A ID _n	Ring _B ID _n
--------------------	--------------------	---------	-----------------------------------	-----------------------------------

where

$$\text{EL-ID}_n = \text{EL-ID} \times 1000 + n, \quad (2)$$

$$\text{Ring}_A \text{ ID}_n = \text{Ring}_A \text{ ID} + 1000000 \times n, \quad (3)$$

$$\text{Ring}_B \text{ ID}_n = \text{Ring}_B \text{ ID} + 1000000 \times n, \quad (4)$$

for $n = 1, 2, \dots, N$.

MODULE FUNCTIONAL DESCRIPTIONS

3. CTRAPAX card

-Input Card-

CTRAPAX	EL-ID	Prop-ID	Ring _A ID	Ring _B ID	Ring _C ID	Ring _D ID	Theta
---------	-------	---------	----------------------	----------------------	----------------------	----------------------	-------

- N Output Cards -

CTRAPA _n	EL-ID _n	Prop-ID	Ring _A ID _n	Ring _B ID _n	Ring _C ID _n	Ring _D ID _n	Theta
---------------------	--------------------	---------	-----------------------------------	-----------------------------------	-----------------------------------	-----------------------------------	-------

where

$$\begin{aligned} \text{EL-ID}_n &= \text{EL-ID} \times 1000 + n, \\ \text{Ring}_A \text{ ID}_n &= \text{Ring}_A \text{ ID} + 1000000 \times n, \\ \text{Ring}_B \text{ ID}_n &= \text{Ring}_B \text{ ID} + 1000000 \times n, \\ \text{Ring}_C \text{ ID}_n &= \text{Ring}_C \text{ ID} + 1000000 \times n, \\ \text{Ring}_D \text{ ID}_n &= \text{Ring}_D \text{ ID} + 1000000 \times n, \end{aligned}$$

for $n = 1, 2, \dots, N$

4. CTRIAX card

-Input Card-

CTRIAAX	EL-ID	Prop-ID	Ring _A ID	Ring _B ID	Ring _C ID	Theta
---------	-------	---------	----------------------	----------------------	----------------------	-------

- N Output Cards -

CTRIAA _n	EL-ID _n	Prop-ID	Ring _A ID _n	Ring _B ID _n	Ring _C ID _n	Theta
---------------------	--------------------	---------	-----------------------------------	-----------------------------------	-----------------------------------	-------

where

$$\begin{aligned} \text{EL-ID}_n &= \text{EL-ID} \times 1000 + n \\ \text{Ring}_A \text{ ID}_n &= \text{Ring}_A \text{ ID} + 1000000 \times n, \\ \text{Ring}_B \text{ ID}_n &= \text{Ring}_B \text{ ID} + 1000000 \times n, \\ \text{Ring}_C \text{ ID}_n &= \text{Ring}_C \text{ ID} + 1000000 \times n, \end{aligned}$$

for $n = 1, 2, \dots, N$

EXECUTIVE PREFACE MODULE IFP3 (INPUT FILE PROCESSOR 3)

THIS PAGE HAS BEEN LEFT BLANK INTENTIONALLY.

MODULE FUNCTIONAL DESCRIPTIONS

4.6.7.3 Conversion of Input Bulk Data Cards to Output Cards for GEOM3.

1. FØRCE card, RFØRCE card, and MØMENT card

These two cards are output to GEOM3 as input from AXIC.

2. FØRCEAX card and MØMAX card

-Input Cards-

FORCEAX	Set ID	Ring ID	Harm ID	Factor	F_r	F_ϕ	F_z
---------	--------	---------	---------	--------	-------	----------	-------

MØMAX	Set ID	Ring ID	Harm ID	Factor	M_r	M_ϕ	M_z
-------	--------	---------	---------	--------	-------	----------	-------

-Output Cards-

FØRCE	Set ID	Ring ID _H	0	Factor	F_r	F_ϕ	F_z
-------	--------	----------------------	---	--------	-------	----------	-------

MØMENT	Set ID	Ring ID _H	0	Factor	M_r	M_ϕ	M_z
--------	--------	----------------------	---	--------	-------	----------	-------

where

$$\text{Ring ID}_H = \text{Ring ID} + (\text{Harm ID} + 1) \times 1000000 \quad (5)$$

If FØRCE cards and FØRCEAX cards both exist, then the resulting output cards of the FØRCE and FØRCEAX cards are merged in sort on Set ID's. This applies to MØMENT and MØMAX cards also.

3. GRAV card, LØAD card, and TEMPD card

These three cards are output to GEOM3 as they are input from AXIC.

4. PRESAX card

-Input Card-

PRESAX	Set ID	Value	Ring _A ID	Ring _B ID	ϕ_1	ϕ_2
--------	--------	-------	----------------------	----------------------	----------	----------

EXECUTIVE PREFACE MODULE IFP3 (INPUT FILE PROCESSOR 3)

-N Output Cards-

PRESAX _n	Set ID	Value	Ring _A ID _n	Ring _B ID _n	ϕ_1	ϕ_2	n
---------------------	--------	-------	-----------------------------------	-----------------------------------	----------	----------	---

where

$$\text{Ring}_A \text{ ID}_n = \text{Ring}_A \text{ ID} + 1000000 \times n \quad (6)$$

$$\text{Ring}_B \text{ ID}_n = \text{Ring}_B \text{ ID} + 1000000 \times n \quad (7)$$

for $n = 1, 2, \dots, N$.

5. TEMPAX card

-Input Card-

TEMPAX	Set ID	Ring ID	ϕ_1	T_1	Set ID	Ring ID	ϕ_2	T_2
--------	--------	---------	----------	-------	--------	---------	----------	-------

-N x 2 Output Cards -

N-TEMP _{n+1} ^S	Set ID ^S	Ring ID _{n+1}	T_n^S
------------------------------------	---------------------	------------------------	---------

N-TEMP _{n+1} ^C	Set ID ^C	Ring ID _{n+1}	T_n^C
------------------------------------	---------------------	------------------------	---------

where

$$\text{Ring ID}_{n+1} = \text{Ring ID} + 1000000 \times (n+1) \quad (8)$$

$$\text{Set ID}^S = \text{Set ID} + 100000000 \quad (9)$$

$$\text{Set ID}^C = \text{Set ID} + 200000000 \quad (10)$$

for $n = 0, 1, \dots, N-1$.

T_n^S and T_n^C are computed as follows. All TEMPAX cards having the same Set ID and Ring ID are gathered together. The angles ϕ_i and temperatures T_i are arrayed into a two-

dimensional matrix.

ϕ_1	T_1
ϕ_2	T_2
\vdots	\vdots
ϕ_k	T_k

where k is the total number of unique ϕ 's, and the ϕ 's are converted to radians.

The matrix is sorted on the ϕ column, and duplicate angles are removed.

For $n = 0$, we have

$$T_0^C = \frac{1}{4\pi} \sum_{i=1}^k (T_i + T_{i+1}) (\phi_{i+1} - \phi_i), \quad (11)$$

and

$$T_0^S = 0. \quad (12)$$

For $n > 0$

$$T_n^C = \frac{1}{\pi} \sum_{i=1}^k \frac{1}{(\phi_{i+1} - \phi_i)} \left\{ \frac{(T_i \phi_{i+1} - T_{i+1} \phi_i)}{n} (\sin n\phi_{i+1} - \sin n\phi_i) \right. \\ \left. + \frac{T_{i+1} - T_i}{n^2} (\cos n\phi_{i+1} - \cos n\phi_i + n\phi_{i+1} \sin n\phi_{i+1} - n\phi_i \sin n\phi_i) \right\} \quad (13)$$

and

$$T_n^S = \frac{1}{\pi} \sum_{i=1}^k \frac{1}{(\phi_{i+1} - \phi_i)} \left\{ \frac{(T_i \phi_{i+1} - T_{i+1} \phi_i)}{n} (-\cos n\phi_{i+1} + \cos n\phi_i) \right. \\ \left. + \frac{T_{i+1} - T_i}{n^2} (\sin n\phi_{i+1} - \sin n\phi_i - n\phi_{i+1} \cos n\phi_{i+1} + n\phi_i \cos n\phi_i) \right\} \quad (14)$$

where

$$\phi_{k+1} = \phi_1 + 2\pi, \quad (15)$$

and

$$T_{k+1} = T_1. \quad (16)$$

4.6.7.4 Conversion of Input Bulk Data Cards to Output Cards for GEOM4

1. SPCADD card

-Input Card-

SPCADD	Set ID	S1	...	S9	-1	} Open-ended
--------	--------	----	-----	----	----	--------------

-2 Output Cards-

SPCADD ^S	Set ID ^S	S1	...	S9	101	-1
---------------------	---------------------	----	-----	----	-----	----

SPCADD ^C	Set ID ^C	S1	...	S9	102	-1
---------------------	---------------------	----	-----	----	-----	----

where

$$\text{Set ID}^S = \text{Set ID} + 100000000, \quad (17)$$

$$\text{Set ID}^C = \text{Set ID} + 200000000. \quad (18)$$

The following 4 mandatory SPCADD cards are also created and put out on GEOM4 regardless of whether there are or are not SPCADD cards present.

SPCADD	100000101	101	-1
--------	-----------	-----	----

SPCADD	200000102	102	-1
--------	-----------	-----	----

SPCADD	100000000	101	-1
--------	-----------	-----	----

SPCADD	200000000	102	-1
--------	-----------	-----	----

2. MPCADD card

All operations performed for the SPCADD card are performed for the MPCADD card.

EXECUTIVE PREFACE MODULE IFP3 (INPUT FILE PROCESSOR 3)

3. SPCAX card

-Input Card-

SPCAX	Set ID	Ring ID	Harm ID	Comp	Value
-------	--------	---------	---------	------	-------

-3 Output Cards-

SPCADD ^S	Set ID ^S	Set ID	101	-1
---------------------	---------------------	--------	-----	----

SPCADD ^C	Set ID ^C	Set ID	102	-1
---------------------	---------------------	--------	-----	----

where

$$\text{Set ID}^S = \text{Set ID} + 100000000, \quad (19)$$

$$\text{Set ID}^C = \text{Set ID} + 200000000; \quad (20)$$

and

SPC	Set ID	Ring ID _H	Comp	Value
-----	--------	----------------------	------	-------

where

$$\text{Ring ID}_H = \text{Ring ID} + (\text{Harm ID} + 1) \times 1000000. \quad (21)$$

4. MPCAX card

-Input Card-

MPCAX	Set ID	Ring ID	Harm ID	Comp ID	Value	...	-1	-1	-1	-1	Open Ended
	Repeats										

-3 Output Cards-

MPCADD ^S	Set ID ^S	Set ID	101	-1
---------------------	---------------------	--------	-----	----

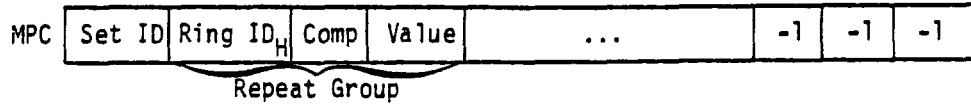
MPCADD ^C	Set ID ^C	Set ID	102	-1
---------------------	---------------------	--------	-----	----

$$\text{Set ID}^S = \text{Set ID} + 100000000, \quad (22)$$

$$\text{Set ID}^C = \text{Set ID} + 200000000. \quad (23)$$

MODULE FUNCTIONAL DESCRIPTIONS

and

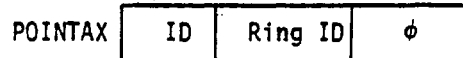


For each 3-word group output,

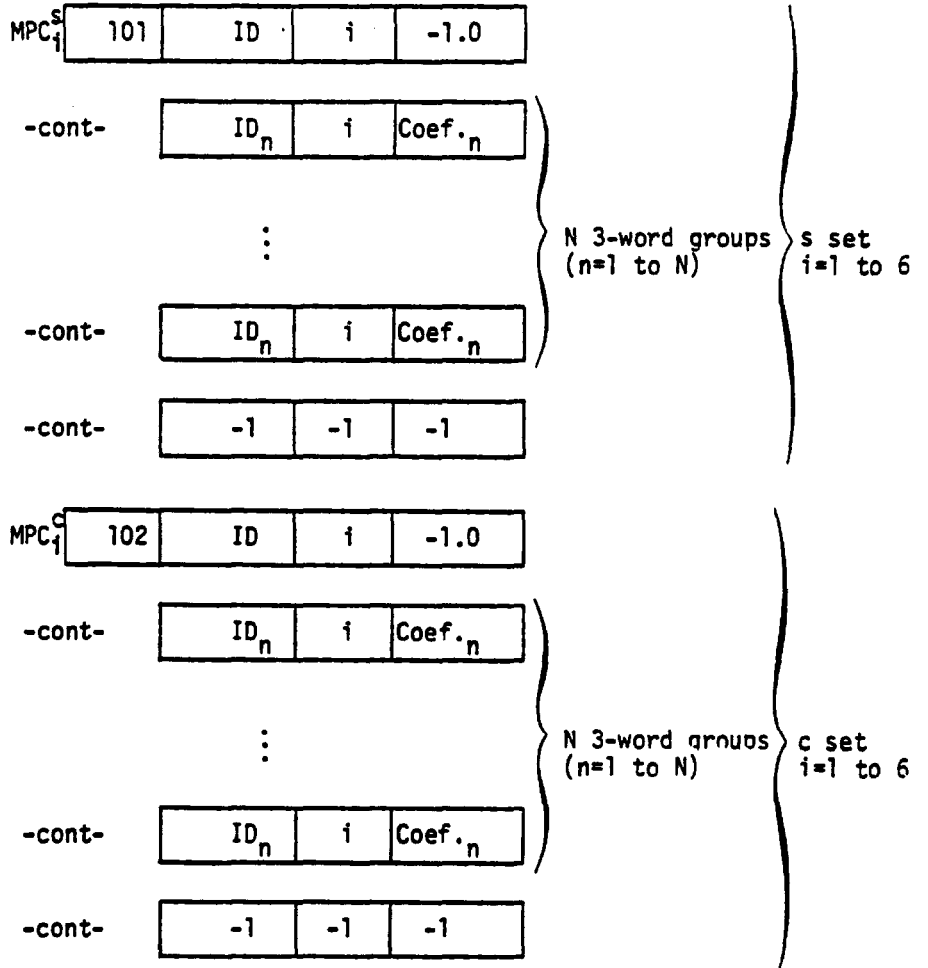
$$\text{Ring ID}_H = \text{Ring ID} + (\text{Harm ID} + 1) \times 1000000. \quad (24)$$

5. POINTAX card (also processed in GEOM1 section)

-Input Card-



-12 Output Cards-



EXECUTIVE PREFACE MODULE IFP3 (INPUT FILE PROCESSOR 3)

where the Coef._n above are defined by:

i	c set- Coef._n	s set- Coef._n
1 (u_r)	$\cos(n\phi)$	$\sin(n\phi)$
2 (u_ϕ)	$\sin(n\phi)$	$-\cos(n\phi), = +1, n = 0$
3 (u_t)	$\cos(n\phi)$	$\sin(n\phi)$
4 (θ_r)	$\sin(n\phi)$	$-\cos(n\phi), = +1, n = 0$
5 (θ_ϕ)	$\cos(n\phi)$	$\sin(n\phi)$
6 (θ_z)	$\sin(n\phi)$	$-\cos(n\phi), = +1, n = 0$

and

$$\text{ID}_n = \text{Ring ID} + (n \times 1000000). \quad (25)$$

6. SECTAX card (also processed in GEOM1 section)

-Input Card-

SECTAX	ID	Ring ID	R	ϕ_1	ϕ_2
--------	----	---------	---	----------	----------

-6 Output Cards-

MPC _i ^s	101	ID	i	-1.0
-------------------------------	-----	----	---	------

-cont-

ID_n	i	Coef._n
---------------	---	------------------

:

-cont-

ID_n	i	Coef._n
---------------	---	------------------

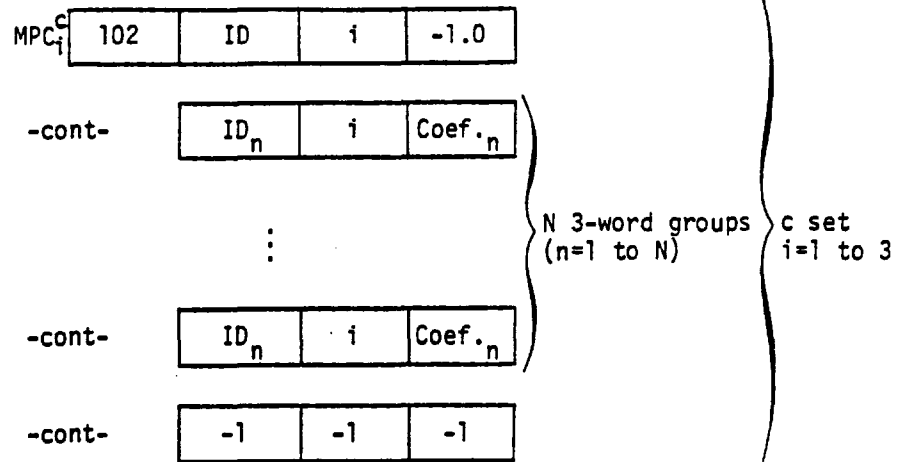
-cont-

-1	-1	-1
----	----	----

N 3-word groups
(n=1 to N)

s set
i=1 to 3

MODULE FUNCTIONAL DESCRIPTIONS



where the Coef._n above are defined by:

i	c set-Coef. _n	s set-Coef. _n
1	$\frac{R}{n} (\sin(n\phi_2) - \sin(n\phi_1))$ or $R(\phi_2 - \phi_1)$ for $n=0$	$-\frac{R}{n} (\cos(n\phi_2) - \cos(n\phi_1))$ or 0 for $n=0$
2	$-\frac{R}{n} (\cos(n\phi_2) - \cos(n\phi_1))$ or 0 for $n=0$	$-\frac{R}{n} (\sin(n\phi_2) - \sin(n\phi_1))$ or $R(\phi_2 - \phi_1)$ for $n=0$
3	$\frac{R}{n} (\sin(n\phi_2) - \sin(n\phi_1))$ or $R(\phi_2 - \phi_1)$ for $n=0$	$-\frac{R}{n} (\cos(n\phi_2) - \cos(n\phi_1))$ or 0, for $n=0$

and

$$ID_n = \text{Ring ID} + (n \times 1000000) \quad (26)$$

7. ØMITAX card

-Input Card-

ØMITAX	Ring ID	Harm ID	Comp
--------	---------	---------	------

-Output card-

ØMIT	Ring ID _H	Comp
------	----------------------	------

$$\text{Ring ID}_H = \text{Ring ID} + (\text{Harm ID} + 1) \times 1000000. \quad (27)$$

8. SUPAX card

-Input Card-

SUPAX	Ring ID	Harm ID	Comp
-------	---------	---------	------

-Output Card-

SUPØRT	Ring ID _H	Comp
--------	----------------------	------

$$\text{Ring ID}_H = \text{Ring ID} + (\text{Harm ID} + 1) \times 1000000.$$

(28)

9. RINGAX card (also processed in GEØM1 section)

-Input Card-

RINGAX	Ring ID	R	Z	Comp
--------	---------	---	---	------

- 2 x N Output Cards -

SPC ^S	101	RingID _n ^I	Dof ^S	0
------------------	-----	----------------------------------	------------------	---

SPC ^C	102	RingID _n ^I	Dof ^C	0
------------------	-----	----------------------------------	------------------	---

where

$$\text{RingID}^I = \text{RingID} + 1000000 \times n$$

(29)

for $n = 1, 2, \dots, N$ and Dof^S and Dof^C are defined by:

ELEMENT	Dof ^S	Dof ^C
CCØNEAX	135	246
CTRIAAX	13	2
CTRAPAX	13	2

MODULE FUNCTIONAL DESCRIPTIONS

4.6.7.5 Conversion of Input Bulk Data Cards to Output Cards for GEOM1.

1. PØINTAX card (also processed in GEØM4 section)

-Input Card-

PØINTAX	ID	Ring ID	ϕ
---------	----	---------	--------

-Output Card-

GRID	ID	0	ϕ	0.0	0.0	0	0	0
------	----	---	--------	-----	-----	---	---	---

2. SECTAX card (also processed in GEØM4 section)

-Input Card-

SECTAX	ID	Ring ID	R	ϕ_1	ϕ_2
--------	----	---------	---	----------	----------

-Output Card-

GRID	ID	0	R	ϕ_1	ϕ_2	0	0	0
------	----	---	---	----------	----------	---	---	---

3. RINGAX card (also processed in GEØM4 section)

-Input Card-

RINGAX	Ring ID	R	Z	Comp
--------	---------	---	---	------

-N Output Cards-

GRID	Ring ID _n	0	R	Z	0.0	0	Comp	0
------	----------------------	---	---	---	-----	---	------	---

where

$$\text{Ring ID}_n = \text{Ring ID} + 1000000 \times n, \quad (30)$$

for $n = 1, 2, \dots, N$.

4.6.7.6 Order of Output for Generated Card Images

IFP3 has the responsibility to output cards in the sort order output by IFP. This causes IFP3 to simultaneously process some cards.

4.6.8 Subroutines

4.6.8.1 Subroutine Name: IFP3B

1. Entry Point: IFP3B
2. Purpose: To process the cards causing output to be created for GEØM4 and GEØM1.
3. Calling Sequence: CALL IFP3B

4.6.9 Design Requirements

The design requirements are:

1. To produce card images equivalent to those put out by IFP.
2. To output those images on GEØM1, GEØM2, GEØM3, GEØM4 in the proper sort and order.

The following COMMON blocks are required for data interface between subroutines IFP3 and IFP3B.

1. COMMON/IFP3LV/

This COMMON block contains local variables common between IFP3 and IFP3B only.

2. COMMON/IFP3BD/

This COMMON block contains constants common between IFP3 and IFP3B and is initialized in the Block Data subprogram AXICBD.

3. COMMON/IFP3ZZ/

This COMMON block defines the beginning of open-core.

4.6.10 Diagnostic Messages

IFP3 error messages are all user-oriented. They pertain to Bulk Data card errors for the axisymmetric conical shell problem, and are output in summary form by IFP3 on the system output file.

EXECUTIVE PREFACE MODULE XGPI (EXECUTIVE GENERAL PROBLEM INITIALIZATION)

4.7 EXECUTIVE PREFACE MODULE XGPI (EXECUTIVE GENERAL PROBLEM INITIALIZATION).

4.7.1 Entry Point: XGPI

4.7.2 Purpose

To translate (compile) a DMAP program into an internal form (the ØSCAR) for use by the NASTRAN Executive System, and, if restarting the problem, to initialize data blocks and named common blocks for proper restart of the problem. See section 2 for format of the ØSCAR.

4.7.3 Calling Sequence

CALL XGPI. XGPI is called only by subroutine SEMINT, the Preface driver.

4.7.4 Method

XGPI calls XGPIBS to initialize data for the module and to initialize the Link Specification table in named common block /XLINK/. Upon return from XGPIBS, XGPI loads the XCSA Executive Table into core from the Problem Tape. If restarting the problem, XGPI modifies table MEDMSK in named common block /XMDMSK/ if necessary. See discussion of the INM table in the description for the XCSA module, in section 4.2.6.2.

XGPI calls XØSGEN to execute phase 1 of the DMAP program compilation. XØSGEN processes the DMAP instructions and generates the skeleton of the Operation Sequence Control Array (ØSCAR). See section 2.4.2.1 for details on the format of the ØSCAR.

XGPI calls XFLØRD to execute phase 2 of the compilation. XFLØRD fills in the ØSCAR entries with the information needed for allocating files (by SFA) when DMAP modules are executed. If restarting a problem, XFLØRD determines which data blocks are needed from the Old Problem Tape to restart the problem and, when necessary, turns on execute flags for DMAP modules to regenerate missing data blocks.

At this point, XGPI terminates the job if any errors were found in compilation; if not, XGPI writes the ØSCAR onto the Data Pool File. If the problem is a restart, XGPI copies the data blocks specified by XFLØRD from the Old Problem Tape onto the Data Pool File and initializes various named common blocks.

MODULE FUNCTIONAL DESCRIPTIONS

XGPI calls ØSCDMP to print the ØSCAR if requested by the user via the DIAG card in the Executive Control Deck. The same DIAG card will also produce a cross-reference to the DMAP program compilation. XGPI then positions the Data Pool File at the first ØSCAR entry in preparation for executing DMAP modules. If checkpointing is requested by the user, the Problem Tape Dictionary is initialized and written on the Problem Tape. XGPI then returns to the calling routine SEMINT.

4.7.5 Subroutines

The following labeled common blocks are used to communicate data and constants among the complex of XGPI subroutines.

1. CØMMØN/XGPIC/ - Contains 30 individual cells containing various flags, integer and BCD constants, and machine dependent data. Also an additional 40 cell array contains a series of required masks.
2. CØMMØN/XGPID/ - Contains restart type codes and approach type codes plus masks and flags required in ØSCAR generation.
3. CØMMØN/XGPI1/ - Defines the beginning of open core for the XGPI module and contains the ØSCAR as it is generated.
4. CØMMØN/XGPI2/ - Contains the MPL table (see Section 2.4.2.2).
5. CØMMØN/XGPI2X/ - Contains the default parameters required by the MPL table.
6. CØMMØN/XGPI3/ - Contains the PVT table (see Section 2.4.2.4) prior to it being written on the Problem Tape.
7. CØMMØN/XGPI4/ - Contains individual DMAP cards as they are output from XRCARD plus the various flags and pointers required to process each DMAP instruction.
8. CØMMØN/XGPI5/ - Contains solution, solution subset, approach and start codes along with data pertaining to DMAP ALTER numbers.
9. CØMMØN/XGPI6/ - Contains various pointers into the Module Execution Decision Table, MED (see Section 1.10).
10. CØMMØN/XGPI7/ - Contains data pertaining to the IFILE Table (see Section 4.7.6.3).

11. COMMON/XGPB/ - Contains pointers into the ICPDPL Table (see section 4.7.6.3).

Further details regarding these common blocks may be obtained from the source listings for XGPB and XGPBS.

4.7.5.1 XGPIDG

1. Entry Points: XGPIDG, XGPIMW.
2. Purposes: For XGPIDG, to write all fatal and non-fatal diagnostic messages for module XGPI. For XGPIMW, to write all non-diagnostic messages for module XGPI.
3. Calling Sequences:

For XGPIDG:

CALL XGPIDG (NCODE,I,J,K)

NCODE - Message code number

I,J,K - Integer values determined by NCODE.

For XGPIMW:

CALL XGPIMW (MSGNO,I,J,A)

MSGNO - Message code number

I,J,A - Integer values determined by MSGNO

4.7.5.2 XGPBS

1. Entry Point: XGPBS.
2. Purpose: To initialize module data and the Link Specification table in named common block /XLINK/ (see section 2.4).
3. Calling Sequence:

CALL XGPBS

4.7.5.3 XPSGEN

1. Entry Point: XPSGEN.

MODULE FUNCTIONAL DESCRIPTIONS

2. Purpose: To execute phase 1 of the compilation by translating the DMAP program into a skeleton Operation Sequence Control Array (ØSCAR).

3. Calling Sequence:

CALL XØSGEN

4.7.5.4 XLNKHD

1. Entry Point: XLNKHD.

2. Purpose: To generate the header section of an ØSCAR entry and for problem re-starts, to determine whether or not to set the ØSCAR entry execute flag.

3. Calling Sequence:

CALL XLNKHD

4.7.5.5 XIPFL

1. Entry Points: XIPFL, XØPFL.

2. Purpose: For XIPFL, to generate the input data block section of an ØSCAR entry. For XØPFL, to generate the output data block section of an ØSCAR entry.

3. Calling Sequences:

CALL XIPFL

CALL XØPFL

4.7.5.6 XPARAM

1. Entry Point: XPARAM.

2. Purpose: To generate the parameter section of an ØSCAR entry.

3. Calling Sequence:

CALL XPARAM

4.7.5.7 XSCNDM

1. Entry Point: XSCNDM.

2. Purpose: To scan all DMAP instructions and return to the calling program each item in an instruction along with its identification (i.e., delimiter, BCD name,

EXECUTIVE PREFACE MODULE XGPI (EXECUTIVE GENERAL PROBLEM INITIALIZATION)

value or end of instruction) as it is requested.

3. Calling Sequence:

CALL XSCNDM

4.7.5.8 XFLØRD

1. Entry Point: XFLØRD.

2. Purpose: To compute the LTU (Last Time Used) and NTU (Next Time Used) values for the input and output sections of ØSCAR entries, and for problem restarts, to determine which data blocks are needed from the Old Problem Tape to restart the problem.

3. Calling Sequence:

CALL XFLØRD

4.7.5.9 XFLDEF

1. Entry Point: XFLDEF.

2. Purpose: To search the Old Problem Tape restart dictionary for a requested data block name and flag name if found; and if not found, and if restart is modified and the calling routine requests it, to attempt to regenerate the data block by turning on the proper ØSCAR execute flags.

3. Calling Sequence:

CALL XFLDEF (NAMI, NAM2, NØFIND)

NAMI, NAM2 - Data block name (8 characters, 4 characters/word).

NØFIND - For input, NØFIND < 0 indicates that the calling routine wants the data block regenerated if it is not in restart dictionary. NØFIND ≥ 0 indicates no regeneration is desired. For output, NØFIND indicates to the calling routine what XFLDEF did. NØFIND < 0, the data block was regenerated. NØFIND = 0, the data block was in the restart dictionary and was flagged for use in restarting the problem. NØFIND > 0, the data block was not found and was not regenerated.

MODULE FUNCTIONAL DESCRIPTIONS

4.7.5.10 DUMPER

1. Entry Point: DUMPER
2. Purpose: To generate a complete and detailed listing of the ØSCAR if requested by the user.
3. Calling Sequence: CALL DUMPER
4. Remark: DUMPER will be executed if no errors have been detected during a DMAP compilation.

4.7.5.11 MPLPRT

1. Entry Point: MPLPRT
2. Purpose: To print the contents of the Module Properties List (MPL) as defined by the Block Data Program XMPLBD. This printout, which occurs whenever a DIAG 31 card exists in the Executive Control Deck, is formatted for easy readability by programmers and users alike.

4.7.5.12 ØSCXRF

1. Entry Point: ØSCXRF
2. Purpose: To generate a cross-reference of the DMAP program is requested by the user.
3. Calling Sequence: CALL ØSCXRF (IØP,AVAIL)
IØP - The number of files to skip over in order to reposition the ØSCAR after completion of the routine.
AVAIL - The starting address of open core in XGPI.
4. Remarks: ØSCXRF will only be executed if no fatal errors have been detected during DMAP compilation.

4.7.5.13 AUTØCK

1. Entry Point: AUTØCK
2. Purpose: To automatically generate CHKPNT instructions when the user elects the PRECHK option in a DMAP program.
3. Calling Sequence: CALL AUTØCK (IADD)
IADD - Open core address of start of output data block section of a functional module ØSCAR entry that is to be automatically CHKPNTed.

CØMMØN / AUTØCM / PREFLG,NNAMES,PRENAM(100)

EXECUTIVE PREFACE MODULE XGPI (EXECUTIVE GENERAL PROBLEM INITIALIZATION)

PREFLG - Denotes the type of PRECHK condition specified.

- 1 - inclusive list of data block names
- 2 - ALL option
- 3 - exclusive list of data block names

If PREFLG is negative, an automatic CHPNT is generated for an EQUIV or PURGE instruction.

NNAMES - The number of data block names in the inclusive or exclusive list.

PRENAM - Data block names.

4.7.5.14 AUTØSV

1. Entry Point: AUTØSV
2. Purpose: To automatically generate SAVE instructions when the user specifies parameter as / S,N,PARAM /.
3. Calling Sequence: CALL AUTØSV
COMMON / AUTØSM / NWØRDS,SAVNAM
NWØRDS - The number of words in the list of parameter names to be saved.
SAVNAM - Pointers to the parameters in the VPS.

4.7.5.15 LINKUP

1. Entry Point: LINKUP
2. Purpose: To generate a linked list of data block, parameter, or module names to be used in a DMAP source program cross-reference listing.
3. Calling Sequence: CALL LINKUP (\$n,NAME)
n - Integer - is statement number to which return is made if open core has been exhausted.
NAME - The BCD name to be added to the list.
COMMON / LNKLIST / ITØP,IBØT,ISN,KIND,ITYPE,MASK1,MASK2,MASK3
ITØP, IBØT - Limits of open core that will contain list.
ISN - DMAP statement number currently being analyzed
KIND - Code for type of name
ITYPE - Module type
MASKi - Masks used for packing words into linked lists

MODULE FUNCTIONAL DESCRIPTIONS

Open core resides in / XGPI1 / .

4.7.5.16 ØUTPAK

1. Entry Point: ØUTPAK
2. Purpose: To prepare DMAP cross-reference data for output; packs leading zeros into statement number and generates output for individual lines.
3. Calling Sequence: CALL ØUTPAK (II,IØUT,ISN)
 - II - Position in output array where DMAP statement number will be packed
 - IØUT - Final output array - BCD
 - ISN - DMAP statement number

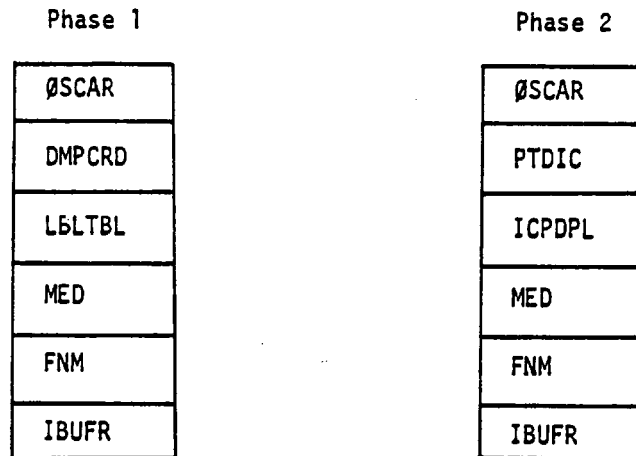
4.7.6 Design Requirements

4.7.6.1 Open Core Layout

The ØSCAR array in named common block / XGPI1 / defines the first location in open core. All other arrays to be put in open core are equivalenced to ØSCAR and are offset from ØSCAR(1) by an amount determined at execution time. This dynamic allocation of arrays in open core optimizes the space available on a given machine, which means that any restrictions on data (except those noted below) are due to the machine's core size and not the program.

EXECUTIVE PREFACE MODULE XGPI (EXECUTIVE GENERAL PROBLEM INITIALIZATION)

The diagrams below show the order in which tables reside in open core during phase 1 and phase 2 of the compilation.



In phase 1 the final sizes of the arrays ØSCAR and L5LTBL are not known until the DMAP program has been completely scanned by XØSGEN. These two arrays request space as needed until it runs out. At this time the user is informed that the DMAP should be shortened or core storage should be increased.

MODULE FUNCTIONAL DESCRIPTIONS

4.7.6.2 Data Necessary for Operation

The Problem Tape provides Executive Tables XCSA, XALTER, and for restarts, XPTDIC (see Section 2.4 for details). Data in named common blocks is initialized by the BLOCK DATA routines XMPLBD, XGPBID and XBSBD or common block data is initialized by routine XGPBS. If PARAM cards are present, a scratch file with the PVT is provided from IFP.

4.7.6.3 Table Formats

1. ØSCAR: Located in named common block /XGP11/. See Section 2.4.2.1 for format.
2. MED, CNM, FNM: Equivalenced to the ØSCAR table. See IS1, INM and JNM table descriptions in XSCA Module Functional Description (4.2.6.2).
3. PTDIC, ICPDPL: Equivalenced to ØSCAR array.

Sample entry:

Word 1	---							
2	DBN							
3	EQ	ET	ER	RU	R	F		
	s	31	30	29	28	17	16	1

<u>Word</u>	<u>Item</u>	<u>Description</u>
1,2	DBN	Data block name (BCD) of the data block from the restart dictionary. Note, a data block name appears only once in the table except for table VPS where it appears twice.
3	R,F	Reel number and file number where the data block is <u>last</u> located on Old Problem Tape. For XVPS there is an entry (the first in PTDIC) which indicates where the first XVPS data block is located on the Old Problem Tape. For purged or not-generated data blocks, R = 0 and F = 0.
	EQ	Equivalence flag. EQ = 0 indicates the data block is equivalenced to another data block.
	ET	End of tape flag. ET = 1 indicates that the data block is split across two reels of the Old Problem Tape.

EXECUTIVE PREFACE MODULE XGPI (EXECUTIVE GENERAL PROBLEM INITIALIZATION)

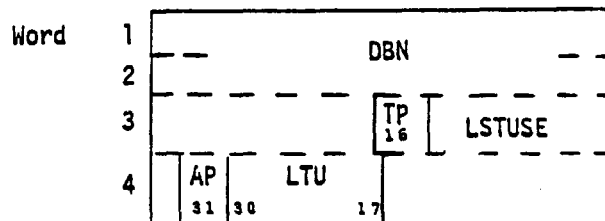
<u>Word</u>	<u>Item</u>	<u>Description</u>
	ER	End of logical record flag. ER = 1 indicates that the complete logical record was written out prior to changing reels when ET = 1.
	RU	Reuse flag. RU = 1 indicates that this data block is to be used to restart the problem.

Table ICPDPL contains all entries from PTDIC which had the RU flag set.

4. MPL: The MPL is located in named common block /XGPI2/. See section 2.4.2.2 for details

5. IØRDNL: Equivalenced to MPL, the IØRDNL table is used in phase 2 when the MPL is no longer needed. Data block names are entered into IØRDNL in the order that they are output from functional modules and IFP.

Sample entry:

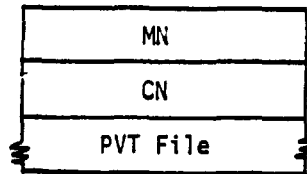


<u>Word</u>	<u>Item</u>	<u>Description</u>
1,2	DBN	Data block name (BCD)
3	LSTUSE	Pointer to input or output section entry of a functional module ØSCAR entry where the data block was last referenced. LSTUSE is used to fill in NTU's (Next Time Used) in ØSCAR entries.
	TP	Tape flag. TP = 1 if the data block was declared TAPE in a FILE DMAP instruction.
	LTU	Last time used. Record number of ØSCAR entry beyond which the data block need not be saved for input.
	AP	Append flag. AP = 1 if the data block was declared APPEND in a FILE DMAP instruction.

6. PVT: Located in named common block /XGPI3/.

MODULE FUNCTIONAL DESCRIPTIONS

Sample entry:

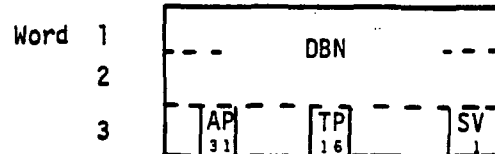


see section 2.4.2.4 for details.

MN - Maximum number of words in PVT (integer).

CN - Current number of words being used (integer).

7. IFILE: Located in named common block /XGPI7/. The purpose of IFILE is to save information from FILE DMAP instructions.



<u>Word</u>	<u>Item</u>	<u>Description</u>
1,2	DBN	Data block name (BCD)
3	SV	SAVE flag
	TP	TAPE flag
	AP	APPEND flag

4.7.6.4 Restrictions on Data

There are only three fixed length tables which might be overflowed by excess user data. These tables are PVT, IFILE and IØRDNL.

4.7.7 Diagnostic Messages

Every effort is made to detect syntactical and logical errors in the DMAP program, and, for restarts, to make sure that the problem is being restarted correctly. All tables are checked for overflow.

The NØGØ flag in named common block /SYSTEM/ is set according to the severity of the errors found. NØGØ = 1 indicates compilation is to be discontinued after phase 2. NØGØ = 2

EXECUTIVE PREFACE MODULE XGPI (EXECUTIVE GENERAL PROBLEM INITIALIZATION)

indicates a serious error and causes XGPI to terminate the program immediately.

See the Diagnostic Message section of the User's Manual (section 6.2) for a detailed discussion of XGPI diagnostic messages. XGPI messages include numbers 1 through 53.

EXECUTIVE PREFACE MODULE UMFEDIT (USER MASTER FILE EDITOR)

4.8 EXECUTIVE PREFACE MODULE UMFEDIT (USER MASTER FILE EDITOR)

4.8.1 Entry Point: UMFEDT

4.8.2 Purpose

To create and manipulate User Master Files.

4.8.3 Calling Sequence

CALL UMFEDT. UMFEDT is called only by SEMINT, the Preface Driver.

4.8.4 Method

UMFEDIT functions as a post-processor to Executive Module XSØRT. Its primary task is to generate a User Master File by repeatedly transferring sorted bulk data decks generated by XSØRT from the New Problem Tape (NPTP) to the New User Master File (NUMF) based on control cards read from the System Input File. See section 2 of the User's Manual for a description of these control cards and how they control the contents of the NUMF.

In addition to creating a User Master File, UMFEDIT is used to list and/or punch Bulk Data Decks from an existing User Master File (UMF). Control cards read from the System Input File also control this process.

4.8.5 Subroutines

The UMFEDIT module has no auxiliary subroutines but uses XRCARD (see section 3.4 for a description).

4.8.6 Design Requirements

1. Open core is defined at /UMFXXX/ and is utilized as follows:

COMMON/UMFXXX/	
GINØ Buffer for UMF	
GINØ Buffer for UMF	
GINØ Buffer for NPTP	
:	
: Unused core	

MODULE FUNCTIONAL DESCRIPTIONS

2. The Block Data subprogram UMFZBD fills /UMFZZZ/.
3. UMFEDIT operates only in the Preface environment. The Bulk Data Deck must have been processed by XSORT and accepted by the Input File Processor (IFP).

4.8.7 Diagnostic Messages

Bad Bulk Data Decks (indicated by ABORT = .TRUE.) will not be accepted by UMFEDIT for inclusion on the NUMF. Subsequent Bulk Data Decks will be included, however, if acceptable.

Other errors detected in UMFEDIT will result in appropriate diagnostic messages being written on the System Output File and termination via PEXIT. These messages are: 1703 through 1721 (active) and 1722 through 1725 (reserved).

4.8.8 Remarks

1. Only a single value of BUFFSIZE is allowed during a NASTRAN execution. Therefore, the BUFFSIZE used during the creation of the UMF tape must be subsequently used during any executions using the tape.

EXECUTIVE MODULE XSFA (EXECUTIVE SEGMENT FILE ALLOCATOR)

4.9 EXECUTIVE MODULE XSFA (EXECUTIVE SEGMENT FILE ALLOCATOR)

4.9.1 Entry Point: XSFA

4.9.2 Purpose

The Segment File Allocator (SFA) manages the data block to physical file relationships throughout a NASTRAN problem. Since, in general, the number of data blocks required for problem solution far exceeds the number of physical files available, allocation of files to data blocks is done dynamically as the module sequence proceeds. The SFA will allocate forward for as many modules as possible. A group of modules allocated by one operation of the SFA is termed a segment.

4.9.3 Calling Sequence

CALL XSFA (\emptyset SCP \emptyset S)

\emptyset SCP \emptyset S - When input to XSFA, this integer argument is the current position (record number) within the \emptyset SCAR. Upon return from XSFA, (1) if allocation was successful, \emptyset SCP \emptyset S is the \emptyset SCAR position of the end of the segment as defined above; (2) if allocation was unsuccessful, the input argument is set negative.

4.9.4 Method

SFA is called by GNFIST (see section 3.3.9 for a description of GNFIST) when GNFIST fails to find the necessary data block names in the FIAT table to construct a complete FIST table for the next operating module (see section 2.4 for descriptions of the FIST and FIAT). The FIST table must contain an entry for each input, output, and scratch data block required by the module. SFA operates by processing the \emptyset SCAR from its present position (next module to be operated) through all remaining modules. Only functional module and output processor \emptyset SCAR entries flagged for execution are processed. The \emptyset SCAR is read and processed by an internal subroutine named XS \emptyset SGN (Serial \emptyset SCAR Sequence Generator) and the S \emptyset S table is formed. From this table all allocation is performed. Following XS \emptyset SGN, another internal subroutine named XCLEAN operates. XCLEAN acts to "clean up" the FIAT and DPL tables prior to the basic allocation procedure. This clean-up involves deleting data block names not needed for subsequent modules, removing equivalence flags if one member of an equivalent

MODULE FUNCTIONAL DESCRIPTIONS

pair is deleted, and closing up gaps in the FIAT caused by deletions. Following operation of XCLEAN, basic allocation begins. Allocation is accomplished during two passes through the SØS table.

Pass one first checks to see if each data block from the SØS is already in the FIAT. If so, it is considered allocated; otherwise the possibility of data block stacking is investigated. Stacking is defined as assigning two or more data blocks to the same physical file by considering their use span. The various use span attributes available are: First-Time-Used (FTU), Next-Time-Used (NTU), and Last-Time-Used (LTU). Data block A may use (be stacked on) the same file as data block B if the first use (FTU) of data block B is subsequent to the last use (LTU) of data block A. Thus many data blocks may be allocated to use the same physical file if their use spans do not overlap. INPUTS may not be stacked. Following pass one, if any data blocks within SØS remain un-allocated, pass two is begun.

Pass two first checks for files within the FIAT currently not assigned to any data block. These files are considered empty and are assigned to the un-allocated data blocks. Once the empty files are exhausted a check is made to determine if the next module to be operated has all its data blocks allocated to files. If the next module (at the least) is allocated, basic allocation is completed. If the next module has not been completely allocated, the second part of pass two will force pooling of sufficient data blocks to provide the necessary empty files for allocation of the remaining data blocks needed for the next module. Pooling is accomplished by flagging the data blocks for copying onto a separate file called the Data Pool File. The Data Pool File will therefore contain many different data blocks where all other files contain only one data block at a time. Data blocks are chosen for pooling by checking the next-time-used (NTU) attribute. The data block with the greatest NTU will be pooled first. Data blocks pooled are considered un-allocated; when they are subsequently re-allocated to their own file, they will be flagged for unpooling.

Following basic allocation, subroutine XPUNP (Pool-Unpool) is operated. All data blocks flagged for pooling are copied to the Pool File followed by all data blocks flagged for unpooling being copied from the Data Pool File. Lastly subroutine XDPH (Data Pool Housekeeper) operates to clean-up and if necessary re-copy the Data Pool File. SFA then returns control to GNFIST with the calling argument set negative if it was un-able to allocate the next module. Figure 1 illustrates the functional flow.

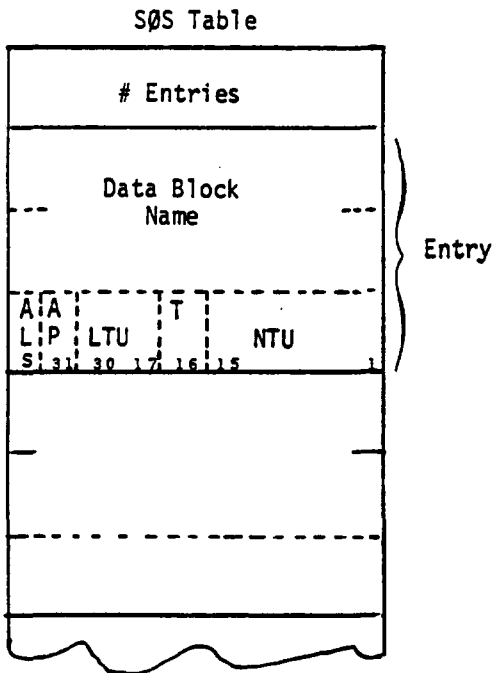
EXECUTIVE MODULE XSFA (EXECUTIVE SEGMENT FILE ALLOCATOR)

4.9.5 Subroutines

XSFA performs basic allocation and its entry point, purpose and calling sequence are given above. Below, it should be noted that XPLEQK and XFILPS are secondary entry points in XPØLCK.

4.9.5.1 Subroutine Name: XSØSGN (Serial ØSCAR Sequence Generator)

1. Entry Point: XSØSGN
2. Purpose: XSØSGN reads the ØSCAR and creates the SØS (Serial ØSCAR Sequence) and MD (Module Descriptor) tables. The SØS table contains the data block names and various attributes, while the MD table contains the ØSCAR sequence numbers and the number of input, output, and scratch data blocks required by each module.



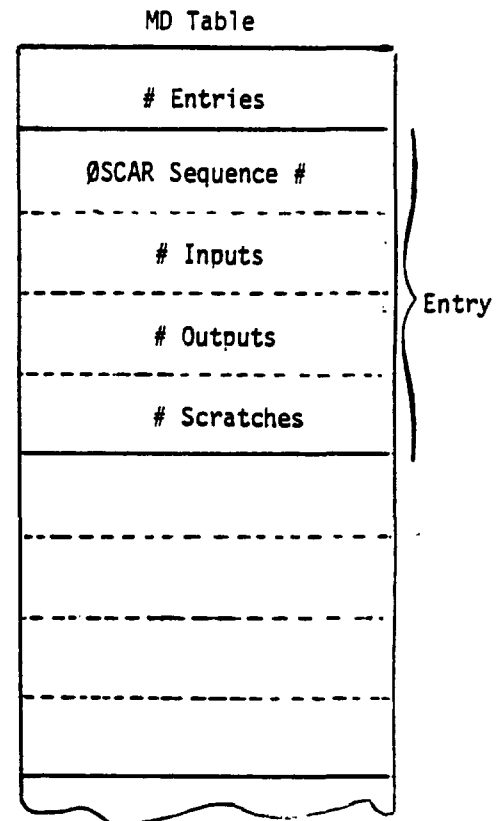
AL = Allocation Flag, set on by SFA when data block is allocated.

AP = Append Flag, set on by module XGPI if data block is to be added to.

LTU = Last-Time-Used, created by XGPI as a data block attribute.

T = Tape Request Flag, set by XGPI to indicate a physical tape file is requested for data block.

NTU = Next-Time-Used, created by XSØSGN as a data block attribute.



= Full word integer values for items

MODULE FUNCTIONAL DESCRIPTIONS

Note: Items created or set by XGPI are passed via ØSCAR.

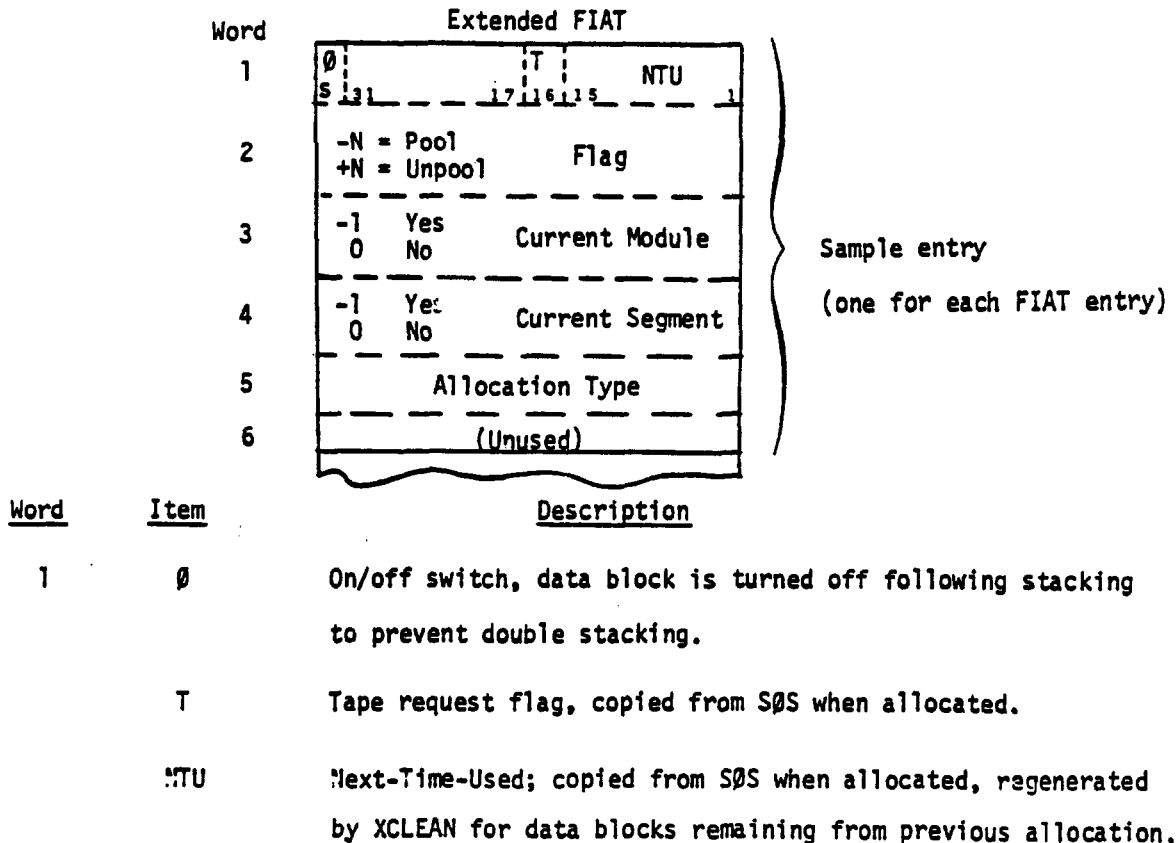
These tables are contained within the /XSFA1/ common block.

3. Calling Sequence: CALL XSØSGN

4.9.5.2 Subroutine Name: XCLEAN (FIAT and DPL Clean-up)

1. Entry Point: XCLEAN

2. Purpose: XCLEAN deletes data block names from the FIAT and DPL when they are no longer needed (their LTU has been reached). Following these deletions, equivalenced data blocks flags within FIAT are checked for continuing validity. If, for example, one member of an equivalenced pair has been deleted, the equivalence flag on the remaining member is removed. Finally, empty spaces within FIAT are removed by closing up the table. XCLEAN also regenerates and stores various parameters into the extended FIAT table. Since this table is non-resident and exists only during SFA operation, values such as NTU and the on/off switches must be restored.



EXECUTIVE MODULE XSFA (EXECUTIVE SEGMENT FILE ALLOCATOR)

<u>Word</u>	<u>Item</u>	<u>Description</u>
2	-N	Flags XPUNP to pool data block, N = number of equivalent names for data block (Equals 1 if data block not equivalenced).
	+N	Flags XPUNP to unpool data block, N = Data Pool File number of data block.
3		Flag set yes if data block allocated for module currently being allocated - cleared between module allocations.
4		Flag set yes if data block allocated for module within current segment - i.e., all data blocks allocated during one SFA operation.
5		Allocation type - 1 = data block match, name already in FIAT 2 = data block stacking, name on file with another 3 = empty file used for data block 5 = data block using file freed by pooling another data block 7 = same as 5 except pooled data block is equivalenced
6		Unused

This table is contained within the /XSFA1/ common block.

4.9.5.3 Subroutine Name: XPUNP (Pool-Unpool)

1. Entry Point: XPUNP
2. Purpose: XPUNP checks the Pool and Unpool flags within the extended FIAT and performs the I/O operations necessary to copy data blocks from their separate files to the Data Pool File and vice-versa. Data block trailers are copied from the FIAT onto the Data Pool File as an additional record during pooling and are replaced from this record during unpooling. All requested pooling operations are performed prior to any unpooling operations. As data blocks are added to the Data Pool File, appropriate entries are added to the Data Pool Dictionary (DPL). The extended FIAT (4.9.5.2) is contained within the /XSFA1/ common block.
3. Calling Sequence: CALL XPUNP

4.9.5.4 Subroutine Name: XDPH (Data Pool Housekeeper)

1. Entry Point: XDPH
2. Purpose: XDPH scans the Data Pool Dictionary (DPL) to determine the number and size of the data blocks on the Data Pool File which have been flagged as no longer needed. If a sufficient quantity of data is flagged, a complete housekeeping operation is performed. This complete process involves re-copying the Data Pool File onto a scratch file while skipping those data blocks flagged for removal. Following the re-copy, the scratch file becomes the new Data Pool File and the old Data Pool File is released as a scratch file. XDPH will also perform a partial housekeeping if a (several) flagged data block(s) appears as the last data block on the Data Pool File. For this partial operation only the DPL entries are modified to release the pool space, no re-copy is necessary.
3. Calling Sequence: CALL XDPH

4.9.5.5 Subroutine Name: XPØLCK (Data Pool Dictionary Check)

1. Entry Point: XPØLCK
2. Purpose: XPØLCK scans the Data Pool Dictionary for a particular data block name. If found, the position within the dictionary and the Data Pool file number are returned to the calling program.
3. Calling Sequence: CALL XPØLCK (DBN1,DBN2,FN,L)

where:

- DBN1, DBN2 - Request data block name (8 characters), 4 characters in each word, left justified and filled with blanks (if necessary).
- FN - $\begin{cases} 0, & \text{if data block not on the Data Pool File.} \\ N, & \text{if data block on the Data Pool File; } N = \text{Data Pool File number.} \end{cases}$
- L - Position of data block entry within the dictionary if data block found.

4.9.5.6 Subroutine Name: XPLEQK (Data Pool Equivalence Check)

1. Entry Point: XPLEQK

EXECUTIVE MODULE XSFA (EXECUTIVE SEGMENT FILE ALLOCATOR)

2. Purpose: XPLEQK scans the Data Pool Dictionary for any equivalence to the called data block name. If found, a copy of the equivalent names is moved from the Data Pool Dictionary to the FIAT.

3. Calling Sequence: CALL XPLEQK (PØØLX,FIATX)

where:

PØØLX - Position of data block entry within the dictionary (see argument L within XPØLCK).

FIATX - Position of same data block entry within the FIAT.

4.9.5.7 Subroutine Name: XFILPS (Data Pool File Positioner)

1. Entry Point: XFILPS

2. Purpose: XFILPS positions the Data Pool File to the beginning of a requested data block. (The Data Pool File is a multi-file file).

3. Calling Sequence: CALL XFILPS (FNEW)

where:

FNEW - The file count of the requested position. The current or old file position is stored in the /XSFA1/ common block.

4.9.6 Design Requirements

1. Open core is used only for GINØ buffers (2 maximum). The open core origin is common block /ESFA/ located following all SFA subroutines.

2. SFA communicates data internal to the module subroutines via common block /XSFA1/. The SØS, MD and extended FIAT tables reside in /XSFA1/.

3. The ØSCAR must be at the entry where allocation is to begin. Following allocation, that initial ØSCAR position is restored.

4. BLOCK DATA subprogram defines the lengths of tables in /XSFA1/.

4.9.7 Diagnostic Messages

Special DMAP routing may cause the warning 3022 message to be printed by SFA.

Following output of this message, SFA flags the data block as allocated (although it will not appear in the FIAT) and continues. Since SFA cannot predict conditional DMAP routing, the data block in question may not be required and the problem will proceed satisfactorily. If the data block is required, the problem will terminate in the requesting module. Under these circumstances the DMAP routing should be studied.

All other error messages generated by SFA are fatal in nature and indicate serious I/O malfunctions or executive table overflow. See section 6 of User's Manual for listing and explanation of these messages. XSFA messages include numbers 1001 through 1004, 1011 through 1014, 1021, 1031 through 1035, 1041, and 1051.

EXECUTIVE MODULE XSFA (EXECUTIVE SEGMENT FILE ALLOCATOR)

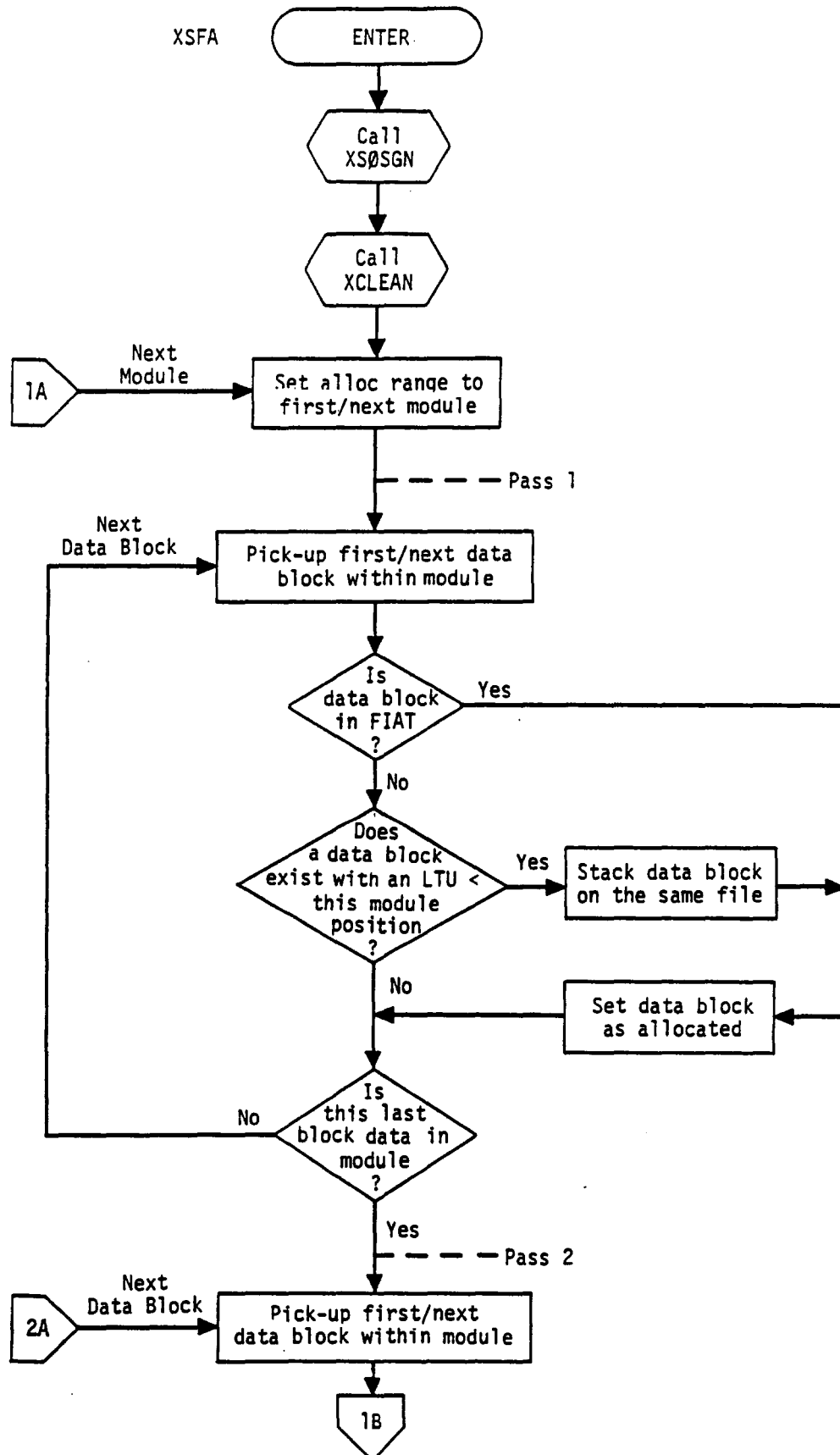


Figure 1.(a) Flowchart for module XSFA.

MODULE FUNCTIONAL DESCRIPTIONS

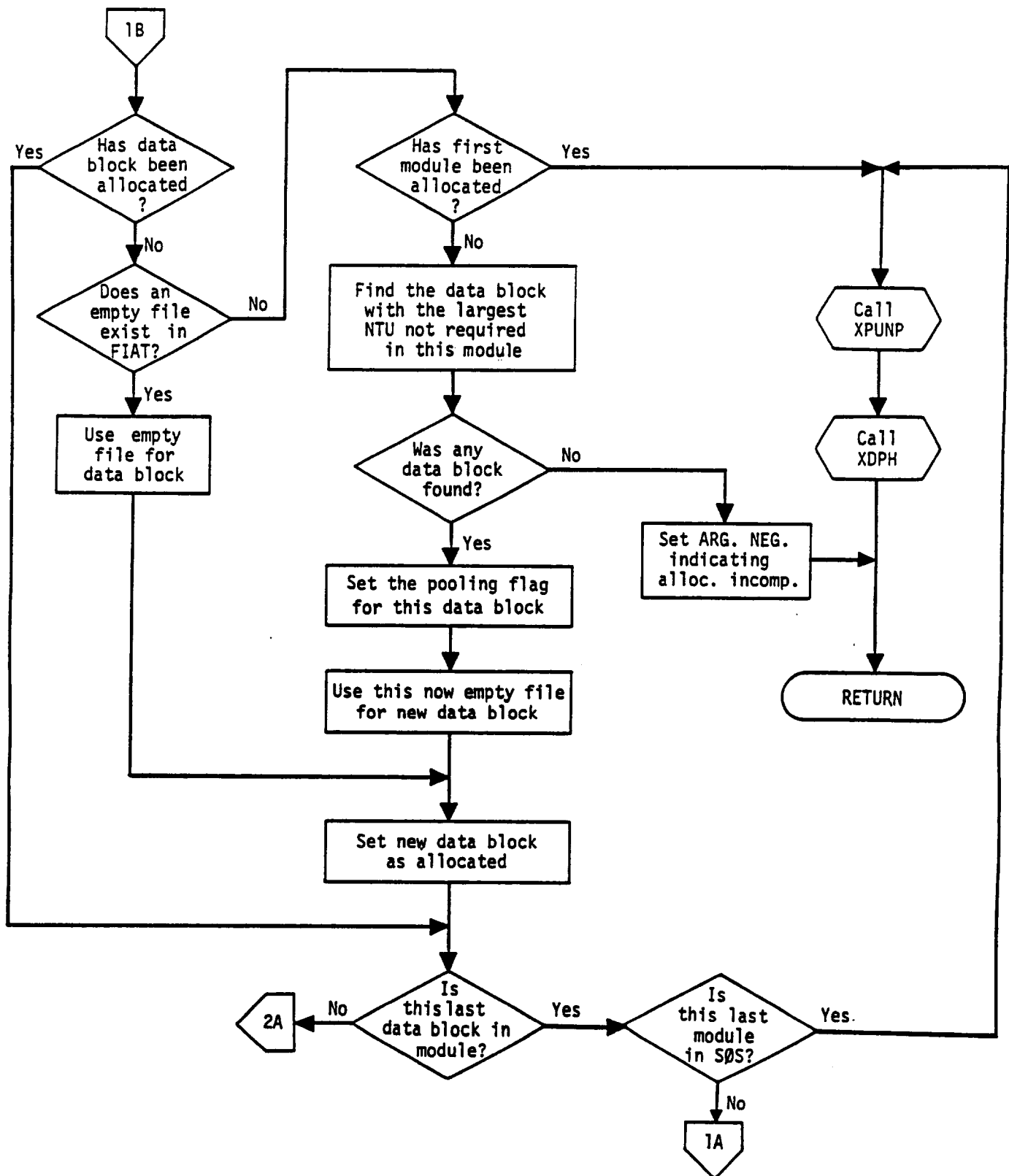


Figure 1.(b) Flowchart for module XSFA.

EXECUTIVE DMAP MODULE CHPNT (CHECKPOINT)

4.10 EXECUTIVE DMAP MODULE CHPNT (CHECKPOINT)

4.10.1 Entry Point: XCHK

4.10.2 Purpose

To save, on the Problem Tape, specified data blocks along with other data necessary for restarting a problem.

4.10.3 DMAP Calling Sequence

CHKPNT DB1,DB2,...,DBN \$

where DB1, DB2,...,DBN ($N \geq 1$) are data blocks to be copied onto the Problem Tape for use in restarting a problem.

4.10.4 Method

The Problem Tape Dictionary (XPTDIC), see section 2.4.2.3, is brought into core from the Problem Tape, and the data block list in the CHPNT OSCAR entry is scanned. Data blocks that have been generated are entered into the local DICT table (see discussion below) along with all data blocks equivalenced to them. Data blocks that are purged, or have not yet been generated, are entered in the local PURGE table. The data blocks are assigned file numbers and are placed in the FDICT table. Data blocks are then copied onto the Problem Tape according to their file number unless the data block is equivalenced and is already in the Problem Tape Dictionary. Core resident data necessary for restart is also written on the Problem Tape as the VPS Executive Table. Entries from FDICT and PURGE are entered into the Problem Tape Dictionary, and the new checkpoint entries are punched for user submittal in the Executive Control Deck upon problem restart. The updated Problem Tape Dictionary is written back on the Problem Tape, and the Problem Tape is positioned to the beginning of the last file (i.e., XPTDIC) in preparation for the next execution of the CHPNT module.

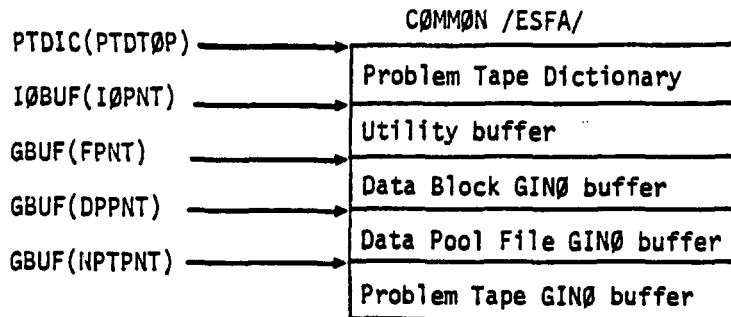
4.10.5 Subroutine

Module XCHK has no auxiliary subroutines.

4.10.6 Design Requirements

4.10.6.1 Open Core Layout

Named common block /ESFA/ defines the start of the open core area. The use of open core is optimized by originating arrays GBUF, PTDIC and IØBUF at /ESFA/ and computing their offset from the origin at execution time. The diagram below shows how the arrays are placed in open core.



4.10.6.2 Data Necessary for Operation

The data blocks, named common blocks and files needed by the CHKPNT module are listed below, along with type of access required (i.e. fetch and/or store data) and reasons for use.

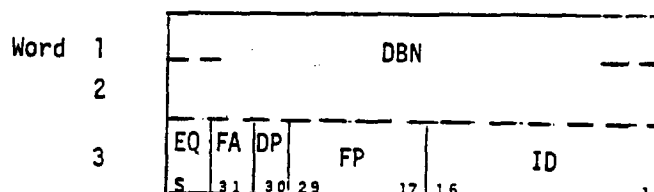
1. Data Pool File - fetch data blocks to be copied onto the Problem Tape.
2. Problem Tape - fetch and store data blocks.
3. Executive Table XPTDIC - fetch and store. Used to generate new checkpoint entries which are then added to XPTDIC.
4. Common /XFIST/ - store temporary entry in FIST for copying data blocks.
5. Common /ØSCENT/ - fetch. CHKPNT ØSCAR entry resides here.
6. Common /XCEITB/, /XVPS/ and /SYSTEM/ - fetch. Contains data to be written in VPS Executive Table.
7. Common /XFIAT/ and Common /XDPL/ - fetch. Contains data block names of data blocks to be copied.
8. Common /ØUTPUT/ - store. Prints out page heading for Checkpoint Dictionary printout.
9. Common /STAPID/ - fetch and store. Used to write Problem Tape ID file on new reel of of the Problem Tape when an end of reel is encountered.

4.10.6.3 Formats of Local Arrays

1. DICT table

The purpose of the DICT table is to hold preliminary data to be used in generating the FDICT table.

Sample DICT entry:



<u>Word</u>	<u>Item</u>	<u>Description</u>
1,2	DBN	BCD name (8 characters, 2 words) of the data block whose status is generated and is to be checkpointed.
3	EQ	Equivalence flag. EQ = 1 indicates all data blocks equivalenced to this data block also reside in DICT table.
	FA	File number assigned flag. FA = 1 indicates that an FDICT entry has been generated for this DBN and that a file number of where the data block will reside on the Problem Tape has been assigned.
	DP	Data pool flag. DP = 1 indicates that the data block is on the Data Pool File.
	FP	Varies according to DP flag. For DP = 1, FP is not used. For DP = 0, FP is a pointer to the FIAT entry containing DBN or equivalenced DBN.
	ID	Varies according to DP flag. For DP = 1, ID is the file number where the data block resides on the Data Pool File. For DP = 0, ID is the file (unit) identifier. For the IBM 7094, ID is the unit control block pointer for the file. For machines using the FØRTRAN GINØ, ID is the FØRTRAN logical unit number assigned to the file.

2. FDICT table

The purpose of the FDICT table is to hold final data to be used in generating XPTDIC entries for data blocks whose status is generated.

MODULE FUNCTIONAL DESCRIPTIONS

Sample FDICT entry:

Word	1										
	2	DBN									
	3	EQ	ET	ER	R			F			
		5	31	30	29		17	16			1

<u>Word</u>	<u>Item</u>	<u>Description</u>
1,2	DBN	BCD name of data block.
	EQ	Equivalence flag. EQ = 1 indicates all data blocks equivalenced to this data block also reside in the FDICT table.
	ET	End-of-tape flag. ET = 1 indicates that the data block is split across two reels of Problem Tape.
	ER	End-of-logical-record flag. ER = 1 indicates that the complete logical record was written out prior to changing reels when ET = 1.
3	R,F	Reel number and file number where the data block will be written on the Problem Tape.

4.10.6.4 Restrictions

XPTDIC cannot be written across two reels of Problem Tape (i.e. a fatal error occurs if an end-of-tape is encountered while writing XPTDIC).

4.10.7 Diagnostic Messages

See Diagnostic Message section of User's Manual (section 6.2) for a detailed discussion of CHPNT module diagnostic messages. XCHK messages include numbers 1101 through 1109.

EXECUTIVE DMAP INSTRUCTURE REPT (REPEAT A GROUP OF DMAP INSTRUCTIONS)

4.11 EXECUTIVE DMAP INSTRUCTION REPT (REPEAT A GROUP OF DMAP INSTRUCTIONS)

4.11.1 Entry Point: XCEI

The XCEI module executes the DMAP control instructions: REPT, EXIT, COND, and JUMP.

4.11.2 Purpose

To repeat a group of DMAP instructions a specified number of times.

4.11.3 DMAP Calling Sequence

REPT n,c \$ or REPT n,p \$

where:

1. n is a BCD name appearing in a LABEL instruction which specifies the location of the beginning of a group of DMAP instructions to be repeated.
2. c is an integer constant which specifies the number of times to repeat the instructions.
3. p is the BCD name of a variable set by a previously executed module specifying the number of times to repeat the instruction.

4.11.4 Example:

```
BEGIN $
.
.
.
LABEL L1 $
MØD1 A / B / V,Y,P1 $
.
.
.
MØDN B / C / V,Y,P2 $
REPT L1,3 $
.
.
.
END $
```

```
BEGIN $
.
.
.
LABEL L1 $
MØD1 A / B / V,Y,P1 $
.
.
.
MØDN C / D / V,Y,P2 $
REPT L1,PLØØP $
.
.
.
END $
```

The DMAP instructions from MØD1 to MØDN will be repeated 3 times (i.e., executed four times). Note that REPT is placed at the end of the group of instructions to be repeated.

4.11.5 Method

Executive table CEITBL in named common block / XCEITB / (See Section 2.4 for format) is searched for the REPT entry. If the second word of the entry has the sign bit on, the left half of the word is a pointer into the Executive Table VPS (See Section 2.4) which contains the current value of the variable loop parameter. This value is then placed in the location formerly occupied by the pointer and the sign bit is turned off. The execution then progresses as though a constant value had been specified. The entry is updated after determining whether to execute the loop again. If the loop is not to be repeated, a return is made to the calling routine. If a repeat of the loop is to be executed, the Problem Tape Dictionary (XPTDIC) is read into core from the Problem Tape, and dictionary entries created inside the loop are deleted. The updated XPTDIC is written back on the Problem Tape. Data blocks that are referenced only inside the loop and which have been declared saved in a FILE DMAP instruction have their status changed to not-generated (i.e., data block trailers within FIAT are cleared and if the data block name appears in the DPL it is removed). The ØSCAR on the Data Pool Tape is positioned to the top of loop and a return is made to the calling routine.

4.11.6 Subroutine

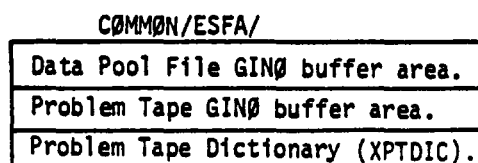
4.11.6.1 Subroutine Name: XCEI.

1. Entry Point: XCEI
2. Purpose: To execute DMAP Control modules REPT, JUMP, CØND, and EXIT as described in the respective Executive DMAP Module Descriptions in this section and in sections 4.12, 4.13 and 4.14.
3. Calling Sequence:
CALL XCEI

4.11.7 Design Requirements

4.11.7.1 Open Core Layout

Named common block /ESFA/ defines the start of the open core area. The following diagram shows the layout.



EXECUTIVE DMAP INSTRUCTION REPT (REPEAT A GROUP OF DMAP INSTRUCTIONS)

4.11.7.2 Data Necessary For Operation

The tables, named common blocks and files needed by the control modules are listed below, along with type of access required (i.e. fetch and/or store data) and reasons for use.

1. Data Pool File - XCEI must re-position the ØSCAR to the correct entry when a transfer is to be executed.
2. Problem Tape - fetch and store the Problem Tape Dictionary when looping.
3. Data Block XPTDIC - fetch and store. The Problem Tape Dictionary must be updated when looping.
4. COMMON /XVPS/ - fetch.
/XVPS/ contains the COND instruction parameter value.
5. COMMON /XCEITB/ - fetch and store.
/XCEITB/ contains control parameters for REPT and EXIT instructions.
6. COMMON /ØSCENT/ - fetch.
/ØSCENT/ contains the Control Module ØSCAR entry.
7. COMMON /XFIAT/ - fetch and store.
Must be updated when looping.
8. COMMON /XDPL/ - fetch and store.
Must be updated when looping.

EXECUTIVE DMAP INSTRUCTION JUMP (UNCONDITIONAL DMAP TRANSFER)

4.12 EXECUTIVE DMAP INSTRUCTION JUMP (UNCONDITIONAL DMAP TRANSFER)

4.12.1 Entry Point: XCEI

4.12.2 Purpose

To alter the normal order of execution of DMAP modules by unconditionally transferring program control to a specified location in the DMAP program.

The normal order of execution of DMAP modules is the sequential order of occurrence of the modules as DMAP instructions in the DMAP program.

4.12.3 DMAP Calling Sequence

JUMP n \$

where n is a BCD name appearing on a LABEL instruction which specifies where control is to be transferred.

4.12.4 Method

If control is being transferred to a previous DMAP module in the ØSCAR (i.e., looping), the Problem Tape Dictionary is read into core from the Problem Tape and dictionary entries created inside the loop are deleted. The updated Problem Tape Dictionary is written back out on the Problem Tape. Data blocks that are referenced only inside the loop and which have not been declared saved in a FILE DMAP instruction have their status changed to not-generated. The ØSCAR is positioned to the specified location. Table CEITBL in named common block /XCEITB/ is searched for REPT entries and the loop count is zeroed if the jump is transferring control from inside the loop to outside the loop. A return is then made to the calling program.

If control is being transferred to a subsequent DMAP module in the ØSCAR, the ØSCAR is positioned to the specified location and a return is made to the calling program.

See description of the Executive DMAP instruction REPT (see section 4.11) for further details.

EXECUTIVE DMAP INSTRUCTION CØND (CONDITIONAL TRANSFER)

4.13 EXECUTIVE DMAP INSTRUCTION CØND (CONDITIONAL TRANSFER)

4.13.1 Entry Point: XCEI

4.13.2 Purpose

To alter the normal order of execution of DMAP modules by conditionally transferring program control to a specified location in the DMAP program.

4.13.3 DMAP Calling Sequence

CØND n,V \$

where:

1. n is a BCD name appearing on a LABEL instruction which specifies where control is to be transferred.
2. V is a BCD name of a variable parameter whose value indicates whether or not to execute the transfer. If $V < 0$, the transfer is executed.

4.13.4 Example

BEGIN \$

.

.

.

CØND L1,K \$

MØDULE1 A/B/V,Y,P1 \$

.

.

.

LABEL L1 \$

MØDULEN X/Y/ \$

.

.

.

END \$

If $K \geq 0$, MØDULE1 is executed. If $K < 0$ control is transferred to L1 and MØDULEN is executed.

4.13.5 Method

The parameter value for the CØND instruction is examined. If the value is greater than or equal to zero, a return is made to the calling routine. If the value is less than zero, the CØND instruction is executed exactly like the JUMP instruction. See description of the Executive DMAP instruction JUMP (section 4.12) for further details.

EXECUTIVE DMAP INSTRUCTION EXIT (TERMINATE DMAP PROGRAM)

4.14 EXECUTIVE DMAP INSTRUCTION EXIT (TERMINATE DMAP PROGRAM)

4.14.1 Entry Point: XCEI

4.14.2 Purpose

To terminate a NASTRAN job.

4.14.3 DMAP Calling Sequence

EXIT c \$

where c is an integer constant which specifies the number of times the instruction is to be ignored before terminating the program. If c = 0 the calling sequence may be shortened to EXIT \$.

4.14.4 Example

BEGIN \$

.

.

.

LABEL L1 \$

MODULE1 A/B/V,Y,P1 \$

.

.

.

EXIT 3 \$

REPT L1,3 \$

.

.

.

END \$

The EXIT instruction will be executed the third time the loop is repeated (i.e., the instructions within the loop will be executed four times).

MODULE FUNCTIONAL DESCRIPTIONS

4.14.5 Method

A determination is made whether or not to terminate the job by examining the loop count of the EXIT entry in named common block /XCEITB/. If the job is to be terminated, routine PEXIT is called; if not, the loop count in the EXIT entry is incremented, and a return is made to the calling program.

See description of the Executive DMAP instruction REPT (section 4.11) for further details.

EXECUTIVE DMAP MODULE SAVE (SAVE VARIABLE PARAMETER VALUES)

4.15 EXECUTIVE DMAP MODULE SAVE (SAVE VARIABLE PARAMETER VALUES)

4.15.1 Entry Point: XSAVE

4.15.2 Purpose

To specify which variable parameter values are to be saved from the preceding functional module for use by subsequent modules.

4.15.3 DMAP Calling Sequence

SAVE V1,V2,...,VN \$

where V1,V2,...,VN ($N \geq 1$) are the BCD names of some or all of the variable parameters which appear in the immediately preceding functional module DMAP instruction. A SAVE DMAP instruction must immediately follow the functional module instruction wherein the parameters being saved are generated.

4.15.4 Method

The specified parameter values are transferred from blank common to the VPS Executive Table located in named common block /XVPS/. See description of the ØSCAR in section 2.4.2.1 for the format of a SAVE ØSCAR entry.

4.15.5 Subroutine

The XSAVE module has no auxiliary subroutines

4.15.6 Design Requirements

SAVE must access blank common and named common blocks /XVPS/ and /ØSCENT/.

EXECUTIVE DMAP MODULE PURGE (EXPLICIT DATA BLOCK PURGE)

4.16 EXECUTIVE DMAP MODULE PURGE (EXPLICIT DATA BLOCK PURGE)

4.16.1 Entry Point: XPURGE

4.16.2 Purpose

To flag a data block so that it will not be allocated to a physical file and so that modules attempting to access it will be signaled.

4.16.3 DMAP Calling Sequence

PURGE DBN1A, DBN2A, D3N3A/PARMA/DBN1B, DBN2B/PARMB \$

Note: The number of data block names (DBN α) prior to each parameter (PARM α) and the number of sets of data block names and parameters in a particular calling sequence is variable.

4.16.4 Input Data Blocks

DBN1A, DBN2A, etc. - Any data block names appearing within the DMAP sequence.

4.16.5 Output Data Blocks

(None specified or permitted)

4.16.6 Parameters

PARMA, etc. - One required for each data block name or set of names.

4.16.7 Method

4.16.7.1 Summary

The data blocks (within the DMAP calling sequence) are purged if the value of the associated parameter is < 0 or if the pointer normally reserved to locate the parameter value in the VPS has been set to -1. If the data blocks are already purged and the parameter value is ≥ 0 , the purged data blocks are unpurged so that they may be subsequently reallocated. If the data blocks are not purged and if the parameter value is ≥ 0 , no action is taken.

4.16.7.2 Functional Flow

PURGE operates by modifying entries within the FIAT (File Allocation Table) and DPL (Data Pool Dictionary). The FIAT contains an upper section (unique part) and a lower section (tail part). Both parts contain entries structured as described in the Executive Table description for the FIAT, section 2.4.1.2. The length of the unique part is defined by the unique files available count in the FIAT header. The tail part is defined as the remainder of the FIAT. The unique part contains one entry for each unique (separate) file available for allocation, and the file ID's within these entries are not modified through a NASTRAN run. The tail part contains entries for stacked files (see description for Executive Module XSFA, section 4.9), purged files, and members of equivalenced sets. An entry within the FIAT is purged by flagging (setting all bits on) its file ID. Therefore, if a data block within the unique part is to be purged, its name is moved to the tail. A data block entry within DPL is purged by removing its entry from the DPL. A data block which is already purged is unpurged by removing the flagged entry from the FIAT so that it may be subsequently allocated to a physical file. Figure 1 illustrates the logic flow.

4.16.8 Design Requirements

1. No open core is required by this module.
2. The ØSCAR record containing the DMAP purge request must reside in the labeled common block /ØSCENT/.
3. The validity of all data block names and controlling parameters is checked during NASTRAN initialization by module XGPI.

4.16.9 Diagnostic Messages

PURGE may produce the following System Fatal messages:

1201, FIAT ØVERFLØW

1202, DPL ØVERFLØW

Both of these messages indicate that the assembled size of the particular table has been exceeded. Although it is unlikely that either message will occur, a study of the erroneous problem's operation along with diagnostic prints of the FIAT and DPL, obtained via the DIAG Executive Control card (see User's Manual, section 2), should indicate some

EXECUTIVE DMAP MODULE PURGE (EXPLICIT DATA BLOCK PURGE)

corrective action. Possible corrective actions include: increasing the basic table size through re-assembly; providing more physical files to the NASTRAN system; and altering the DMAP operations.

MODULE FUNCTIONAL DESCRIPTIONS

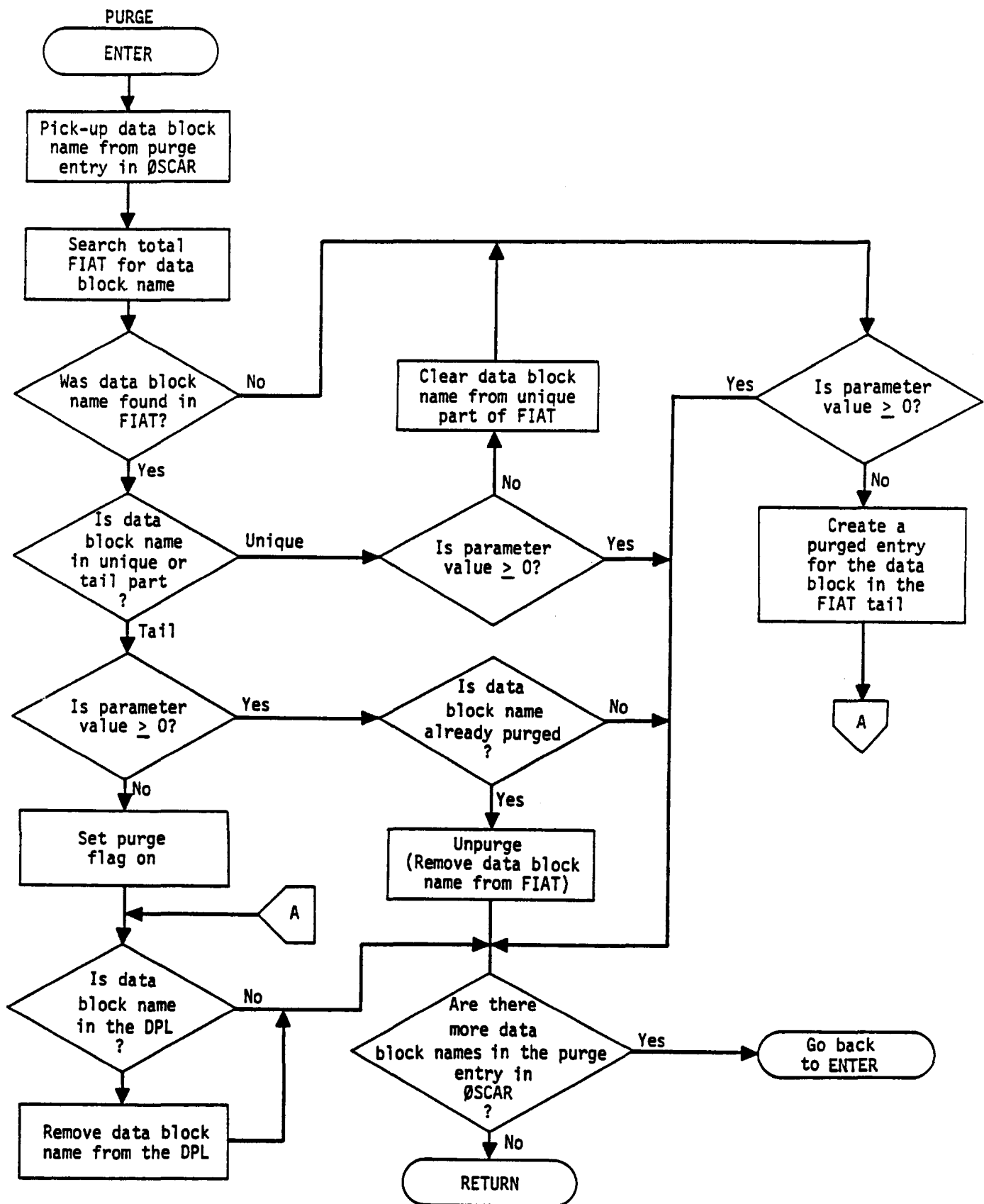


Figure 1. Flowchart for module PURGE.

EXECUTIVE DMAP MODULE EQUIV (DATA BLOCK NAME EQUIVALENCE)

4.17 EXECUTIVE DMAP MODULE EQUIV (DATA BLOCK NAME EQUIVALENCE)

4.17.1 Entry Point: XEQUIV

4.17.2 Purpose

To attach one or more equivalent (alias) data block names to an existing data block so that module accesses to data by equivalenced names will be identical.

4.17.3 DMAP Calling Sequence

EQUIV DBN1A,DBN2A,DBN3A/PARMA/DBN1B,DBN2B/PARMB \$

Note: The number of data block names (DBN α) prior to each parameter (PARM α) and the number of sets of data block names and parameters in a particular calling sequence are variable.

4.17.4 Input Data Blocks

DBN1A,DBN2A, etc. - Any data block names appearing within the DMAP sequence. The 1st data block name in each series (DBN1A and DBN1B) is primary and the 2nd, etc. data block names become equivalent to the primary.

4.17.5 Output Data Blocks

(None specified or permitted)

4.17.6 Parameters

PARMA, etc. - One required for each set of data block names.

4.17.7 Method

4.17.7.1 Summary

The data block names are made equivalent if the value of the associated parameter is < 0 or if the pointer normally reserved to locate the parameter value in the VPS is set to -1. If a set of data blocks is already equivalenced and the parameter value is ≥ 0 , the equivalence is broken and the data block names again become unique. If the data blocks are not equivalenced and if the parameter value is ≥ 0 , no action is taken.

4.17.7.2 Functional Flow

EQUIV operates by modifying entries within the FIAT (File Allocation Table) and DPL (Data Pool Dictionary). The FIAT contains an upper section (unique part) and a lower section (tail part). Both parts contain entries structured as described in the Executive Table description for the FIAT, section 2.4. The length of the unique part is defined by the unique files available count in the FIAT header. The tail part is defined as the remainder of the FIAT. The unique part contains one entry for each unique (separate) file available for allocation and the file ID's within these entries are not modified through a NASTRAN run. The tail part contains entries for stacked files (see description for Executive Table XSFA), purged files and members of equivalenced sets. Entries within the FIAT and DPL are made equivalent by setting their EQUIV flags (sign bit within an entry) and making their file ID's identical. Since a data block within the unique part of the FIAT must have a unique file ID, only one member of an equivalence set may reside within the unique section, all others will be placed in the FIAT tail. Thus, if two data blocks occupying unique physical files are equivalenced, one will be moved to the FIAT tail. Data blocks previously equivalenced are unequivalenced (broken) by removing the EQUIV flags and the secondary entries. When two or more data blocks are equivalenced, the first data block of the set is considered the primary data block. All others are considered secondary. The file containing the primary data block is logically attached to all data blocks in the set: primary and secondary. Data on files attached to secondary data blocks prior to equivalencing is lost upon equivalence. If the primary data block is purged, the secondary(s) will be purged. Figure 1 illustrates the logic flow.

4.17.8 Design Requirements

1. No open core is required by the module.
2. The ØSCAR record containing the DMAP EQUIV request must reside in the labeled common block /ØSCENT/.
3. The validity of all data block names and controlling parameters is checked during NASTRAN initialization by XGPI.
4. XEQUIV is an entry point in XPURGE.

4.17.9 Diagnostic Messages

EQUIV may produce the following System Fatal Messages:

1201 FIAT OVERFLOW

1202 DPL OVERFLOW

Both of these messages indicate that the assembled size of the particular table has been exceeded. Although it is unlikely that either message will occur, a study of the erroneous problem's operation along with diagnostic prints of the FIAT and DPL obtained, via the DIAG Executive Control card (see User's Manual, section 2), should indicate some corrective action. Possible corrective actions include: increasing the basic table size through re-assembly; providing more physical files to the NASTRAN system; and altering the DMAP operations.

MODULE FUNCTIONAL DESCRIPTIONS

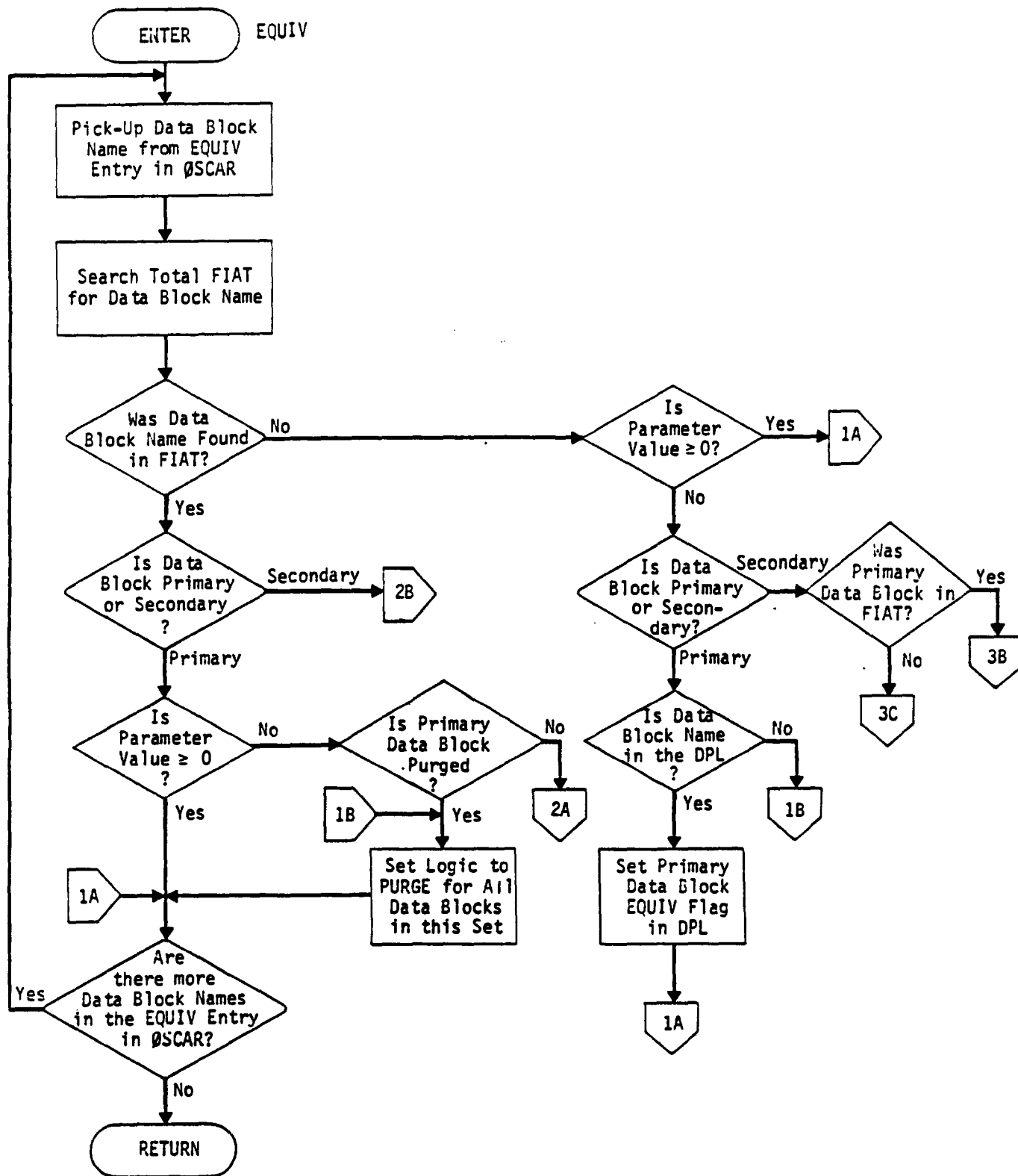


Figure 1. (a) Flowchart for EQUIV module

EXECUTIVE DMAP MODULE EQUIV (DATA BLOCK NAME EQUIVALENCE)

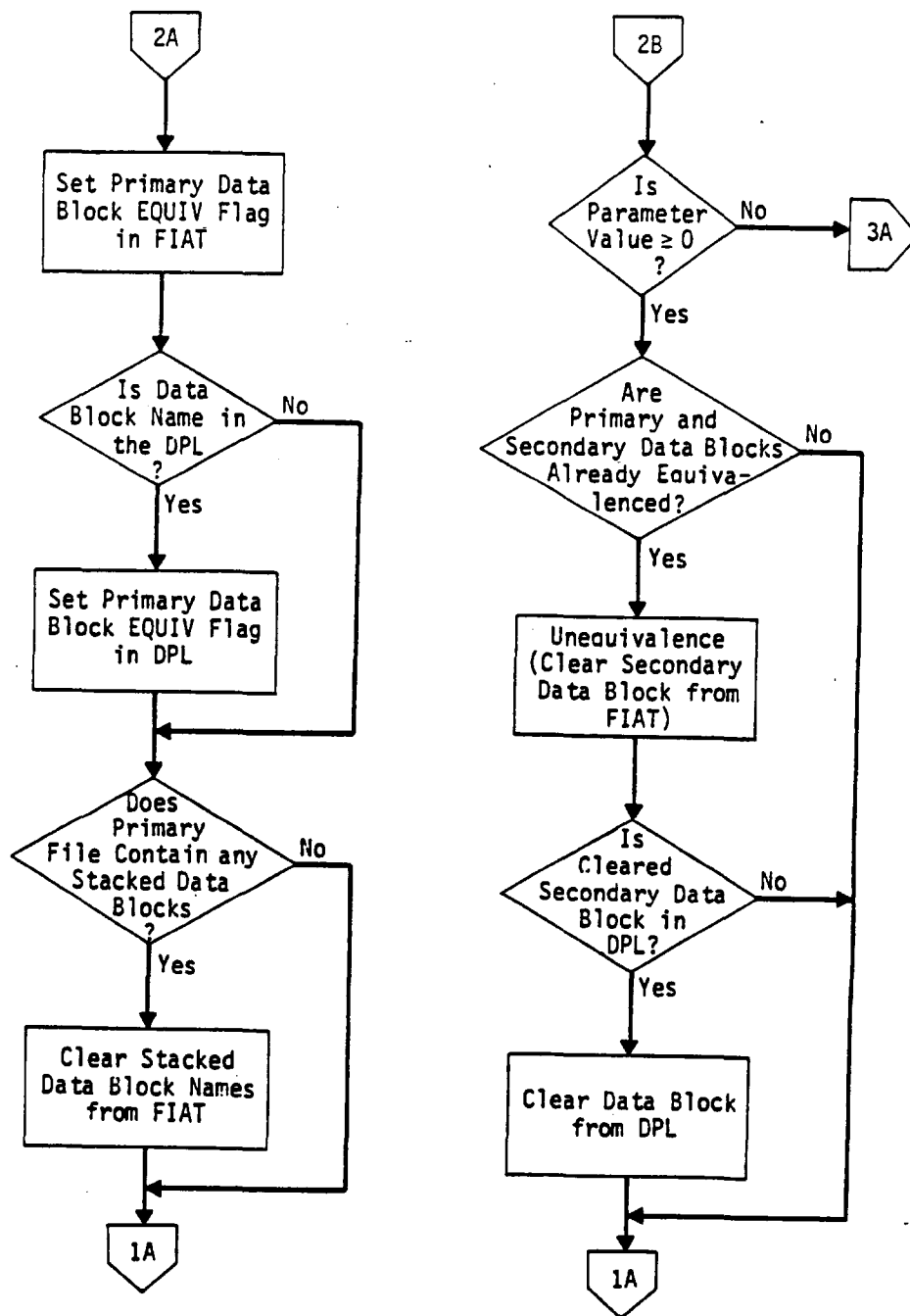


Figure 1. (b) Flowchart for EQUIV module

MODULE FUNCTIONAL DESCRIPTIONS

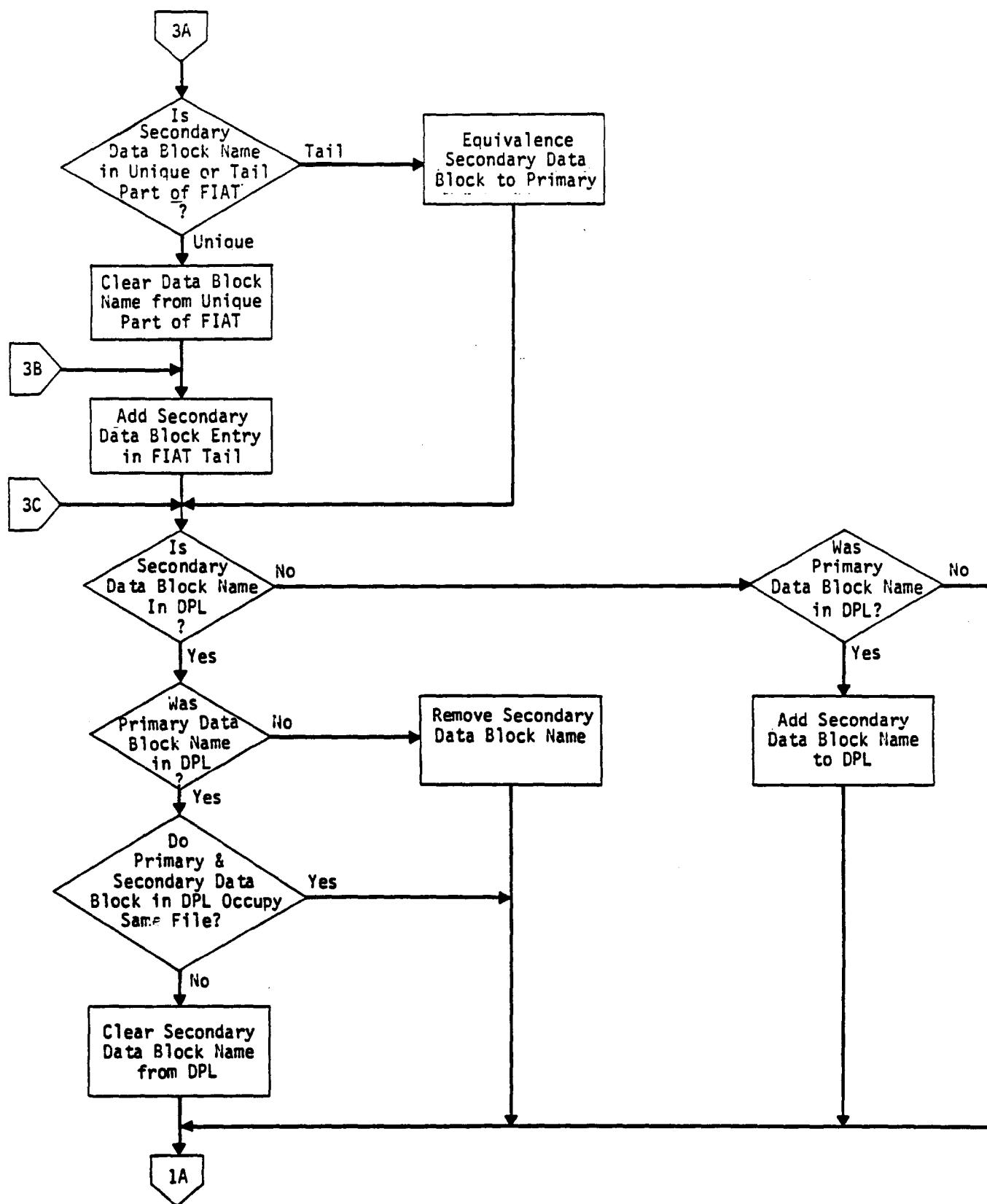


Figure 1. (c) Flowchart for EQUIV module

EXECUTIVE DMAP INSTRUCTION END (END OF DMAP PROGRAM)

4.18 EXECUTIVE DMAP INSTRUCTION END (END OF DMAP PROGRAM)

4.18.1 Entry Point: XCEI

The XCEI module executes the DMAP control instructions: REPT, EXIT, COND, and JUMP.

4.18.2 Purpose

To denote the end of a DMAP program. This DMAP instruction performs a function similar to an END statement in a FORTRAN compilation, i.e., to signal the end of the source program.

4.18.3 DMAP Calling Sequence

END \$

Note: An END DMAP instruction is operationally equivalent to an EXIT 0 \$ or EXIT \$ DMAP instruction.

4.18.4 Method

The END instruction is translated during a DMAP program compilation in module XGPI into an EXIT \$ instruction. (see section 4.14).

EXECUTIVE DMAP MODULE PARAM (PARAMETER PROCESSOR)

4.19 EXECUTIVE DMAP MODULE PARAM (PARAMETER PROCESSOR)

4.19.1 Entry Point: QPARAM

4.19.2 Purpose: To perform specified arithmetic and logical operations on DMAP parameters.

4.19.3 DMAP Calling Sequence:

PARAM //C,N,ØP/V,N,ØUT/V,N,IN1/V,N,IN2 \$

where the following operations (ØP) are available:

ØP	ØUT	IN1	IN2	
AND	-1	< 0	< 0	} Logical
	+1	< 0	≥ 0	
	+1	≥ 0	< 0	
	+1	≥ 0	≥ 0	
ØR	-1	< 0	< 0	
	-1	< 0	≥ 0	
	-1	≥ 0	< 0	
	-1	≥ 0	≥ 0	
IMPL	-1	< 0	< 0	} Integer Arithmetic
	+1	< 0	≥ 0	
	-1	≥ 0	< 0	
	-1	≥ 0	≥ 0	
ADD	IN1+IN2			} Integer Arithmetic
SUB	IN1-IN2			
MPY	IN1*IN2			
DIV	IN1/IN2			
NØT	-IN1		*	} Relational
EQ	-1	= IN2	= IN1	
	+1	≠ IN2	≠ IN1	
GT	-1	> IN2	< IN1	
	+1	≤ IN2	≥ IN1	
LT	-1	< IN2	> IN1	
	+1	≥ IN2	≤ IN1	

MODULE FUNCTIONAL DESCRIPTIONS

ØP	ØUT	IN1	IN2	} Relational continued
LE	-1	\leq IN2	\geq IN1	
	+1	$>$ IN2	$<$ IN1	
GE	-1	\geq IN2	\leq IN1	
	+1	$<$ IN2	$>$ IN1	
NE	-1	\neq IN2	\neq IN1	
	+1	$=$ IN2	$=$ IN1	
NØP	ØUT	*	*	
KLØCK	CALL KLØCK(ØUT)	*	*	
TMTØGØ	CALL TMTØGØ(ØUT)	*	*	
PREC	ØUT=SYSTEM(55)	*	*	
SSST	ØUT > 0 turns on DIAG ØUT ØUT < 0 turns off DIAG ØUT	*	*	
SSSR	ØUT = SYSTEM cell which contains DIAG data SYSTEM cell which contains DIAG data = ØUT	+1	*	
		-1	*	
STSR	ØUT will contain SYSTEM (ISUB) SYSTEM (ISUB) will contain ØUT.	> 0	*	
		< 0	*	
SYSR	SYSTEM(IN1)	> 0	*	
DIAG	DIAG IN1 turned on DIAG IN1 thru DIAG IN2 turned on	\geq IN2 $<$ IN2	\leq IN1 $>$ IN1	
DIAGØFF	Reverse of DIAG	\geq IN2 $<$ IN2	\leq IN1 $>$ IN1	
SYST	SYSTEM(IN1)=IN2	> 0	\geq 0 $<$ 0	

Notes:

1. *not used.
2. PARAM does its own SAVE; therefore, a DMAP SAVE instruction is not needed following the module.
3. PARAM has no input or output data blocks.
4. ØP must be a "C,N" parameter.

EXECUTIVE DMAP MODULE PARAM (PARAMETER PROCESSOR)

4.19.4 Examples

1. PARAM // C,N,NØP / V,N,P1=5 \$ - this example sets the value of parameter P1 to 5 and saves it in the VPS.

2. PARAM // C,N,NØT / V,N,XYZ / V,N,NØXYZ \$ - this example changes the sense of parameter NØXYZ which may be useful for the CØND or EQUIV instructions. Alternatively, XYZ could have been set in the following way:

PARAM // C,N,MPY / V,N,XYZ / V,N,NØXYZ / C,N,-1 \$

4.19.5 Method

QPARAM performs the indicated parameter operation and stores the result in the VPS (/XVPS/). QPARAM processes its own DMAP call from common block /ØSCENT/.

4.19.6 Diagnostic Messages

Fatal error message 2024 may be issued by PARAM.

EXECUTIVE DMAP MODULE SETVAL (SET VALUES)

4.20 EXECUTIVE DMAP MODULE SETVAL (SET VALUES)

4.20.1 Entry Point: SETVAL

4.20.2 Purpose

To set DMAP parameters equal to other DMAP parameters or to constants.

4.20.3 DMAP Calling Sequence

SETVAL //V,N,X1/V,N,Y1/V,N,X2/V,N,Y2/V,N,X3/V,N,Y3/V,N,X4/V,N,Y4/V,N,X5/V,N,Y5 \$

4.20.4 Input Data Blocks

None.

4.20.5 Output Data Blocks

None.

4.20.6 Parameters

X1, X2, X3, X4, X5 - Output-integers-no default values.

Y1, Y2, Y3, Y4, Y5 - Input-integers-default values = -1.

4.20.7 Method

This module does nothing except set X1 = Y1, X2 = Y2, X3 = Y3, X4 = Y4, and X5 = Y5.
Only two parameters need be specified in the calling sequence (X1 and Y1).

4.20.8 Subroutines

SETVAL has no auxiliary subroutines.

4.20.9 Design Requirements

SETVAL should reside in the root segment in all links.

4.20.10 Diagnostic Messages

None.

4.21 FUNCTIONAL MODULE GP1 (GEOMETRY PROCESSOR - PHASE 1)

4.21.1 Entry Point: GP1

4.21.2 Purpose

GP1 performs basic geometry processing for the model. A list of all grid and scalar points is assembled and placed in internal order. Coordinate system transformation matrices are computed, and all grid points are transformed to the basic coordinate system.

4.21.3 DMAP Calling Sequence

GP1 GEØM1,GEØM2,X/GPL,{HEQEXIN}, GPDT,CSTM,BGPD,T,{HSIL},V,N,LUSET/V,N,NØCSTM/V,N,NØGPDT \$

4.21.4 Input Data Blocks

GEØM1 - Grid point, coordinate system, sequence data.

GEØM2 - Element connection data.

X - Dummy data block

4.21.5 Output Data Blocks

GPL - Grid Point List.

{EQEXIN }
{HEQEXIN } - Equivalence between external grid or scalar numbers and internal numbers.

GPDT - Grid Point Definition Table.

CSTM - Coordinate System Transformation Matrices.

BGPD,T - Basic Grid Point Definition Table.

SIL - Scalar Index List.

HSIL - Scalar Index List for heat problems.

Note: No output data block may be purged.

4.21.6 Parameters

LUSET - Output, integer, no default. Total degrees of freedom in the g displacement set.

NØCSTM - Output, integer, no default. Number of coordinate systems defined in the Bulk Data Deck, -1 if no coordinate systems defined.

NØGPDT - Output, integer, no default. -1 if no grid or scalar points defined in Bulk Data Deck, +1 otherwise.

4.21.7 Method

4.21.7.1 Construction of the GPL and First Logical Record of the EQEXIN.

The SPØINT cards and the scalar element cards (CELAS_i, CDAMP_i, CMASS_i, $i = 1,2,3,4$) are read from GEØM2, and a list is made of all referenced scalar points. The GRID cards are read from GEØM1, and a merged list of all grid and scalar points is constructed and written on SCR1, a scratch file. The list is expanded to pairs of numbers. The first number is the identification number, ID, the second is the resequenced number which is given on the SEQGP cards or is 1000*ID if not given on SEQGP cards. The paired list is sorted by SØRT on the sequence numbers. The resulting set of first numbers is written as the first logical record in the GPL (Grid Point List). These are the point identification numbers in order of their sequence numbers. The sequenced paired list is written as the second logical record of the GPL data block. The second numbers in the sequenced paired list are replaced by the indices 1, 2, 3,..., according to position. The list is sorted again, this time using the first number of each pair (the identification number). The resulting paired list is the first logical record of the EQEXIN data block which is used to convert external numbers, given by the first number of a pair, to the internal grid point indices, given by the second number in the pair.

4.21.7.2 Formats of GPDT, BGPDT and CSTM.

The geometry data blocks are the GPDT, the BGPDT and the CSTM. Their formats, although described in section 2.3.3, are repeated here since the following terms will be referenced in the discussion below on the construction of the CSTM.

GPDT - There is one entry for each grid or scalar point. The order of the entries is by the internal (sequenced) order. Each entry contains:

1. Internal sequence number.
2. Locating coordinate system ID.
3. x,y,z for a rectangular system.
4. r,θ,z for a cylindrical system.
5. ρ,θ,ϕ for a spherical system.
6. Global coordinate local coordinate system ID.
7. Permanent single-point constraint coordinate (1 = x , 2 = y , etc.).

For scalar points, word 2 = -1 and words 3 through 7 are zero. The data is essentially a duplicate of the GRID bulk data card except that the identification number is replaced by the internal sequence number.

. BGPDT - Contains one entry for each grid or scalar point. The contents are:

1. Local coordinate system ID for global coordinate definition.
2. x_i } Locations of point
3. y_i } in basic coordinate
4. z_i } system.

CSTM - The CSTM contains one entry for each local coordinate system. The order is by coordinate system identification numbers. Each entry contains 14 words:

<u>Word</u>	<u>Item</u>
1	N - the coordinate system ID.
2	Type _N - the coordinate system type (rectangular, cylindrical or spherical).
3-5	{R _{ON} } - the location of the system origin in basic coordinates.
6-14	[T _{ON}] - the three-by-three matrix defining the orientation of the coordinate system principal axes.

4.21.7.3 Construction of the GPDT.

The GPDT data block is formed in core sized groups. The grid and scalar data are read one entry at a time from SCR1. EQEXIN (in core) is searched to find the internal number, and the grid data are stored (if possible) in the internal position allocated in core. If core will not hold the GPDT, the data are written on SCR2, and SORT is called to sort and write the data on the GPDT.

4.21.7.4 Construction of the CSTM.

Sixteen words are allotted for each local coordinate system, and five words are allotted for each referenced grid point. The CORDij data is read from GEOM1 and stored in core. External point ID's on CORDij cards are replaced with internal numbers. A CORDij card references three grid points. It may be converted to a CSTM entry if these grid points have their locations reduced to basic coordinates. A CORD2j card references another local coordinate system. It may

be converted to a CSTM entry if that referenced system has been reduced to a CSTM entry. The basic logic is to make repeated passes over the coordinate system data, each time reducing one or more coordinate systems and, when possible, converting referenced grid points to basic system location.

A CORD1j card image references three grid points - a, b and c. If the locations of these points in basic coordinates are the vectors $\{R_a\}$, $\{R_b\}$, $\{R_c\}$, the solution for coordinate system N is

$$\{R_{ON}\} = \{R_a\}, \quad (1)$$

$$\{V_k\} = \{R_b\} - \{R_a\}, \quad (2)$$

$$\{V_i\} = \{R_c\} - \{R_a\}, \quad (3)$$

$$\{k\} = \frac{\{V_k\}}{|\{V_i\}|} \text{ (unit "z" vector),} \quad (4)$$

$$\{j\} = \frac{\{k\} \times \{V_i\}}{|\{k\} \times \{V_i\}|} \text{ (unit "y" vector),} \quad (5)$$

$$\{i\} = \{j\} \times \{k\} \text{ (unit "x" vector).} \quad (6)$$

Point a is the origin, point b lies on the z (or polar) axis, point c lies in the x-z plane ($\theta = 0$ or $\phi = 0$). The three-by-three matrix $[T_N]$ is defined as:

$$[T_N] = \begin{bmatrix} i_1 & j_1 & k_1 \\ i_2 & j_2 & k_2 \\ i_3 & j_3 & k_3 \end{bmatrix}. \quad (7)$$

N , type_N , $\{R_{ON}\}$ and $[T_N]$ form the CSTM entry for the coordinate system.

A GRID point (j) referenced to coordinate system (N) may be reduced to basic coordinates (X_0, Y_0, Z_0) by the equations:

1. If $\text{type}_N = \text{Rectangular (R)}$, X_j, Y_j and Z_j are given by

$$\begin{pmatrix} X_0 \\ Y_0 \\ Z_0 \end{pmatrix} = \{R_{ON}\} + [T_N] \begin{pmatrix} X_j \\ Y_j \\ Z_j \end{pmatrix}. \quad (8)$$

2. If $\text{type}_N = \text{Cylindrical (C)}$, r, θ and Z are given by

$$X_j = r \cos \theta, \quad (9)$$

$$Y_j = r \sin \theta, \quad (10)$$

$$Z_j = Z. \quad (11)$$

X_0, Y_0 and Z_0 are calculated as in Equation 8.

3. If $\text{type}_N = \text{Spherical (S)}$, ρ, θ and ϕ are given by.

$$X_j = \rho \sin \theta \cos \phi, \quad (12)$$

$$Y_j = \rho \sin \theta \sin \phi, \quad (13)$$

$$Z_j = \rho \cos \theta. \quad (14)$$

X_0, Y_0 and Z_0 are calculated as above.

When the basic location of a grid point has been calculated, the entry in the list is changed such that the reference coordinate system (entry No. 2) is zero and the three values are X_0, Y_0, Z_0 .

MODULE FUNCTIONAL DESCRIPTIONS

The CORD2j card image references another coordinate system and defines three points in the referenced system: a, b and c. If system number N is defined by system number M, the solution is

1. If $\text{type}_M = \text{rectangular}$, the numbers defining the three points are the vectors: $\{a\}$, $\{b\}$, and $\{c\}$.
2. If $\text{type}_M = \text{cylindrical}$, the numbers are r , θ and z . The equations to convert these to rectangular vectors are

$$\begin{Bmatrix} a_1 \\ a_2 \\ a_3 \end{Bmatrix} = \begin{Bmatrix} r_a \cos \theta_a \\ r_a \sin \theta_a \\ z_a \end{Bmatrix} = \{a\} . \quad (15)$$

The $\{b\}$ and $\{c\}$ vectors are calculated similarly.

3. If $\text{type}_M = \text{spherical}$, the numbers given for the points are ρ , θ , and ϕ . We calculate

$$\begin{Bmatrix} a_1 \\ a_2 \\ a_3 \end{Bmatrix} = \begin{Bmatrix} \rho \sin \theta \cos \phi \\ \rho \sin \theta \sin \phi \\ \rho \cos \theta \end{Bmatrix} = \{a\} \quad (16)$$

and similarly for points b and c.

4. The definition of the new system is that point a is the origin, point b lies on the z (or polar) axis and point c lies in the x-z (or $\theta = 0$) plane. The equations for the CSTM data are

$$\{R_{ON}\} = \{R_{OM}\} + [T_{ON}] \{a\}. \quad (17)$$

In system M the vectors defining the axes of system N are

$$\{V_k\} = \{b\} - \{a\}; \quad (18)$$

$$\{k\} = \frac{\{V_k\}}{|\{V_k\}|}, \text{ ("z" unit vector);} \quad (19)$$

$$\{V_i\} = \{c\} - \{a\}; \quad (20)$$

$$\{j\} = \frac{\{k\} \times \{V_1\}}{|\{k\} \times \{V_1\}|}, \text{ ("Y" unit vector);} \quad (21)$$

$$\{i\} = \{j\} \times \{k\}, \text{ ("X" unit vector).} \quad (22)$$

The orientation of the axes is defined by the matrix

$$[T_{ON}] = [T_{OM}] \begin{bmatrix} i_1 & j_1 & k_1 \\ i_2 & j_2 & k_2 \\ i_3 & j_3 & k_3 \end{bmatrix} \quad (23)$$

5. On each pass of the CØRDij data at least one new system must be converted. After each pass the referenced GRID data is checked and converted. The resulting CØRDij data will be the CSTM data block with each entry reduced from 16 to 14 words.

4.21.7.5 Construction of the BGPDT, the SIL and the Second Logical Record of the EQEXIN.

The BGPDT and the SIL data blocks are formed simultaneously. The SIL data block is simply a list of the first scalar index for each grid or scalar point. The number of scalar indices (or degrees of freedom) for each point is determined by examining the elements connected to each point. The maximum number of degrees of freedom for each element type is listed in /GPTA1/. The maximum degrees of freedom for each point is determined by reading data block GEØM2 and examining the connection information in conjunction with the degree of freedom information in /GPTA1/.

The GPDT data are read a point at a time. The basic location coordinates of the point are formed using Equation 8 through Equation 14 and these data are written on the BGPDT file. The SIL value for the next point is calculated by incrementing the last value by six (grid point) or by one (scalar point).

A test is made on the value of the displacement coordinate system (field 6) in the GPDT data. If this value is the integer, -1, the point is a special RINGFL, GRIDF, or GRIDS fluid point. It is given one scalar index, the displacement coordinate system is basic (0), and its location coordinates in the BGPDT data block are calculated like a normal grid point.

Finally the second logical record of EQEXIN is written. This record contains pairs of external numbers, 10*scalar index + type where type = 1 for a grid point, 2 for a scalar point.

MODULE FUNCTIONAL DESCRIPTIONS

4.21.8 Subroutines

GP1 has no auxiliary subroutines.

4.21.9 Design Requirements

4.21.9.1 Allocation of Core Storage

During the assembly of the GPL, space for 2*(number of grid points plus number of scalar points) plus two GINØ buffers is required. During the assembly of the CSTM, core storage is allocated as follows:

COMMON/GPA1/Z(1)		
1	External point number	Two words per entry one entry per grid or scalar point
	Internal number	
	:	
	:	
ICSDT	Coordinate system ID	Sixteen words per entry one entry per coordinate system
	Coordinate system type { 1 = rectangular 2 = cylindrical 3 = spherical	
	Coordinate system definition { 1 = CØRD1j 2 = CØRD2j	
	Reference coordinate system ID	
	$\begin{Bmatrix} g_1 \\ g_2 \\ g_3 \end{Bmatrix}$ or $\begin{Bmatrix} a_1 \\ a_2 \\ a_3 \end{Bmatrix}$ $\begin{Bmatrix} b_1 \\ b_2 \\ b_3 \end{Bmatrix}$ $\begin{Bmatrix} c_1 \\ c_2 \\ c_3 \end{Bmatrix}$	
ILIST	:	Five words per entry one entry for each grid point referenced on a CØRD1j card.
	Internal grid number	
	Defining coordinate system ID	
	$\begin{Bmatrix} x \\ y \\ z \end{Bmatrix}$	
	:	
BUF1	GINØ buffer	

FUNCTIONAL MODULE GP1 (GEOMETRY PROCESSOR - PHASE 1)

Total storage requirements during this phase, therefore, equals $2 * (\text{number of grid} + \text{number of scalar points}) + 16 * (\text{number of coordinate systems}) + 5 * (\text{number of grid points referenced on CØRD1j cards}) + \text{one GINØ buffer}$.

4.21.9.2 Environment

Open core for GP1 is defined by /GPA1/. The table /GPTA1/ must be in core when GP1 is executed. GP1 uses two scratch files.

4.21.10 Diagnostic Messages

The following diagnostic messages may be issued by GP1:

2001, 2002, 2003, 2004, 2005, 2006, 2012

4.22 FUNCTIONAL MODULE GP2 (GEOMETRY PROCESSOR - PHASE 2)

4.22.1 Entry Point: GP2

4.22.2 Purpose

GP2 processes element connection data and converts external point numbers to internal numbers.

4.22.3 DMAP Calling Sequence

GP2 GEØM2,EQEXIN/ECT \$

4.22.4 Input Data Blocks

GEØM2 - Element connection data.

EQEXIN - Equivalence between external grid or scalar numbers and internal numbers.

Note: EQEXIN may not be purged.

4.22.5 Output Data Blocks

ECT - Element Connection Table.

Note: ECT may not be purged.

4.22.6 Parameters

None

4.22.7 Method

The first data record of EQEXIN (containing pairs of external point identification and internal index) is read into core. GEØM2 is opened, and the header record is skipped. The ECT is opened, and the header record is written. The following process is repeated for each logical record in GEØM2.

1. The 3-word header is read. If an end-of-file is encountered, step (4) is executed. Otherwise, /GPTA1/ (see description in section 2.5) is searched for a match. If found, step (2) is executed. If not found, an internal table, CARDS, which defines additional cards processed

by GP2 (e.g. GENEL) is searched. If a match is found, step (3) is executed. Otherwise, the record is skipped, and step (1) is repeated.

2. The 3-word header from GEØM2 is written on the ECT. Parameters defining the element are fetched from /GPTA1/. If the number of words per element is less than 5, the sort flag in the GEØM2 trainer is fetched. Each element card of the current type on GEØM2 is read. Each external grid identification is converted to an internal index by performing a binary search in the EQEXIN table. If the point is not in the table, an error message is queued and the NØGØ flag is turned on. If the data is not to be sorted, the element is written directly on the ECT. Otherwise it is saved in core (or written on a scratch file if core is full). When all elements of a given type have been processed, the sort flag is again tested. If off, the ECT record is closed and return to step (1) is made. Otherwise, the data are sorted by SØRT and the ECT record is then written.

3. For GENEL, SEQBFE and QDSEP data (the latter two are Force Method only), each entry is read, all external point identifications are converted to internal indices as in (2) and the entry is written on the ECT. When the logical record on GEØM2 is exhausted, the ECT record is closed and return to step (1) is made.

4. The ECT trailer is written, and all files are closed. If the NØGØ flag was turned on, PEXIT is called. Otherwise, a normal exit is made.

4.22.8 Subroutines

The module GP2 consists of one subroutine, GP2.

4.22.9 Design Requirements

4.22.9.1 Allocation of Core Storage

GP2 requires space for $2 * (\text{number of grid points} + \text{number of scalar points}) + \text{three GINØ buffers}$.

4.22.9.2 Environment

Open core is defined by /GPA2/. The table /GPTA1/ must be in core when GP2 is executed. GP2 uses up to four scratch files.

4.22.10 Diagnostic Messages

The following diagnostic messages may be issued by GP2:

2007, 2059, 2060, 2061, 2138.

4.23 FUNCTIONAL MODULE PLTSET (PLOT SET DEFINITION PROCESSOR)

4.23.1 Entry Point: DPLTST

4.23.2 Purpose

To generate the structural element sets to be used by the structural plotter (functional module PLØT).

4.23.3 DMAP Calling Sequence

PLTSET PCDB,EQEXIN,ECT/PLTSETX,PLTPAR,GPSETS,ELSETS/V,N,NGP/V,N,NPSET \$

4.23.4 Input Data Blocks

PCDB - Plot Control Data Block for the structure plotter.

EQEXIN - Equivalence between external grid or scalar numbers and internal numbers.

ECT - Element Connection Table.

Note: If PCDB is purged, nothing is done in this module. However, if PCDB is not purged, neither EQEXIN nor ECT may be purged.

4.23.5 Output Data Blocks

PLTSETX - User error messages related to the definition of element plot sets for the structure plotter.

PLTPAR - Plot parameters and plot control table.

GPSETS - Grid point sets related to the element plot sets.

ELSETS - Element plot set connection tables.

Note: None of these data blocks may be pre-purged unless PCDB is also purged.

4.23.6 Parameters

NGP - Output-integer-no default. Total number of grid and scalar points

NPSET - Output-integer-default value = -1. Number of element plot sets (set to -1 if none).

MODULE FUNCTIONAL DESCRIPTIONS

4.23.7 Method

Subroutine SETINP reads each logical card in the plot control data block (PCDB). If the first entry on a card is not "SET", the card is assumed to be a plot parameter or control card meaningful only to the PLOT module. In this case, the logical card is copied onto the PLTPAR data block.

If the first entry on a card is "SET", it is assumed to be a definition of a new element plot set. As each entry on the card is read, it is decided whether a list of elements (by type, range or explicit id's) or a list of grid points (by range or explicit id) is being included or excluded. An element specified by type (e.g., "R0D") is located in the /GPTA1/ data block and the index (I^{th} type) and current pointer into the EL array are placed in a table (TYP, 100 words long). The EL array (first part of open core) is created when element numbers are specified. Note that "THRU" is more efficient than listing each element number. Finally, if grid point numbers are specified, they are inserted into the end of open core (the GP array). When the set has been completely defined, duplicate element types are deleted and it is written out onto a scratch file (MSET) in the following format:

Number of Words

1	Setid
1	NEL (number of entries in the EL array)
NEL	the entries in the EL array
1	NTYP (number of entries in the TYP array)
NTYP	the entries in the TYP array
1	NPT (number of entries in the GP array)
NPT	the entries in the GP array

After the PCDB has been processed, subroutine C0MECT is called to set up a shortened element connection table (ECTX). For each element type, a record is created as follows:

for each element of this type	Word 1	=	internal NASTRAN element type number
	Word 2	=	number of grid points per element
	Word 3	=	element id
	Word 4	=	index in ECT of these element types
	Word 5 etc.	=	internal grid point numbers of the grid points connected by this element

FUNCTIONAL MODULE PLTSET (PLOT SET DEFINITION PROCESSOR)

This table, in conjunction with MSET, is used by subroutine CNSTRC to create the GPSETS and ELSETS data blocks. The ELSETS data block is simply a duplicate of ECTX for each plot set, except that only those elements which are in the set are included and the element type number is converted to the two character BCD element type symbol (left justified). The GPSETS data block for each plot set is simply a list of sequential indicies (negative means cannot be labeled) into the subset of grid points which pertain to this set.

4.23.8 Subroutines

Utility routines CLSTAB, FREAD, GOPEN, INTGPX, INTGPT, INTLST, RDMØDX, RDMØDE and RDWØRD are used by PLTSET. See Section 3.4 for their descriptions.

4.23.8.1 Subroutine Name: SETINP

1. Entry Point: SETINP
2. Purpose: To create the plot parameter and control data block (PLTPAR) and interpret the plot set definition cards from the Plot Control Data Block (PCDB).

3. Calling Sequence: CALL SETINP

COMMON / XXPSET / X(1)

COMMON // SKP11,NSETS,SKP12(8),PCDB,SKP2(9),MERR,PLOT,MSETID,SKP3(7),MSET

NSETS - Number of plot sets

PCDB - Plot Control Data Block

MERR - Error message file

PLØT - PLTPAR data file

MSETID - GPSETS data file

MSET - SCRATCH1 output data file

X - Start of open core

4.23.8.2 Subroutine Name: CØMECT

1. Entry Point: CØMECT
2. Purpose: To create a shortened form of the Element Connection Table (ECT).
3. Calling Sequence: CALL CØMECT (ELE, MAX)

COMMON / XXPSET / X(1)

MODULE FUNCTIONAL DESCRIPTIONS

COMMON // SKP(12),ECT1,SKP22(7),MERR,SKP1(10),ECT2

ECT1 - ECT data file
MERR - Error message file
ECT2 - SCRATCH2 output data file
X - start of open core

where

ELE - maximum number of locations used to set up the element set data block (ELESETS)
MAX - amount of core available for the ELE array (open core)

4.23.8.3 Subroutine Name: CNSTRC

1. Entry Point: CNSTRC
2. Purpose: To construct the element and grid point plot set data blocks (ELSETS,GPSETS).

3. Calling Sequence: CALL CNSTRC (GP,ELE,BUF,MAX)

COMMON // NGP,NSETS,SKP(9),EXGPID,SKP22(8),MERR,SKP31,GRID,ECT2,SKP32(6),MSET,ECT1

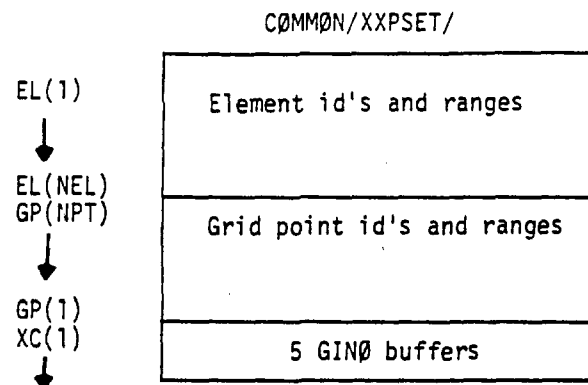
NGP - number of grid points in the structure
NSETS - number of plot sets
EXGPID - EQEXIN data file
MERR - Error message file
GRID - GPSETS data file
ECT2 - ELSETS data file
MSET - SCRATCH1 set input data file
ECT1 - SCRATCH2 shortened ECT input data file
GP - NGP locations used to set up the grid point index list for the grid point set data block (GPSETS).
ELE - MAX locations used to set up the element set data block (ELSETS).
BUF - Location of 3 GINØ buffers.
MAX - Amount of core available for the ELE array (open core)

4.23.9 Design Requirements

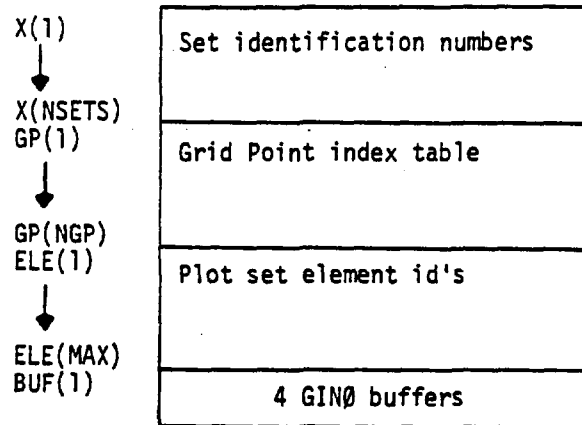
Open core Design (Common Block XXPSET)

FUNCTIONAL MODULE PLTSET (PLOT SET DEFINITION PROCESSOR)

1. Subroutine SETINP



2. Subroutine CNSTRC and CØMECT



4.23.10 Diagnostic Messages

A fatal message occurs in SETINP if a set specification is so large that open core is filled (i.e., array EL meets array GP). All other diagnostics are non-fatal and are written on the PLTSETX data block for printing by the PRTMSG module.

FUNCTIONAL MODULE PLOT (STRUCTURAL PLOTTER)

4.24 FUNCTIONAL MODULE PLOT (STRUCTURAL PLOTTER)

4.24.1 Entry Point: DPLØT

4.24.2 Purpose

To draw structural shapes on a variety of different plotters.

4.24.3 DMAP Calling Sequence

```
PLØT    PLTPAR,GPSETS,ELSETS,CASECC,BGPDØT,EQEXIN,SIL,PLTDSP1,{PLTDSP2}{HGPECT}{HØES1}
        {ECT}{GPECT}{ECPT}{ØES1}/
        PLØTX/V,N,NGP/V,N,LSIL/V,N,NPSET/V,N,PLTFLG/V,N,PLTNUM $
```

4.24.4 Input Data Blocks

PLTPAR - Plot parameters and plot control table.

GPSETS - Grid point sets related to the element plot sets.

ELSETS - Element plot set connection tables.

CASECC - Case Control Data Table.

BGPDØT - Basic Grid Point Definition Table.

EQEXIN - Equivalence between external grid or scalar numbers and internal numbers.

SIL - Scalar Index List.

PLTDSP1 - Translational deformation (statics).

PLTDSP2 - Translational deformations (dynamics).

ECT - Element Connection Table (required if plotting property ids).

$\left. \begin{array}{l} \text{HGPECT} \\ \text{GPECT} \\ \text{ECPT} \end{array} \right\}$ - Grid point element connection table. (required for stress plotting - not implemented)

$\left. \begin{array}{l} \text{HØES1} \\ \text{ØES1} \end{array} \right\}$ - Output element stress requests. (required for stress plotting - not implemented)

Notes:

1. Only SIL, PLTDSP1, and PLTDSP2 may be purged. If this is the case, only undeformed shapes may be drawn.
2. If either PLTDSP1 or PLTDSP2 is purged, that type of deformed shape will not be drawn.
3. If either PLTDSP1 or PLTDSP2 is not purged, SIL may not be purged.

MODULE FUNCTIONAL DESCRIPTIONS

4.24.5 Output Data Blocks

PLØTX - User messages.

Note: PLØTX may not be purged.

4.24.6 Parameters

NGP - Integer-input-no default value. Number of grid points.

LSIL - Integer-input-no default value. Last scalar index value.

NPSET - Integer-input-no default value. Number of element plot sets.

PLTFLG - Integer-input/output-default value = 1. Displacement plot flag.

= 1 if undeformed shapes have not yet been drawn.

= -1 if undeformed shapes have been drawn.

PLTNUM - Integer-input/output-default value = 0. Plot number.

4.24.7 Method

The set definitions are checked in DPLØT for no elements in the set. If no elements exist, the default set will be used (the set ID is set negative). The default set is the first well defined set. If no default set exists, the module exits. Otherwise, subroutine PARAM is then called.

Subroutine PARAM reads each card in the plot parameter control table (PLTPAR). If the first entry on a card is not 'FIND' or 'PLØT', it is assumed to be a plot parameter card to be processed within PARAM (e.g., PROJØCTION, PLØTTER, etc.). Within PARAM, an implied 'FIND' card is initially set up to automatically find an origin, vantage point, and scale. In addition this same implied "FIND" card is set up each time a new projection is defined or a 'PLØTTER' card is encountered. At the same time, the view angles are re-initialized to their default values, the regions pertaining to each origin are reset to full pictures, and all previously defined origins are nullified.

When a 'FIND' card is encountered, subroutine FIND is called both to interpret the card and act upon its requests. And finally, when a 'PLØT' card is encountered, subroutine PLØT is called both to interpret the card and to act upon its requests. However, in this case, if the implied 'FIND' card set up by subroutine PARAM still exists (i.e., if no origin, scale, or vantage point has been defined) the FIND subroutine is called to satisfy these needs before subroutine PLØT is called.

FUNCTIONAL MODULE PLØT (STRUCTURAL PLOTTER)

In subroutine FIND, after the interpretation of the 'FIND' card is completed, a coordinate system rotation matrix is calculated relative to the current view angles, and then the vantage point, scale factors, and the origin requested are calculated as needed.

In subroutine PLØT, after the interpretation of the 'PLØT' card is completed, a list of messages to the plotter operator is generated. Then all plots requested on the plot card are generated by calling subroutine DRAW for each plot/set request.

Subroutine DRAW plots each set requested on the current frame. It sets up the region of the plot. For each request DRAW controls the rotation and scaling of the grid points and deformations based on the current viewing angle and translates these coordinates to the origin specified for this plot. It also controls the various options of a plot as specified on the 'PLØT' card, e.g., drawing a shape, labeling grid points, etc.

In addition, DRAW controls the creation of a grid point connection table (for the set being plotted) so that unique lines for 'SHAPE' options may be created by way of subroutine SHAPE. If sufficient core is not available, each element gets drawn by SHAPE.

4.24.8 Subroutines

The following utility routines are called by PLØT: CLSTAB, FREAD, GØPEN (INTGPX, INTGPT), INTLST, (RDMØDX, RDMØDY, RDMØDE, RDWØRD). See the subroutine descriptions in Section 3. The sub-routines FNDSET, MINMAX, PERPEC, PRØCES and WRTPRT are support subroutines used by more than one of the following subroutines.

4.24.8.1 Subroutine Name: PARAM

1. Entry Point: PARAM
2. Purpose: To interpret the plot parameter cards and to detect the 'FIND' and 'PLØT' plot control cards. In addition, it serves as a driver for subroutine FIND and PLØT.
3. Calling Sequence: CALL PARAM (SETID, X, BUF4)
 CØMMØN/XXPARM/ - See XXPARM table description below (Section 4.24.9.2)
 CØMMØN/PLTDAT/ - See PLTDAT miscellaneous table description (Section 2.5)
 CØMMØN // - See description in Section 4.24.9.2.

where

SETID = Various plot set id's created in the PLTSET module (Record 1 of GPSETS).

X = Open core.

BUF4 = Buffer location

4. Method: All plot parameters are inserted in the XXPARM table. Any parameter which is not recognizable causes a message to be created to this effect, and the parameter is then ignored.
5. Additional Subroutines Requires: FIND, PLØT

4.24.8.2 Subroutine Name: FIND

1. Entry Point: FIND
2. Purpose: To interpret a 'FIND' card and to calculate the parameters requested on the card.
3. Calling Sequence: CALL FIND (MØDE, BUF1, BUF4, SETID, X)
 CØMMØN/XXPARM/ - See XXPARM table description below (Section 4.24.9.2).

FUNCTIONAL MODULE PLØT (STRUCTURAL PLOTTER)

CØMMØN/PLTDAT/ - See PLTDAT miscellaneous table description (Section 2.5).

CØMMØN // - See description in Section 4.24.9.2.

where:

MØDE = Current value of the XRCARD mode value as read and modified by subroutine RDMØDX.

BUF1 = Buffer location.

BUF4 = Buffer location.

SETID = Various plot set id's created in the PLTSET module (record 1 of GPSETS).

X = Open core.

4. Method: After interpreting the 'FIND' card, the coordinate system rotation matrix is calculated (based upon the current view angles), the vantage point, scale factor, and desired origin are calculated.

5. Additional Subroutines Required: FNDSET, MINMAX, PRØCES, PERPEC.

4.24.8.3 Subroutine Name: PLØT

1. Entry Point: PLØT

2. Purpose: To interpret the 'PLØT' card, and produce all the plots requested on the card by acting as a driver to subroutine DRAW.

3. Calling Sequence: CALL PLØT (MØDE, BUF1, B1, SETID, X)

CØMMØN/XXPARM/ - See XXPARM table description below (Section 4.24.9.2).

CØMMØN/PLTDAT/ - See PLTDAT miscellaneous table descriptions (Section 2.5).

CØMMØN // - See description in Section 4.24.9.2.

where:

MØDE = Current value of the XRCARD mode value as read and modified by subroutine RDMØDX.

BUF1 = Buffer location.

B1 = Buffer location.

SETID = Various plot set id's created in the PLTSET module (record 1 of GPSETS).

X = Open core.

4. Method: After interpreting the deformed structure plot requests (there may be many on one 'PLØT' card), the rest of the 'PLØT' card is read into memory. For each deformed

MODULE FUNCTIONAL DESCRIPTIONS

structure request, the appropriate deformation data block (PLTDSP1 or PLTDSP2) is searched for a matching subcase id. If one is found, (this search does not occur if only the undeformed requests are being serviced), then the plot is initiated and rest of the plot card is interpreted for the various plotting options until the next 'SET' is encountered. Subroutine DRAW is then called to service these options and to draw the corresponding picture for each plot element set listed on the 'PLOT' card. If additional 'SET's exist on the 'PLOT' card the process is repeated.

5. Additional subroutines Required: HEAD, FNDSET, GETDEF, DRAW, STPLOT.

4.24.8.4 Subroutine Name: GETDEF

1. Entry Point: GETDEF

2. Purpose: To read the translational components of a set of deformations (in the basic coordinate system).

3. Calling Sequence: CALL GETDEF (DFRM,PH,MAG,CØNV,PLTTYP,BUF,GPT,D)

CØMMØN//XXPARM/ - See XXPARM table descriptions below (Section 4.24.9.2).

CØMMØN// - See description in Section 4.24.9.2.

where:

DFRM = Deformation data block to be read (pre-positioned at the set of deformations to be read).

PH = Phase angle for complex numbers.

MAG = 0 - complex numbers will use $\sqrt{(U_r)^2 + (U_i)^2}$ for each component.
1 - complex numbers will use

$$U = \text{CONV}[U_r \cos (PH + \phi) - U_i \sin (PH + \phi)]$$

for each component where $\phi = 0.0$ (displacement), $\pi/2$ (velocity) or π (acceleration).

PLTTYP = $\left. \begin{array}{l} 1 - \text{Displacement} \\ 2 - \text{Velocity} \\ 3 - \text{Acceleration} \end{array} \right\}$ used only for complex numbers.

BUF = Buffer location.

CØNV = Conversion factor for velocity ($2\pi f$), acceleration $(2\pi f)^2$ or displacement (1.0) for complex numbers.

GPT = List of grid point indices defining a subset of grid points.

D = Array into which the components are to be read (3 per grid point - X,Y,Z).

FUNCTIONAL MODULE PLØT (STRUCTURAL PLOTTER)

4. Method: The scalar index list (SIL data block) is used to determine at which grid point a particular deformation component is specified. While reading the components, a maximum absolute component (MAXDEF) is determined. For complex numbers, MAXDEF is set after conversion.

4.24.8.5 Subroutine Name: PLTØPR

1. Entry Point: PLTØPR

2. Purpose: To generate printed output to be used by the plotter operator in setting up the plotting equipment, and to generate output informing the user of the plotting parameters used to generate the plots.

3. Calling Sequence: CALL PLTØPR

COMMON/PLTDAT/ - See PLTDAT miscellaneous table description (Section 2.5).

COMMON/XXPARM/ - See XXPARM table description below (Section 4.24.9.2).

4. Method: All output is written on the PLØTX data block for subsequent processing by the PRTMSG module. The resulting output can be used by the user to alter certain plot parameters on a subsequent run, if he desires, in order to slightly alter the plots produced.

MODULE FUNCTIONAL DESCRIPTIONS

THIS PAGE HAS BEEN LEFT BLANK INTENTIONALLY.

4.24.8.6 Subroutine Name: DRAW

1. Entry Point: DRAW
2. Purpose: To service the many possible plotting options and generate the corresponding picture.
3. Calling Sequence: CALL DRAW (GPLST,X,U,S,DEFØRM,STEREØ,ØPCØR,BUF1)
 CØMMØN/PLTDAT/ - See PLTDAT miscellaneous table description (Section 2.5).
 CØMMØN/XXPARM/ - See XXPARM table description below (Section 4.24.9.2).
 CØMMØN/RSTXXX/ - See RSTXXX table description below (Section 4.24.9.2).
 CØMMØN/DRWDAT/ - See DRWDAT table description below (Section 4.24.9.2).
 CØMMØN // - See description in Section 4.24.9.2.

where:

- GPLST = List of indices (one for each structural grid point) into the subset of grid points which pertain to the element set appropriate to this plot.
- X = Coordinates of the grid points in this element set (3 per grid point - r,s,t).
- U = Deformation components for each grid point in this element set (3 per grid point - X,Y,Z).
- S = Location into which the s and t deformed structure grid point coordinates are to be placed (same as U for undeformed requests). Also used by subroutine LINEL.
- DEFØRM = $\begin{cases} 0 & \text{if an undeformed structure plot is requested} \\ 1 & \text{if a deformed structure plot is requested} \end{cases}$
- STEREØ = $\begin{cases} 0 & \text{if the left image of a stereo plot is to be generated.} \\ 1 & \text{if the right image of a stereo plot is to be generated.} \end{cases}$
- ØPCØR = Length of open core. (Integer)
- BUF1 = Buffer location. (Integer)

4. Method: Initially, the grid points are rotated based upon the current viewing angles, translated to the selected plot origin, and converted to plotter units using the current scale factor, and the deformation components are reduced to the specified maximum deformation value. If an undeformed or deformed 'SHAPE' will be required, subroutine LINEL is called to create a grid point connection array that will be used by subroutine SUPLT by way of SHAPE. Then the undeformed shape plot is generated if required.
- Next, the deformed structure shape (if requested) is drawn. Then the deformation vectors (as requested) are drawn.

MODULE FUNCTIONAL DESCRIPTIONS

5. Additional Subroutines Required: MINMAX, PROCES, PERPEC, INTVEC, SHAPE, GPTSYM, GPTLBL, DVECTR, ELELBL, LINEL.

4.24.8.7 Subroutine Name: INTVEC

1. Entry Point: INTVEC
2. Purpose: To interpret the user supplied deformation vector plot request.
3. Calling Sequence: CALL INTVEC (VECTØR)

where:

VECTØR = BCD characters specified in the deformation vector request (any combination up to four letters of the characters R, X, Y, Z, N). Input and Output. On input, VECTØR is integer (=0) or BCD. On output, VECTØR is integer (=0, if = 0 upon input).

4. Method: The result is stored into VECTØR, as follows:

X = 2^0

Y = 2^1

Z = 2^2

R = 2^3 (if VECTØR = 'R' only, it is treated as if VECTØR = RXYZ).

N = -VECTØR (the negative of the sum of the other characters)

4.24.8.8 Subroutine Name: SHAPE

1. Entry Point: SHAPE
2. Purpose: To draw a structural shape.
3. Calling Sequence: CALL SHAPE (GPLST,X,U,PEN,DEFØRM,IØPT,IPTL,S,ØPCØR)
COMMON // - See description in Section 4.24.9.2.

where:

GPLST = List of indices into the subset of grid points pertaining to the shape to be drawn.

X = Corresponding grid point coordinates of the undeformed structure (3 per grid point - r, s, t).

U = Corresponding grid point coordinates of the deformed structure (2 per grid point - s, t).

S = Open core.

ØPCØR = Length of open core.

PEN = Pen number or line density to be used to draw the shape.

FUNCTIONAL MODULE PLØT (STRUCTURAL PLOTTER)

DEFØRM = { 0 if the undeformed shape is to be drawn.
 { 1 if the deformed shape is to be drawn.

IØPT = { -1 if grid point connection table (created by LINEL) does not exist.
 { number of words minus one to the start of grid pointers used by SUPLT.

IPTL = { 0 if grid point connection table does not exist.
 { Number of words minus one to start of grid pointers used by SUPLT.

4. Method: The structural shape to be drawn is defined either on the ELSETS data block (assumed open and positioned at the current element set by PLØT) or by a grid point connection array created by LINEL. If the ELSETS version is to be used, as each element type is read it is determined if it is an element with a special plotting algorithm. If so, subroutine LINEL is called to read as many elements as possible for each call. If not, each element is read and drawn taking into account whether the element is one or two dimensional. If the grid point connection array (contained in core) is to be used, subroutine SUPLT is called to create the 'SHAPE'.

5. Additional Subroutines Required: LINEL, SUPLT.

4.24.8.9 Subroutine Name: GPTSYM

1. Entry Point: GPTSYM

2. Purpose: To type a special symbol at each of a subset of grid points.

3. Calling Sequence: CALL GPTSYM (GPLST,X,U,SYM,DEFØRM)

COMMON // - see description in Section 4.24.9.2.

where:

GPLST = List of indices defining the subset of grid points.

X = Corresponding grid point coordinates of the undeformed structure (3 per grid point - r, s, t).

U = Corresponding grid point coordinates of the deformed structure (2 per grid point - s, t).

SYM = Two indices to be used to construct the special symbol.

DEFØRM = 0 if the undeformed grid points are to be used.
 1 if the deformed grid points are to be used.

MODULE FUNCTIONAL DESCRIPTIONS

4.24.8.10 Subroutine Name: GPTLBL

1. Entry Point: GPTLBL
2. Purpose: To type the external grid point id of each of a subset of grid points.
3. Calling Sequence: CALL GPTLBL (GPLST,X,U,DEFØRM,BUF)
COMMON/PLTDAT/ - see the PLTDAT miscellaneous table description (Section 2.5).
COMMON// - see description in Section 4.24.9.2.

where:

- GPLST = List of indices defining the subset of grid points.
- X = Corresponding grid point coordinates of the undeformed structure (3 per grid point - r, s, t).
- U = Corresponding grid point coordinates of the deformed structure (2 per grid point - s, t).
- DEFØRM = $\begin{cases} 0 & \text{if the undeformed grid points are to be used.} \\ 1 & \text{if the deformed grid points are to be used.} \end{cases}$
- BUF = Buffer location.

4. Method: The internal and external id of each structural grid point is read from the EQEXIN data block. If the grid point is part of the specified subset, then the external id is printed to the immediate right of the grid point.

4.24.8.11 Subroutine Name: DVECTR

1. Entry Point: DVECTR
2. Purpose: To draw deformation vectors.
3. Calling Sequence: CALL DVECTR (GPT,X,U,PEN)
COMMON// - See description in Section 4.24.9.2.

where:

- GPT = List of indices defining the subset of grid points at which vectors are to be drawn.
- X = Corresponding grid point coordinates of the undeformed structure (3 per grid point - r, s, t).
- U = Corresponding grid point coordinates of the deformed structure (2 per grid point - s, t).
- PEN = Pen number or line density to be used to draw the vectors.

4. Method: Calls are made to LINE.

4.24.8.12 Subroutine Name: FNDSET

1. Entry Point: FNDSET
2. Purpose: To read the grid point coordinates from the BGPDT data block for the pertinent set of elements.
3. Calling Sequence: CALL FNDSET (GPID,X,IBUF)

where:

IBUF = Buffer location.

GPID = Array with the list of indices defining the subset of grid points in the set.

X = Array into which the corresponding coordinates are to be read (3 per grid point - x, y, z).

4. Method: The nonzero indices of GPID cause the corresponding coordinates from BGPDT to be read into X. The calling program is assumed to have checked for core requirements (3*NGPSET).

4.24.8.13 Subroutine Name: MINMAX

1. Entry Point: MINMAX
2. Purpose: To initialize the minimum and maximum grid point coordinates to a very large and smaller number, respectively.
3. Calling Sequence: CALL MINMAX
COMMON/RSTXXX/ - See the RSTXXX table description below (Section 4.24.9.2).

4.24.8.14 Subroutine Name: PERPEC

1. Entry Point: PERPEC
2. Purpose: To calculate the vantage point and/or translate the grid point coordinates to the vantage point.
3. Calling Sequence: CALL PERPEC (X,STEREØ)
COMMON/XXPARM/ - See XXPARM table description (Section 4.24.9.2).
COMMON/RSTXXX/ - See the RSTXXX table description (Section 4.24.9.2).
COMMON // - See description in Section 4.24.9.2.

MODULE FUNCTIONAL DESCRIPTIONS

where:

X = Set of grid point coordinates to be translated (3 per grid point - r, s, t).
STEREO = $\begin{cases} 0 & \text{if the coordinates of the left image for stereo are to be calculated.} \\ 1 & \text{if the coordinates of the right image for stereo are to be calculated.} \end{cases}$

4. Method: After the vantage point is calculated (if required), each grid point is translated. In the process, unless the projection is stereo, the minimum and maximum s and t coordinates are calculated. Finally, the differences between these minima and maxima, and their averages, are calculated.

4.24.8.15 Subroutine Name: PRØCES

1. Entry Point: PRØCES

2. Purpose: To exchange coordinate axes (as requested) and rotate the grid point coordinates based upon the current view angles.

3. Calling Sequence: CALL PRØCES (X)

CØMMØN/XXPARM/ - See the XXPARM table description below (Section 4.24.9.2).

CØMMØN/RSTXXX/ - See the RSTXXX table description below (Section 4.24.9.2).

CØMMØN // - See the description in Section 4.24.9.2.

where:

X = Grid point coordinates (3 per grid point - x, y, z input; s, t, r output).

4. Method: In addition to its primary purpose, this subroutine calculates the minimum and maximum rotated grid point coordinates, and the differences and averages of these minima and maxima.

4.24.8.16 Subroutine Name: ELELBL

1. Entry Point: ELELBL

2. Purpose: To type the element identification number and the element property identification number.

3. Calling Sequence: CALL ELELBL (GPLST,X,U,DEFØRM)

CØMMØN/CHAR94/ - See the CHAR94 miscellaneous table description (Section 2.5).

CØMMØN/PLTDAT/ - See the PLTDAT miscellaneous table description (Section 2.5).

CØMMØN // - See the description in Section 4.24.9.2.

FUNCTIONAL MODULE PLOT (STRUCTURAL PLOTTER)

where:

- GPLST = List of indices defining the set of grid points associated with the elements to be labeled.
- X = Corresponding grid point coordinates of the undeformed structure.
- U = Corresponding grid point coordinates of the deformed structure.
- DEFØRM = 0 if the undeformed grid points are to be used.
1 if the deformed grid points are to be used.

4. Method: The request is examined to determine if properties are to be labeled (PLABEL=4). This can be accomplished if the ECT file is altered into the DMAP calling sequence in the PLTDSP2 position when doing undeformed plots (the file is created in TAl thus the plot module DMAP instructions must be moved to after TAl). If it cannot be accomplished the request defaults to LABEL ELEMENTS.

The compact element connection table (ELSETS) is read one element at a time. The element id and type is plotted at the center of the element. If the element is a quadrilateral or triangle this label is oriented parallel to the longest side with the minimum slope. The property label is placed under or to the right of element label using the same slope.

4.24.8.17 Subroutine Name: W RTPRT

1. Entry Point: W RTPRT
2. Purpose: To write messages on a data block for subsequent processing by WRTMSG.
3. Calling Sequence: CALL W RTPRT (G,L,F,N)
 - G = GINØ file name
 - L = List vector of length L(1) in cells L(2)-L(L(1)+1)
 - F = Format vector
 - N = Length of format vector

4.24.8.18 Subroutine Name: LINEL

1. Entry Point: LINEL
2. Purpose: To create a line connection list for subroutine SHAPE, and to create a grid point connection array that is used by SUPLT.

MODULE FUNCTIONAL DESCRIPTIONS

3. Calling Sequence: CALL LINEL(IZ,NWDS,ØPCØR,ØPT)

CØMMØN// - See description in Section 4.24.9.2.

where:

ØPT as nonzero input:

IZ = List of connections for as many elements of the type as core space allows (Integer-output)
NWDS = Input-element type (2 character BCD, left adjusted)
Output-number of words in IZ (Integer)
ØPCØR = Open core available.
ØPT = Input-number of grid point connections per element to read from data block ELSETS (Integer). Output-number of connections per element (Integer)

ØPT as zero input:

IZ = List of grid point element connections and pointers to each grid point (Integer-output).
NWDS = Number of words in IZ prior to pointer array. Zero is array not created (Integer-output).
ØPCØR = Open core available.
ØPT = $\begin{cases} \text{NWDS if array IZ was created} \\ 0 \text{ if not created} \end{cases}$ (Integer input, output)

4. Method: For ØPT > 0 input, the ELSETS data block is open and the first two words (Element type and number of grid points per element) for this element type have been read. The element type is compared to a list of elements where special algorithms are needed. The connections are read and placed in IZ. For the special element types a zero is placed where the pen is to be lifted and the pen moved to a new location. A zero is always placed at the end of each element.

If insufficient core is available to read all the elements of the type both ØPT and NWDS are set negative and further calls must be made to complete the read. The minimum currently needed is 3 to 43 words depending on element type.

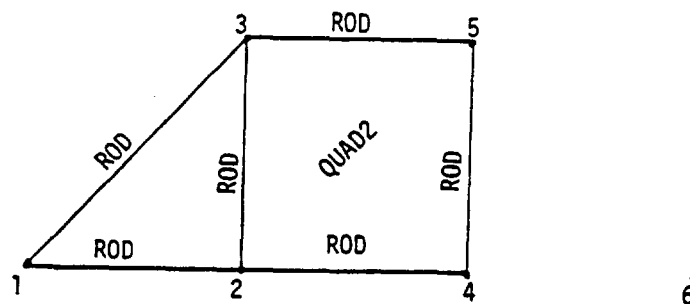
If the element has more than 4 grids and is not recognized, the element type is skipped and a warning message written. NWDS and ØPT are set to zero.

A fatal message 3002 results if ELSETS is incorrent or mispositioned. This should not result.

FUNCTIONAL MODULE PLOT (STRUCTURAL PLOTTER)

For ØPT as nonzero input, the ELSETS record is assumed positioned at the first word of the proper record. Much of the above code is used except for exit conditions and contents of IZ. If any failure occurs, ØPT and NWDS are zero on returning.

As the elements are read, IZ is filled with the 'lines' that would be drawn. The lowest number grid is the first of the pair of numbers for each 'line'. These 'lines' are then sorted and duplicates eliminated. Each pair is then reversed and appended to the IZ array. This is sorted and duplicates eliminated. From this NWDS long array, a connection table is created with pointers to the start of each grid loaded into core at IZ(NWDS+1). As an example the following set and model would contain the following IZ array.



	Word	Contents	description
Variable Length	1	2	
	2	3	connections to grid 1
	3	1	
	4	3	connections to grid 2
	5	4	
	6	1	
	7	2	
	8	5	
	9	2	
	10	5	
	11	3	
	12	4	
	13-24	---	only used for record sort
NGP+1 Length	25	1	word with first connection to grid 1
	26	3	word with first connection to grid 2
	27	6	
	28	9	
	29	11	
	30	13	
	31	13	<u>No</u> connections to grid 6

Note that the QUAD2 element did not affect the resultant array size or core requirements. In general the core size for the first sort may be exceeded (return without creating the array) by excessive number of duplicate lines (e.g., CRØD, CQDMEM, and CSHEAR all connecting the same points).

MODULE FUNCTIONAL DESCRIPTIONS

4.24.8.19 Subroutine Name: SUPLT

1. Entry Point: SUPLT
2. Purpose: To draw a structural shape with unique lines and line continuity when a grid point connection array is available (see LINEL).
3. Calling Sequence: CALL SUPLT (IZ,IY,X,U,GPLST,PEN,DEFØRM)
COMMON // - See description in Section 4.24.9.2.

where:

- IZ = Grid point connection list (see Section 4.24.8.18).
- IY = Grid point connection list pointers (see Section 4.24.8.18).
- X = Undeformed grid point coordinates (3 per grid point in the set - r, s, t).
- U = Deformed grid point coordinates (2 per grid point in the set - s, t).
- GPLST = List of indices into the subset of grid points pertaining to the shape to be drawn.
- PEN = Pen number or line density to be used to draw the shape.
- DEFØRM = 0 if the undeformed shape is to be drawn
1 if the deformed shape is to be drawn

4. Method: The first grid with an odd number of entries is located (if none, the first entry is used) and used as the first line. After this, the following flow diagram is followed (ID1 = starting grid point, ID2 = ending grid point).

- Step 1 Draw line ID1 to ID2. If no more lines exist, go to step 8. Remove the points from IZ. Set ID3=ID1 and ID4=ID2.
- Step 2 If more connections exist for ID4, set ID1=ID4. Otherwise, go to step 4.
- Step 3 Set ID2 to the closest entry (using the GPLST ID) to ID1. If two are equidistant use the increasing/decreasing sequence of ID3 to ID4. Go to step 1.
- Step 4 If more connections exist for ID3, set ID1=ID3 and go to step 3.
- Step 5 Otherwise search IZ for a new ID4*. Start the search to the near end of IZ and if that fails start at ID4 and search to the far end.
- Step 6 If ID4* has an odd number of entries, set ID1=ID4* and go to step 3.
- Step 7 Otherwise search all connections of ID4* for only one entry. If found set ID1= connection and if not found set ID1=ID4* and go to step 3.
- Step 8 For wrap-up, if the undeformed shape was being drawn, the IZ array is restored to its original values.

FUNCTIONAL MODULE PLOT (STRUCTURAL PLOTTER)

Note that the X or U array was not used to determine minimum pen movement. It is assumed that the grid point sequence is meaningful. Ninety percent of the time is spent in steps 1 to 3.

4.24.8.20 Subroutine Name: BØRDER

1. Entry Point: BØRDER
2. Purpose: To draw the structural outline
3. Calling Sequence: CALL BØRDER (GPLST,X,U,ISTØRE,DEFØRM,B1,ØPCØR)

COMMON //

where:

GPLST List of indices defining the subset of grid points

X Corresponding grid point coordinates of the undeformed structure

U Corresponding grid point coordinates of the deformed structure

ISTØRE Temporary storage area in open core

DEFØRM { 0 if the undeformed grid points are to be used
1 if the deformed grid points are to be used

B1 Buffer available in open core

ØPCØR Length of the ISTØRE area

4. Method: The subroutine BØRDER uses the table created on SCR2, scratch file 2 by subroutine ØRDER. BØRDER extracts from the table those grid points that lie on the plot sets boundary. All grid points connected to this grid point by element boundary lines are sorted to find the two element boundary lines which form the structural boundary. This subroutine then draws two connecting lines for each grid point to form the structural outline.

5. Additional Subroutines Required: LINE

4.24.8.21 Subroutine Name: CENTRE

1. Entry Point: CENTRE
2. Purpose: To find the intersection of two lines.

MODULE FUNCTIONAL DESCRIPTIONS

3. Calling Sequence: CALL CENTRE(X1,Y1,X2,Y2,X3,Y3,X4,Y4,CENTER),
 RETURNS(RETURN1)

where

X1,Y1 and X3,Y3 are the end points of one line

X2,Y2 and X4,Y4 are the end points of a second line

CENTER the point of intersection of the two lines.

4. Method: The point of intersection is calculated. The nonstandard return is taken if either line has infinite slope. It is assumed that the two lines are not parallel.

4.24.8.22 Subroutine Name: C0NT0R

1. Entry Point: CØNTØR
2. Purpose: To plot and label contour lines.
3. Calling Sequence: CALL CØNTØR(GPLST,X,S,U,Z,IZ,PEN,DEFØRM,BUF1,ØPCØR)

COMMON //

COMMON /XXPARM/

COMMON /PLTDAT/

COMMON /DRWDAT/

COMMON /SYSTEM/ BUFSIZ

where:

GPLST List of indices into the busset of gridpoints

X Corresponding grid point coordinates of the undeformed structure (3 per grid point)

S Cooresponding grid point coordinates of deformed structure

U Displacement coordinates

Z,IZ Available open core

PEN Line density or pen number

DEFØRM $\begin{cases} = 0 & \text{if undeformed points to be used} \\ = 1 & \text{if deformed grid points are used} \end{cases}$

BUF1 Index in open core to first GINØ buffer

ØPCØR Length of unused open core

FUNCTIONAL MODULE PLOT (STRUCTURAL PLOTTER)

4. Method: If the contour plot is to be a displacement plot, CØNTØR calls the subroutine DISPLA to plot the contour lines. If a stress plot was requested, CØNTØR calls the subroutine CREATE to prepare a table containing the contour value and the centroid for each element in the set. The subroutine ØRDER is then called if necessary, to provide a table of all grid points internal to the structural shape and all elements that share a common grid point in the current plot set.

Once these tables are available, CØNTØR estimates a contour value for each internal grid point by averaging the distance from the centroids and contour values of surrounding elements. Contour lines are found within a triangle formed by connecting an internal grid point and two adjacent elements' centroids. Contour lines are plotted with the assumption that the contour values vary linearly between vertices of this triangle.

Labels are placed on the completed contour plot and the subroutine PLTØPR is called to print the contour plot key as part of the plot messages.

5. Additional Subroutines Required: CREATE, DISPLA, LINE, ØRDER, PLTØPR, TYPINT

4.24.8.23 Subroutine Name: CREATE

1. Entry Point: CREATE

2. Purpose: To extract the stress contour values from the proper data block and calculate the centroid for each element in the set.

3. Calling Sequence: CALL CREATE(GPLST,X,U,DEFØRM,CØNMIN,CØNMAX,ELMTID,STØRE,LCØR,B1,B2)

CØMMØN //

CØMMØN /XXPARM/

where:

GPLST .	List of indices defining the set of grid points
X	Corresponding grid point coordinates of the undeformed structure
U	Corresponding grid point coordinates of the deformed structure
DEFØRM	$\left\{ \begin{array}{l} = 0 \text{ if undeformed grid points are to be used} \\ = 1 \text{ if deformed grid points are to be used} \end{array} \right.$
CØNMIN	Minumum of the contour values
CØNMAX	Maximum of the contour values

MODULE FUNCTIONAL DESCRIPTIONS

ELMTID	Temporary storage area in open core for the element ID
STØRE	Temporary storage area in open core for the stress contour value
LCØR	Length of the ELMTID and 1/2 length of STØRE storage areas
B1 } B2 }	GINØ buffers in open core

4. Method: After determining which stress data block has the contour values for this plot, the appropriate data block is opened and the temporary storage area is filled. The contour values are selected from the information on fibre distance, direction, subcase, and if applicable, range or time specified by the user. Then the ELSETS data block is read to supply information on element types, element identification numbers, and internal grid point numbers of elements contained in this set.

The subroutine makes two assumptions about the stress data block and the ELSETS data block. It assumes that element types are ordered in the same sequence and that for each element type, element information is ordered by increasing element ID. Element types for which contour lines cannot be drawn are ignored as are elements not in this plot set.

Finally the centroid is calculated for each element and a temporary table is made containing the element ID, the contour value, and the centroid. SCR1, scratch file 1, is generated by the subroutine CREATE to be used by the subroutine CØNTØR. The header record is written by the subroutine GØPEN. The table is comprised of information contained in the ELSETS data block and the ØES1 data block or the NEWØES table.

Table Format:

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>			
0	1	I	Header Record			
1	1	I	Element Type			
	2	I	Element Identification Number	}	} repeated for each element type in contour type set	
	3	R	Stress Value			
	4,5	R	Element Centroid			
	.	.	.	}		repeated for each element in plot
	.	.	.			
	.	.	.			
	(4m+2)	I	0			
2			End-of-File			

Note: m = the number of elements of current type in the plot set.

FUNCTIONAL MODULE PLOT (STRUCTURAL PLOTTER)

5. Additional Subroutines Required: CENTRE

4.24.8.24 Subroutine Name: DISPLA

1. Entry Point: DISPLA

2. Purpose: To draw displacement contour lines.

3. Calling Sequence: CALL DISPLA(GPLST,X,S,U,PEN,DEFØRM,LABEL,PT,B1)

COMMON //

COMMON /XXPARM/

COMMON /PLTDAT/

where:

GPLST List of indices defining the set of grid points

X Corresponding grid point coordinates for undeformed structure

S Corresponding grid point coordinates for deformed structure

U Grid point displacement coordinates

PEN Line density or pen number

DEFØRM $\begin{cases} =0 & \text{if the undeformed points are to be used} \\ =1 & \text{if the deformed points are to be used} \end{cases}$

LABEL Contour line labeling information

PT Coordinates of the triangle formed to calculate contour lines

B1 GINØ buffer in open core

4. Method: DISPLA uses the ELSETS data block to extract the two dimensional elements in the plot set. Each element is then broken down to one or more triangles and contour lines are drawn within the triangles with the assumption that the contour values vary linearly between grid points.

It is assumed that the user's coordinate system is a righthanded rectangular coordinate system. If the AXES card is used to change the orientation of the structural model, it will have no effect on the component of displacement used to draw the contour plot.

Labeling information is maintained on a temporary file to be added by subroutine CØNTØR.

5. Additional Subroutines Required: LINE

MODULE FUNCTIONAL DESCRIPTIONS

4.24.8.25 Subroutine Name: ØRDER

1. Entry Point: ØRDER
2. Purpose: To find for each grid point in the plot set, all elements which contain that grid point and to order those elements surrounding interior grid points.
3. Calling Sequence: CALL ØRDER(GPLST,ID,NUMBS,REST,IDTAB,LCØR,B1,B2,B3)

COMMON //

where:

GPLST List of indices defining the subset of grid points
ID Temporary storage area in open core for element ID
NUMBS Temporary storage area in open core for SILS
REST Temporary storage area in open core for grid points
IDTAB Temporary storage area in open core for element ID's GPECT index
LCØR Length of temporary storage areas
B1
B2 Available GINØ buffers in open core
B3

4. Method: ØRDER constructs from data block ELSETS a large in core table containing element IDs, the corresponding element internal IDs, and the ordered grid points associated with each element. The subroutine then takes from the ECPT, the list of element pointers connected to each grid point for each grid point in the plot set. Using the table, ØRDER then finds the proper element ID and its associated grid points. For each grid point in the ECPT, ØRDER finds all two dimensional elements containing that (pivot) grid point. The elements are ordered so that adjacent elements contain a second common grid point. A flag is set indicating whether the pivot grid point is interior to the structural shape or on the boundary of the structural shape.

A temporary table is generated containing one record per grid point. Each record contains a flag indicating whether the grid point is interior to the structural model or on the boundary of the model, and a list of elements connected to the grid point. For boundary grid points, the table contains two grid points for each element that are connected by element boundary lines to that grid point.

FUNCTIONAL MODULE PLOT (STRUCTURAL PLOTTER)

The table will be written to SCR2, scratch file 2, and is to be used by subroutines CØNTØR and BØRDER. The table is in the following format:

<u>Record</u>	<u>Word</u>	<u>Type</u>	<u>Item</u>	
0			Header Record	
1	1	I	Flag	} repeated for each grid point in plot set
	2	I	Number of elements, m.	
	3	I	Grid point internal number	
	4	I	repeated for each element containing the grid point.	
	.	.	Contents dependent on Flag.	

Notes:

- 1) n = the number of grid points in the plot set.
- 2) m = the number of elements in current record.
- 3) Flag = -1:

<u>Word</u>	<u>Type</u>	<u>Item</u>
4	I	Element identification number
.	.	.
.	.	.
.	.	.
(3+m)		

- 4) Flag = -2:

<u>Word</u>	<u>Type</u>	<u>Item</u>
4	I	Element identification number
5	I	Grid point internal number
6	I	Grid point internal number
.	.	.
.	.	.
.	.	.
(3m+1)	I	Element identification number
(3m+2)	I	Grid point internal number
(3m+3)	I	Grid point internal number

MODULE FUNCTIONAL DESCRIPTIONS

4.24.8.26 Subroutine Name: RØTAT

1. Entry Point: RØTAT
2. Purpose: To transform the normal stresses and the shear stresses to a common coordinate system.
3. Calling Sequence: CALL RØTAT(ECT2,B1,GPLST,X)
COMMON //
COMMON /XXPARM/

where:

ECT2 Data block containing element connection data
B1 Include buffer in open core
GPLST List of indices defining the set of grid points
X Corresponding grid point coordinates

4. Method: The subroutine transforms the normal stresses and shear stresses from the local coordinate system to a common coordinate system.

The following expression is used:

$$[T] = [\theta] [T'] [\theta]^T$$

where

$$[T'] = \begin{bmatrix} \sigma_x & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma_y & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma_z \end{bmatrix} \quad \text{is a symmetric matrix}$$

and $[\theta]$ is the rotation matrix which has as its components the direction cosines between the local and common coordinate systems. σ_x , σ_y , σ_z are the normal X, Y, and Z stresses respectively and τ_{xy} , τ_{xz} , τ_{yz} are the shear XY, XZ, and YZ stresses respectively.

Since all elements considered by the contour plotter are plate elements, the elements are all assumed to lie in the local X,Y plane with the X-axis passing from grid point one to grid point two. A new local element stress table (NEWØES) is generated to communicate the transformed stresses to subroutine CØNTØR and subroutine CREATE. There is no header

FUNCTIONAL MODULE PLOT (STRUCTURAL PLOTTER)

record. The table is generated using stress values contained in the ØES1 data block.

The first record in each set of two records on NEWØES is a copy of the corresponding record of the ØES1 data block. The second record of each set contains changes in content for the following element types:

Element types CQUAD1, CQUAD2, CTRBSC, CTRIA1, CTRIA2, CTRPLT

<u>Word</u>	<u>Contents</u>
2	Z1 (Fibre Distance 1)
3	Normal X Stress at Z1
4	Normal Y Stress at Z1
5	Normal Z Stress at Z1
6	Shear XY at Z1
7	Shear XZ at Z1
8	Shear YZ at Z1
9	Maximum Shear at Z1
10	Z2 (Fibre Distance 2)
11	Normal X Stress at Z2
12	Normal Y Stress at Z2
13	Normal Z Stress at Z2
14	Shear XY at Z2
15	Shear XZ at Z2
16	Shear YZ at Z2
17	Maximum Shear at Z2

Element types CQDMEM, CQDMEM1, CQDMEM2, CQDPLT, CTRMEM

<u>Word</u>	<u>Contents</u>
2	Normal X Stress
3	Normal Y Stress
4	Normal Z Stress
5	Shear XY
6	Shear XZ
7	Shear YZ

MODULE FUNCTIONAL DESCRIPTIONS

4.24.9 Design Requirements

4.24.9.1 Open Core Design (Common Block XXPLØT)

Define NPSET = Number of element plot sets.
 NGP = Total number of grid points.
 NGPSET = Number of grid points in an element set.

1. Subroutine DPLØT partitions open core for subroutine PARAM as follows:

	COMMON/XXPLØT/
X(0)	Element plot set id's
X(NPSET)	Open Core
X(BUF)	4 GINØ Buffers

2. Subroutine PARAM partitions open core for subroutines FIND(FNDSET,PERPEC,PRØCES) as follows:

X(0)	Element plot set id's
X(NPSET+1)	GPLST(NGP) Grid point indices into a subset of grid points
X(NGP+NPSET+1)	X(3,NGPSET) Coordinates of the grid points in the subset
X(NGP+3*NGPSET+ NPSET+1)	Rest of open core

FUNCTIONAL MODULE PLOT (STRUCTURAL PLOTTER)

3. Open core for subroutine PLØT is as follows:

OPEN CORE LAYOUT FOR PLØT MODULE

COMMON/XXPLØT/

<u>NAME</u>	<u>TYPE</u>	<u>LENGTH</u>	<u>DESCRIPTION</u>
	I	NPSET	Element plot set id's
	I	NDEF	List of specified deformation subcases
	A	M	Rest of "PLØT" card after subcase ID
GPLST(1)	I	NGP	Grid point indices
X(1,1)	R	3*NGPSET	Coordinates of undeformed points in plot subset
U(1,1)	R	3*NGPSET	Coordinates of deformed points in plot subset
S(1,1)	R	2*NGPSET	Sand T coordinates of deformed points
IZ(ID)	I	LCØR	Element id's. Note that ID=1 (see note 1 below)
IZ(ISAV)	I	LCØR	Temporary storage area (ISAV=ID+LCØR) (see note 1 below)
Z(IVAL)	R	LCØR	Element's stress values (IVAL=ISAV+LCØR) (see note 1 below)
Z(ICEN)	R	2*LCØR	Element's centroid values in pairs (ICEN=IVAL+LCØR)(see note 1 below)
		BUFSIZ	PLT DSP GINØ buffer
		PLTBUF	Plot tape buffer
GPLST(B3)		BUFSIZ	Scratch GINØ buffer
GPLST(B2)		BUFSIZ	Scratch GINØ buffer (PARM buffer)
		BUFSIZ	PLØTX GINØ buffer
GPLST(B1)		BUFSIZ	Scratch GINØ buffer

Notes:

1) $LCØR = ØPCØR / 5$

where

$ØPCØR$ = available open core that exists between S(2,NGPSET) and the beginning of the first GINØ buffer.

MODULE FUNCTIONAL DESCRIPTIONS

4.24.9.2 Block Data Interface

1. COMMON/DRWDAT/ SET,LABEL,ORIGIN,PEN,SHAPE,SYMBOL(2),SYM(6),VECTOR,CØN,EDGE

SET - Element plot set index.

LABEL - Grid point label option.

ORIGIN - Origin index.

PEN - Pen number or density value.

SHAPE - Structural shape drawing option.

SYMBOL - Grid point symbol indices.

SYM - Symmetry options.

VECTOR - Deformation vector options.

CØN - Contour plot option.

EDGE - Outline option.

} Integer

2. COMMON/RSTXXX/ CSTM(3,3),MIN(3),MAX(3),D(3),AVER(3)

CSTM - 3x3 coordinate system rotation matrix.

MIN - Minimum rotated grid point coordinates.

MAX - Maximum rotated grid point coordinates.

D - Differences between the minima and maxima.

AVER - Averages of the minima and maxima.

} Real

3. COMMON/XXPARM/ PBUFSZ,CAMERA,BFRAMS,PLTMDL(2),TAPDEN,
NPENS,PAPSIZ(2),PAPTYP(2),PENSIZ(8),PENCLR(8,2),"SKIP(1)",
SCALE,ØBJMØD,FSCALE,MAXDEF,DEFMAX,
AXIS(3),DAXIS(3),VANGLE(3),BETAØS,BETAP,"SKIP(4)",
FVP,RØ,SØL,SØR,TØ,DØ,DØ2,DØ3,PROJECT,SØS,
FØRG,ØRG,NØRG,ØRIGIN(11),EDGE(11,4),XY(11,3)
NCNTR,CNTR(50),ICNTVL,WHERE,DIRECT,
SUBCAS,FLAG,VALUE,SET

In the following descriptions, the value(s) in parentheses to the right of the variable name, the default value, and the letter in parentheses to the right of the explanation pertain to the type of the variable (I implies integer and R implies real).

PBUFSZ(0) = Plot tape buffer size (I)

FUNCTIONAL MODULE PLOT (STRUCTURAL PLOTTER)

Plotter Data

CAMERA(2)	= Plotter camera number (I).
BFRAMS(1)	= Number of blank frames between plots (I).
PLTMDL(4020,0)	= Plotter model (BCD or I or R).
TAPDEN(0)	= Plot tape density (I).

Pen and Paper Data

NPENS(8)	= Maximum number of pens (I).
PAPSI(8.5,11.)	= Paper size in inches (R).
PAPTYP(VELLUMbb)	= Paper type (BCD).
PENSIZ _i (1)	= Pen sizes (I).
PENCLR _{i,1} (BLAC) & PENCLR _{i,2} (Kbbb)	= Pen colors (BCD).

Scaling Data

SCALE	= Object-to-plotter or model-to-plotter (stereo only) scale factor (R).
OBJMOD(1.)	= Object-to-model scale factor (R-stereo only).
FSCALE(1)	= Find scale factors option (I).
MAXDEF(0.)	= Forced value of the largest deformation component (R).
DEFMAX	= Actual largest deformation component (R).

Viewing Data

AXIS (1,2,3)	= Undeformed structure axis orientation (I).
DAXIS(1,2,3)	= Deformation symmetry/antisymmetry axis orientation (I).
VANGLE(0.,-1.10 ¹⁰ , 34.27)	= View angles (R-alpha,beta,gamma).
BETAØS(23.17)	= Orthographic and stereo default value for the "beta" view angle (R).
BETAP(0.)	= Perspective default value for the "beta" view angle (R).

MODULE FUNCTIONAL DESCRIPTIONS

Projection Data

FVP(1)	=	Find vantage point option (I).
RO	=	"r" component of the vantage point (R).
SOL	=	"s" component of the perspective vantage point (R).
SOL,SOR	=	"s" components of the stereo vantage point (R).
TO	=	"t" component of the vantage point (R).
DO	=	Projection plane separation value (R).
DO2(1.)	=	Perspective default projection plane separation value (R).
DO3(2.)	=	Stereo default projection plane separation value (R).
PROJECT(1)	=	Projection type (I, 1=orthographic, 2=perspective, 3=stereo).
SOS(2.756)	=	Ocular separation value (R).

Origin Data

FORG(1)	=	Find origin point option (I).
ORG(0)	=	Number of active origins (I).
NORG(10)	=	Maximum number of active origins (I).
ORIGIN	=	Active origin id's (I).
EDGE _{i,1} (0.) & EDGE _{i,2} (0.)	=	Lower left corner of the region specified for the i th origin (R).
EDGE _{i,3} (1.) & EDGE _{i,4} (1.)	=	Upper right corner of the region specified for the i th origin (R).
XY _{i,1}	=	x component of the i th origin (R).
XY _{i,3}	=	y component of the i th origin (R).
XY _{i,1} & XY _{i,2}	=	left and right x components of the i th origin for stereo projection (R).

Contour Plotting Data

NCNTR(10)	=	Number of contour values to be plotted (I).
CNTR	=	Contour values (R).
ICNTVL(3)	=	Type of contour plot (I).
WHERE(1)	=	Fibre distance of stress contour value (I).
DIRECT(2)	=	Stress vector direction option (I).

FUNCTIONAL MODULE PLTSET (PLOT SET DEFINITION PROCESSOR)

SUBCAS = Current subcase (I).
FLAG(0.0) = Data identification indicator (R).
VALUE = Current eigenvalue or time step (R).
SET = Last plot set processed by contour plotter (I).

4. COMMON// NGP,LSIL,NSETS,PLTFLG,PLTNUM,
NGPSET,NODEF,SKP1(3),
PLTPAR,GPSETS,ELSETS,CASECC,BGPD, EQEXIN,SIL,
PDEF1,PDEF2,SKP2,PLDXTX,SETD

MODULE FUNCTIONAL DESCRIPTIONS

THIS PAGE HAS BEEN LEFT BLANK INTENTIONALLY.

FUNCTIONAL MODULE PLOT (STRUCTURAL PLOTTER)

PLØT Module Parameters

- NGP - Total number of grid points (I).
- LSIL - Last scalar index value (I)
- NSETS - Number of element plot sets (I)
- PLTFLG - Displacement plot flag
- PLTNUM - Plot number

PLØT Parameters

- NGPSET - Number of grid points in the set being plotted
- NØDEF - 0 - no deformed plot requests were skipped when PLTFLG was 1.
1 - deformed plot requests were skipped when PLTFLG was 1.
- SETD - Default set to use if current set requested does not exist.

GINØ File names

- PLTPAR,GPSETS,ELSETS,CASECC,BGPDT,
EQEXIN,SIL,PDEF1,PDEF2 -- Input
- PLØTX - Output

4.24.9.3 Common Storage Requirements

1. /XXPLØT/ - Open core.
2. /PLTDAT/ - Plotter data (see miscellaneous table description, Section 2.5).
3. /XXPARM/ - Plotting parameters.
4. /RSTXXX/ - Plot coordinate system calculations.
5. /DRWDAT/ - Drawing data.

4.24.10 Diagnostic Messages

A non-fatal message will be generated by subroutine PLØT and/or FIND if not enough core is available for the grid point data needed for a specific element plot set. If this occurs, this set will not be used to generate a plot.

Elements that require a special algorithm for drawing lines, and the algorithm is not present, will be rejected as an illegal element. Special algorithms (subroutine LINEL) are needed for elements with more than four grid points or special line connection patterns (e.g., the CTETRA). Module PLTSET has less restrictive tests. This is a systems problem and should not occur.

MODULE FUNCTIONAL DESCRIPTIONS

Illegal end-of-records and end-of-files will generate fatal messages.

All non-fatal diagnostics are written onto the PLØTX message data block for printing by the PRTMSG module. These messages are all quite self-explanatory and straightforward, and do not have any external message numbers.

FUNCTIONAL MODULE GP3 (GEOMETRY PROCESSOR - PHASE3)

4.25 FUNCTIONAL MODULE GP3 (GEOMETRY PROCESSOR - PHASE 3)

4.25.1 Entry Point: GP3

4.25.2 Purpose

GP3 processes static loads and temperature data. Static load data are collected by set, and external numbers are converted to internal numbers. Similarly, temperature data are collected by temperature set and external numbers are converted to internal numbers.

4.25.3 DMAP Calling Sequence

```
GP3      GEOM3,EQEXIN,GEOM2/{HSLT},GPTT/V,N,NOL0AD/V,N,N0GRAV/V,N,N0TEMP $
```

4.25.4 Input Data Blocks

GEØM3 - Static loads and temperature data.

EQEXIN - Equivalence between external grid and scalar numbers and internal numbers.

GEOM2 - Element connection data.

Note: EQEXIN may not be purged.

4.25.5 Output Data Blocks

HSLT - Static loads table for heat problems.

SLT - Static Loads Table.

GPTT = Grid Point Temperature Table.

4.25.6 Parameters

NØLØAD - Output-integer-no default. -1 if no static loads (i.e. SLT is not created),
+1 otherwise.

NØGRAV - Output-integer-no default, -1 if no GRAV cards in the Bulk Data Deck, +1 otherwise.

NØTEMP - Output-integer-no default. -1 if no TEMP or TEMPD cards in Bulk Data Deck (or if GPTT is purged), +1 otherwise.

4.25.7 Method

Subroutine GP3 is the control program for the module. It executes each of the major subroutines of GP3 (GP3C, GP3A, GP3B) depending on the status of the data blocks. A flow chart for GP3 is included in Figure 1.

4.25.8 Subroutines

4.25.8.1 Subroutine Name: GP3

1. Entry Point: GP3
2. Purpose: Module control program.
3. Calling Sequence: CALL GP3

4.25.8.2 Subroutine: GP3C

1. Entry Point: GP3C
2. Purpose: To convert PLØAD2 data to PLØAD format, merge PLØAD2 data with PLØAD data (if present) and write the resulting data on SCR2, a scratch file.
3. Calling Sequence: CALL GP3C
4. Method: PLØAD2 cards are read into core from GEØM3. Six words are used for each entry. The first word (set identification) is set negative and the sixth word of each entry is set to zero. GEØM2 is opened and the header record is skipped. The following steps occur for each record on GEØM2.

1. The 3-word header is read. /GPTA1/ (see section 2.5) is searched for a match. If ~~no match~~ is found, the record is skipped and the process is repeated. If an end-of-file is encountered, step (3) is executed. If a match is found, a test on element type is made. If a one-dimensional element, the record is skipped and the process repeated. Otherwise, step (2) is executed.

2. An entry of the current element type is read. A linear search through the PLØAD2 data in core is made to find a match on element identification (3rd word of each PLØAD2 entry). If no match is found, the next entry is read. For each match which is found, the grid identification numbers which connect the element are stored

in the corresponding PLØAD2 entry and the first word of the PLØAD2 entry is set positive. When all data for the current element type has been read, a return to step (1) is made.

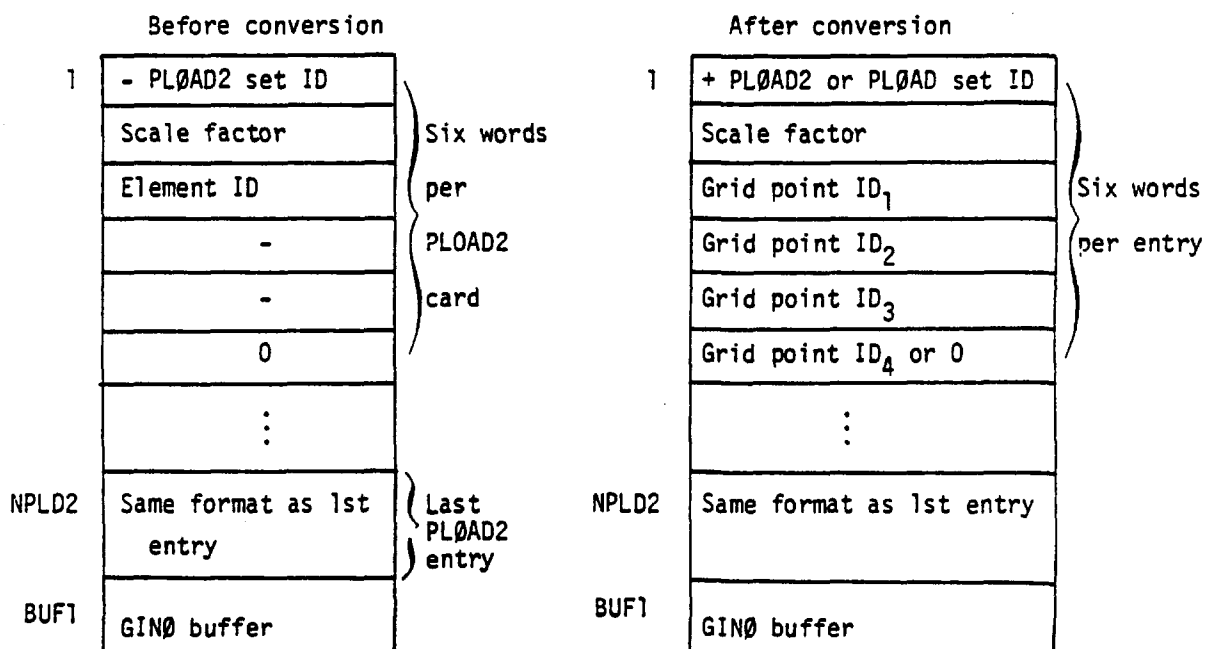
3. A pass through each entry in the PLØAD2 data is made. For each entry for which the first word is negative, an error message is queued and the NØGØ flag turned on. Upon completion of the pass, PEXIT is called if the NØGØ flag was turned on. Otherwise step (4) is executed.

4. LØCATE is called to position GEØM2 to PLØAD data. If none exists, step (5) is executed. Otherwise, the PLØAD data is read into core following the PLØAD2 data. The combined list is sorted by SØRT on set identification number.

5. The data in core is written as one logical record on SCR2. A return to GP3 is given.

Allocation of core storage in GP3C is as follows:

CØMMØN/GP3CØR/Z(1)



MODULE FUNCTIONAL DESCRIPTIONS

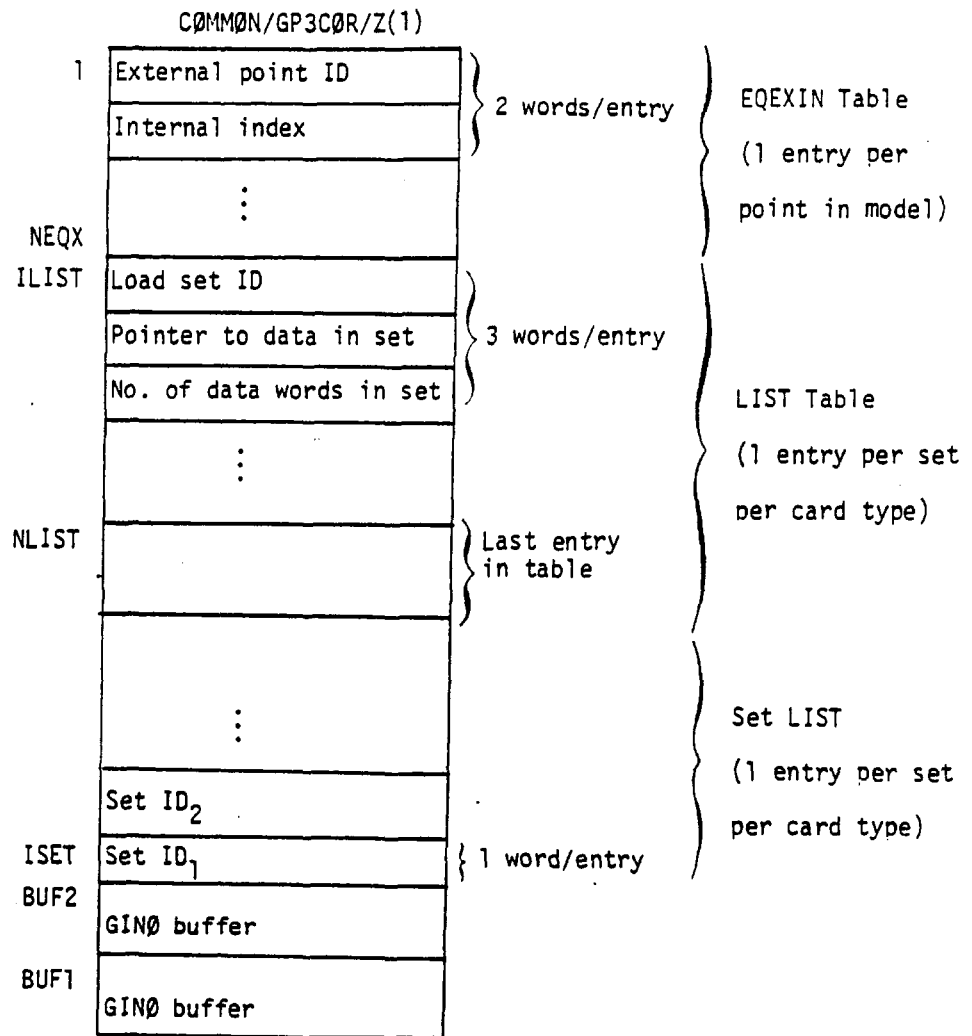
4.25.8.3 Subroutine: GP3A

1. Entry Point: GP3A
2. Purpose: To assemble to Static Loads Table (SLT).
3. Calling Sequence: CALL GP3A
4. Method: GP3A assembles the SLT by making two passes on the load cards (FØRCEi, MØMENTi, etc). On the first pass each of the cards is read from GEØM3, (or SCR2 for PLØAD data), unique set identifications are extracted and saved in core, all external point identifications are converted to internal indices by performing a binary search in the EQEXIN table, the data are written on SCR1, and pointer tables are accumulated. These tables are as follows:

STATUS			
1	Pointer in LIST table to first entry of card type	}	2 words/entry
2	Pointer in LIST table to last entry of card type		
	.	}	1 entry per card type
	.		
	.		
NTYPES		}	Last Entry

Note: entry = (-1, -1) if card type not present

FUNCTIONAL MODULE GP3 (GEOMETRY PROCESSOR - PHASE 3)

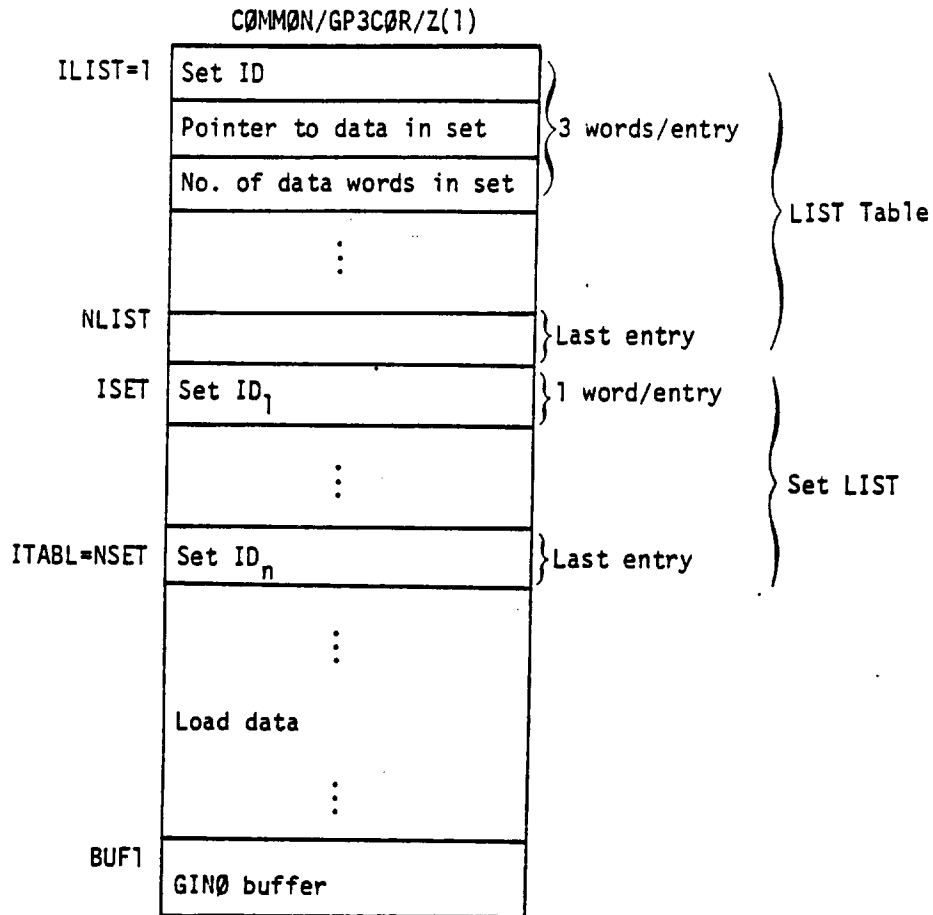


Note:

Set IDs are stored backward in core. ISET points to first entry,
ISET-1 to record entry, etc.

MODULE FUNCTIONAL DESCRIPTIONS

At the end of the first pass, the LIST table is moved to the beginning of open core. The set list is sorted, and duplicate set identifications are discarded. The resulting list is stored immediately following the LIST table. If all data for the load cards will fit in the remaining core, this data is read from SCR1. Core storage is as follows:



FUNCTIONAL MODULE GP3 (GEOMETRY PROCESSOR - PHASE 3)

The SET list is written in the header record on the SLT. For each set ID in the SET list, the LIST table is searched for a match. When found, the pointer to the data is fetched. The data are sorted on the applied point (except GRAV and PLØAD data) and the data written on the SLT. As a result, each logical record of the SLT contains all data for one set. Finally, if combination load cards are present, they are copied from GEØM3 to the last record of the SLT.

If core will not hold the entire load data, the logic is similar to above except that SCR1 is passed once for each set and only data belonging to a single card type within a set are read into core.

4.25.8.4 Subroutine Name: GP3B

1. Entry Point: GP3B
2. Purpose: To assemble the grid point related temperature data and store on the SCRATCH1 file.
3. Calling Sequence: CALL GP3B
4. Method: EQEXIN is read into core. A list of default temperatures (TEMPD cards if present) is read from GEØM3. The temperature data (TEMP cards) are read to determine the number of temperature sets, the set identifications and the number of entries in each temperature set. For each temperature set, a three-word entry is written in the header record of the SCRATCH1:

Word 1 = Set ID

Word 2 = default temperature (real) or -1 (integer)

Word 3 = record number in SCRATCH1 of temperature data for the set or zero if only default temperature is defined.

GEØM3 is backspaced one logical record. The temperature data are re-read. When all temperature data for a set have been read into core, the data are sorted on point identification and written as one logical record on SCRATCH1. This process is repeated for each temperature set.

Allocation of core storage for GP3C is as follows:

MODULE FUNCTIONAL DESCRIPTIONS

COMMON/GP3CØR/Z(1)		
1	External point ID	} 2 words/entry
	Internal index	
	⋮	
		} EQEXIN Table (1 entry per point in model)
ITEMPD	Temperature Set ID	} 2 words/entry
	Default Temperature	
	⋮	
		} Default temperature 1 entry per set
ITABL+1	Number of data words in set 1	} 1 word/entry
	⋮	
		} Definition of data in temperature sets
N1	Point ID	} 2 words/entry
	Temperature	
	⋮	
		} Temperature data for one set
BUF2	GINØ buffer	
BUF1	GINØ buffer	

MODULE FUNCTIONAL DESCRIPTIONS

4.25.8.5 Subroutine Name: GP3D

1. Entry Point: GP3D
2. Purpose: To process TEMPP1, TEMPP2, TEMPP3, and TEMPRB data and assemble the element temperature table referred to as the GPTT.
3. Calling Sequence: CALL GP3D
4. Method: TEMPP1, TEMPP2, TEMPP3, and TEMPRB card data are read from GEOM3, converted for GP3D's use and written on SCRATCH2. The grid point temperature data header is then read from SCRATCH1 (as created in SP3B). A similar header record is then constructed from the union of the grid point temperature set data and the element temperature set data. This is written on GPTT. For each temperature set, for which there is other than a default temperature available, a record is then written on the GPTT containing specific element temperature data by element type and element identification. Following the element temperature data, the grid point temperature data block, input from SCRATCH1, is written on the GPTT data block. Allocation of core storage for GP3D is as follows:

Element temperature set list data (2 words/entry)	Z(1) Z(NLIST) Z(IGPTT)
Grid point temperature set list data (3 words/entry)	Z(NGPTT) Z(IGPT)
Grid point data for current temperature set (2 words/entry)	Z(NGPT) Z(IET1)
TEMPP1, TEMPP2, TEMPP3 data for current temperature set (7/words/entry)	Z(NET1) Z(IET2)
TEMPRB data for current set ID (15 words/entry)	Z(NET2)
: unused	
GINØ buffer	Z(BUF2)
GINØ buffer	Z(BUF1)

4.25.9 Design Requirements

4.25.9.1 Allocation of Core Storage

The core storage maps presented in the method sections of GP3A, GP3B and GP3C provide detailed storage requirements. A summary is presented here.

GP3C: Maximum requirement = $6 * (\text{number of PL0AD2} + \text{number of PL0AD cards}) + \text{one GIN0 buffer.}$

GP3A: Let $NPTS = \text{number of grid} + \text{number of scalar points}$ and $NSETS = (\text{number of load sets}) * (\text{number of card types per load set})$ and $SYSBUF = \text{one GIN0 buffer.}$

Then maximum storage requirement equals $\text{MAX} ((2*NPTS+4*NSETS+2*SYSBUF), (4*NSETS + \text{MAX} (\text{number of words for one set of one card type} + SYSBUF)))$.

GP3B: See storage map.

4.25.9.2 Environment

1. Block Data

The block data program GP3BD initializes /GP3C0M/ with GIN0 file names, data defining the load cards and other miscellaneous data. It must be resident in core when GP3 is executed.

2. General

/GPTA1/ is used by GP3C and must be core resident when GP3 is executed. Open core is defined by /GP3C0R/. The normal overlay is to include GP3BD, GP3, GP3C, GP3A, GP3B in one segment. GP3 uses two scratch files.

4.25.10 Diagnostic Messages

The following messages may be issued by GP3:

2008, 2009, 2015, 3008, 3304, 3305, 4010, 4011 and 4012. See Section 6 of the User's Manual for details.

MODULE FUNCTIONAL DESCRIPTIONS

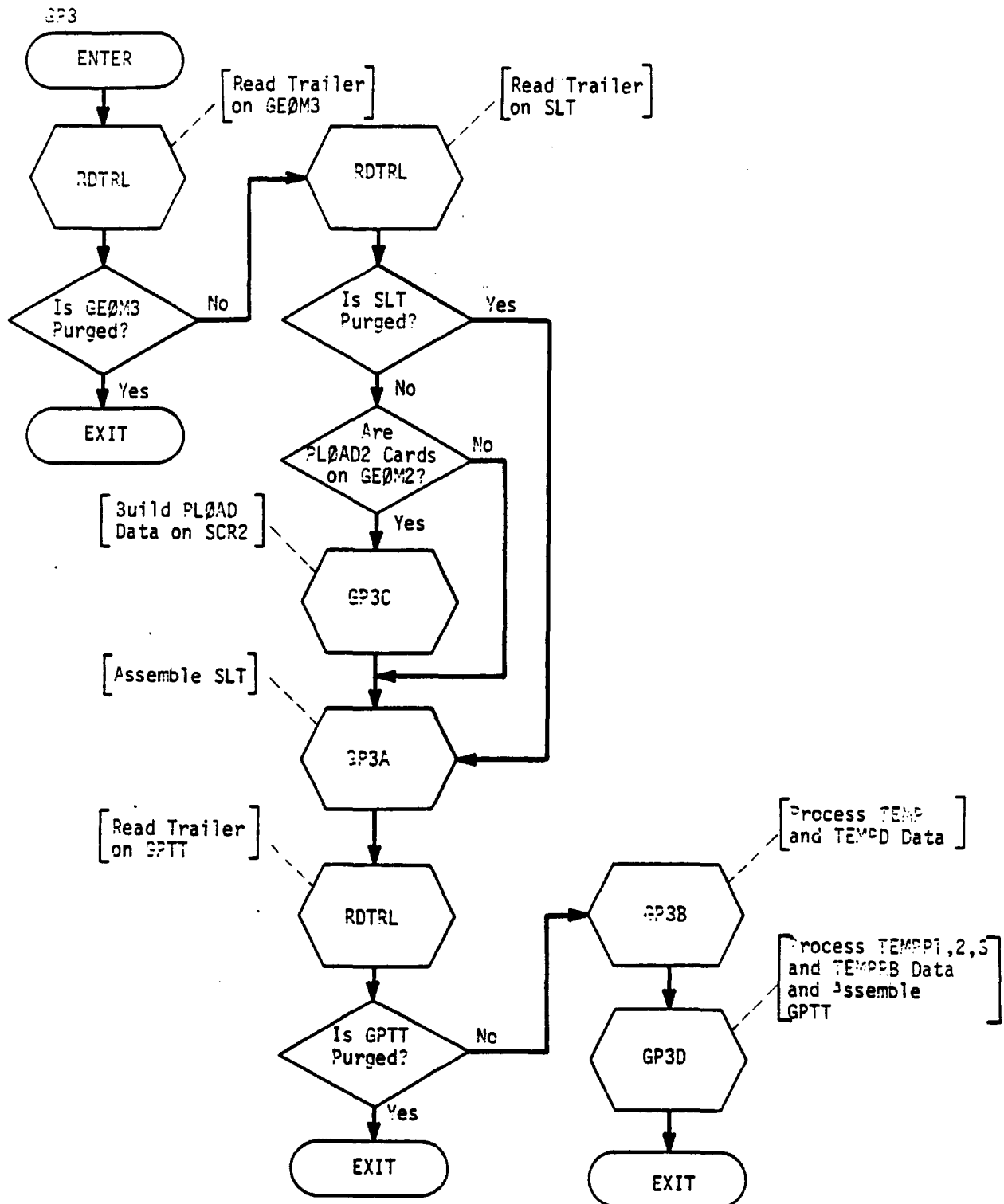


Figure 1. Flowchart for module GP3

FUNCTIONAL MODULE TA1 (TABLE ASSEMBLER)

4.26 FUNCTIONAL MODULE TA1 (TABLE ASSEMBLER)

4.26.1 Entry Point: TA1

4.26.2 Purpose

TA1 processes element connection data, element property data and geometry. These data are merged in two different sorts for efficiency in later processing. The Element Summary Table contains, for each element, connection, property and geometry data. The Element Connection and Properties Table contains, for each grid or scalar point in the model, connection, property and geometry data for all elements connected to the point. Element temperature data are also included in the EST where applicable. Additionally, general elements are processed and the GEI (General Element Input) data block is assembled. The Grid Point Element Connection Table contains, for each grid or scalar point in the model, connection data for all elements connected to the point.

4.26.3 DMAP Calling Sequence

When using SMA1 and SMA2:

```
TA1  ECT,EPT,BGPD,T,SIL,GPTT,CSTM / EST,GEI,,ECPT,GPCT / V,N,LUSET / V,N,NØSIMP /  
      C,N,O / V,N,NØGENL / V,N,GENEL $
```

When using EMA:

```
TA1  ECT,EPT,BGPD,T,{SIL},{GPTT},CSTM / {HEST},GEI,{HGPECT},ECPT,GPCT / V,N,LUSET /  
      {HSIL},{GPD},      {EST},      {GPECT}  
      V,N,NØSIMP / C,N,1 / V,N,NØGENL / V,N,GENEL $
```

4.26.4 Input Data Blocks

ECT - Element Connection Table
EPT - Element Properties Table
BGPD - Basic Grid Point Definition Table
SIL - Scalar Index List
HSIL - Scalar Index List for heat problems
GPTT - Grid Point Temperature Table
GPD - Grid Point Definition Table
CSTM - Coordinate System Transformation Matrices

Note: The ECT, BGPD and SIL data blocks may not be purged.

4.26.5 Output Data Blocks

- EST - Element Summary Table.
- HEST - Element Summary Table for heat problems.
- GEI - General Element Input.
- ECPT - Element Connection and Properties Table.
- GPCT - Grid Point Connection Table.
- GPECT - Grid Point Element Connection Table.
- HGPECT - Grid Point Element Connection Table for heat problems.

4.26.6 Parameters

- LUSET - Input-integer-no default. Degrees of freedom in the g-displacement set.
- NØSIMP - Output-integer-no default. Number of elements in the model (exclusive of general elements) or -1 if no elements.
- NØGENL - Output-integer-no default. Number of general elements in the model or -1 if no general elements.
- GENEL - Output-integer-no default. GENEL = -NØGENL.

4.26.7 Method

4.26.7.1 General Comments

The purpose of the Table Assembler Module is to combine all of the element data in a convenient form for the generation of the structural matrices (ECPT or GPECT) and for the calculation of the element stresses and forces (EST). The complete description of an elements requires: (1) the locations of the connected grid points, (2) necessary orientation data and end conditions, (3) element properties, (4) a material reference, (5) transformations from the basic system to the global coordinate system and (6) element temperature. Scalar elements require no geometric or material data. General elements may require geometric data.

Four data blocks are formed in this module. The ECPT (or GPECT) data block is used in structural matrix generation. It contains all element data for each grid point or scalar point in the order of the sequenced grid point numbers (internal grid point indices). The EST data block contains element data in groups of element type and with sequential element I.D. numbers within each group. It is used to calculate element stresses and forces in a convenient order for output. The GEI data block contains the general element stiffness or flexibility and support matrices. The GPCT data block is used to allocate storage in the structural matrix assemblers.

FUNCTIONAL MODULE TA1 (TABLE ASSEMBLER)

The reason for assembling the ECPT and EST tables rather than generating functions such as element stiffness matrices and stress functions is the expected size of the problems. The computing time used to recalculate certain data is expected to be compensated for by the time savings that result from sorting and merging smaller tables.

Subroutine TA1 is the main control program for the module. It executes each of the major routines of the Table Assembler (TA1A to assemble the EST, TA1B to assemble the ECPT and GPCT (or TA1H to assemble the GPECT), and TA1C to assemble the GEI) depending on the status of the data blocks and data for the problem. A flow chart of TA1 is included as Figure 1.

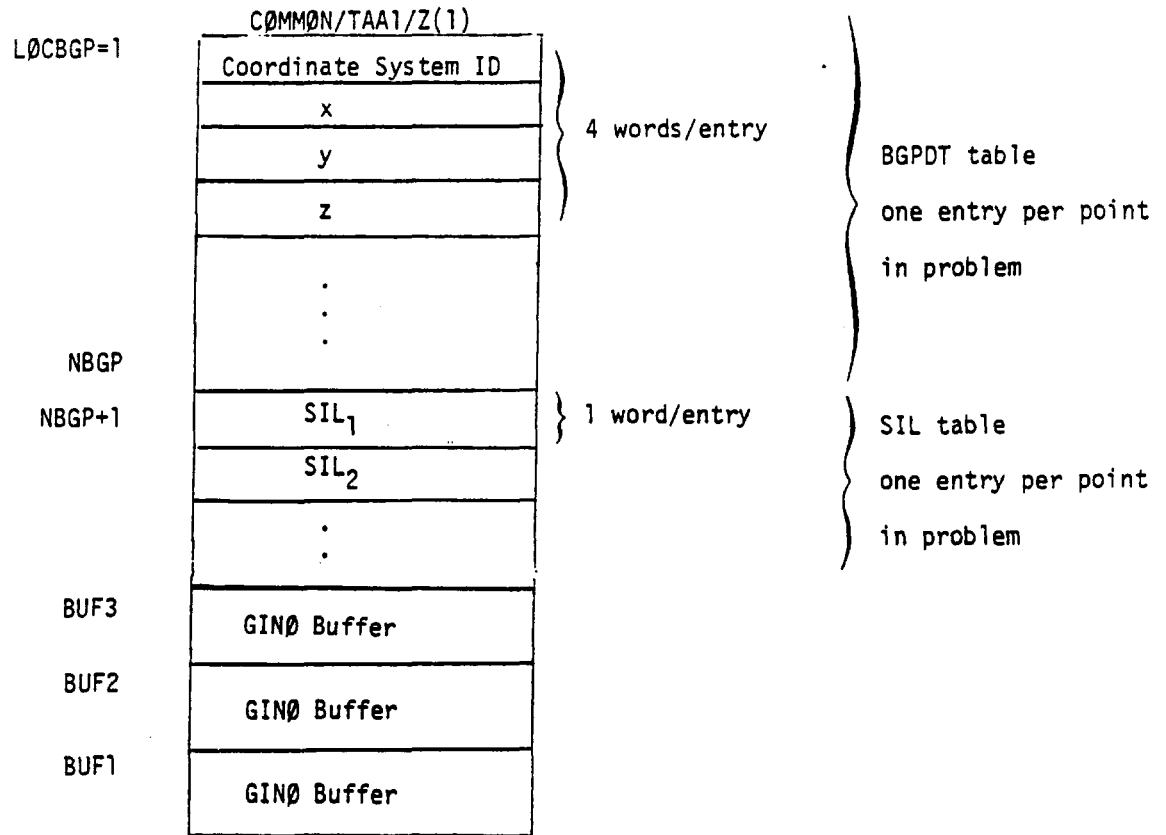
4.26.7.2 TA1A

Assembly of the Element Summary Table is performed in two steps. For the first step, the EPT is read into core one property type at a time. The ECT is read one element at a time. For each element the referenced property data are found by performing a binary search in the EPT in core. The ECT and EPT data are written on SCR1, a scratch file, one element at a time, one logical record per element type.

To initiate the second step, the BGPDT and SIL data blocks are read into core. If a temperature set is selected, the appropriate temperature data from the GPTT are read into core. Data from SCR1 are read one element at a time. Internal indices for the grid points are used as pointers into the BGPDT and SIL tables. The temperature of the element is extracted from the GPTT data block. Each temperature is found by performing a binary search in the GPTT with the element identification number. If the entry is not found in the GPTT, the default temperature for the set is substituted. The internal indices are now replaced with corresponding scalar index values. A line comprising ECT, EPT, BGPDT and GPTT data for the element is written on the EST. Each logical record of the EST comprises all data of one element type.

MODULE FUNCTIONAL DESCRIPTIONS

Allocation of core storage during the second step is as follows:



4.26.7.3 TA1B

The ECPT data block is assembled in TA1B. Each logical record in the ECPT corresponds to a grid or scalar point in the model. For each point, the data for each element connected to the point are listed. The data for each element are identical to the EST data. Each set of element data will be listed in the ECPT "n" times, where "n" is the number of grid points connected by the element. A sample of the ECPT is given in Table 1. The logical phases of the operation are as follows:

A list is formed in core giving the relative locations of the elements in the ECT data in the order which they will be placed in the ECPT data block. Storage is allocated by

FUNCTIONAL MODULE TA1 (TABLE ASSEMBLER)

forming the GPC (Grid Point Counter) and then replacing the GPC by a running sum of elements connected to points. A sample is:

Implied Grid Point internal index	GPC Number of Connected Elements	GPCS Sum of Previous Elements
(1)	5	0
(2)	3	5
(3)	1	8
(4)	2	9
(5)	6	11
.	.	17
.	.	.
.	.	.
.	.	.

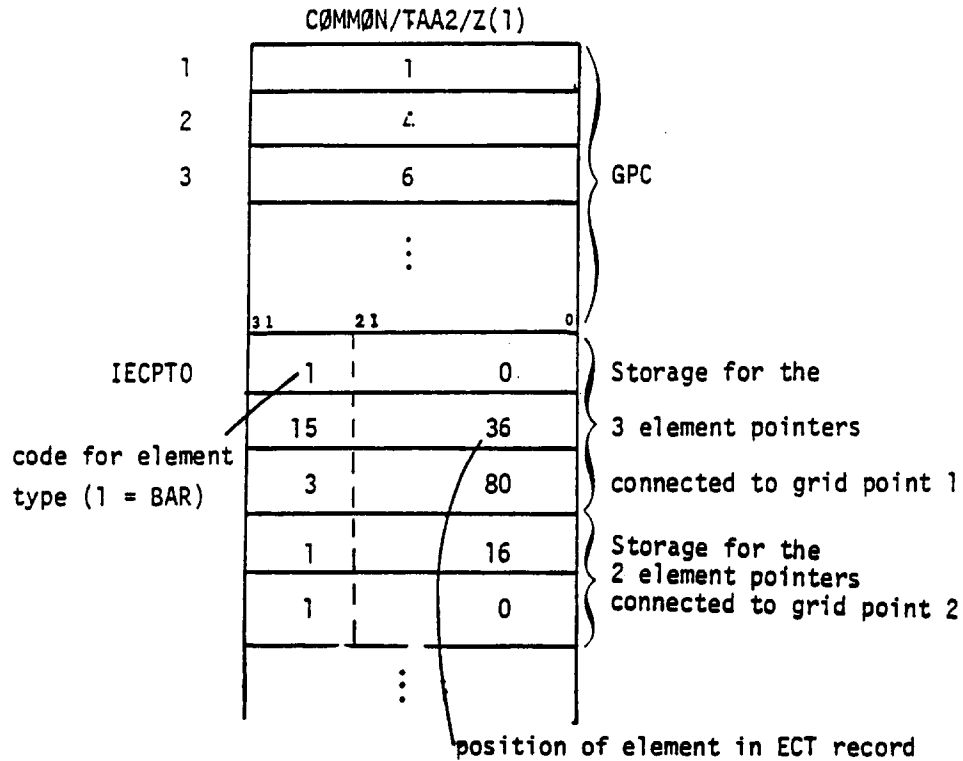
The GPC is formed as follows. An area of core equal to the number of grid and scalar points of the model is set to zero. The ECT is read one element at a time. The storage location corresponding to the internal index of each referenced grid point is incremented by one. When each element in the ECT has been processed, a running sum of the core table is formed.

The contents of each word in the GPC now provide a pointer to the first storage location where a second pointer to the element data will be stored. Since the total number of connected elements may exceed available core storage, spill logic is provided. A band of entries in the GPC is determined. The ECT is read one element at a time. The position of each element of a given element type is determined by summing the number of words for each entry for the element (i.e. if m = number of words per ECT entry, then the position of the element in the ECT record = $(i-1)*m$ where i = entry number in the ECT record). For each point referenced by the element (which is in the band currently being processed), the contents of the associated position in the GPC is fetched. The element position and its type are stored at the indicated locations.

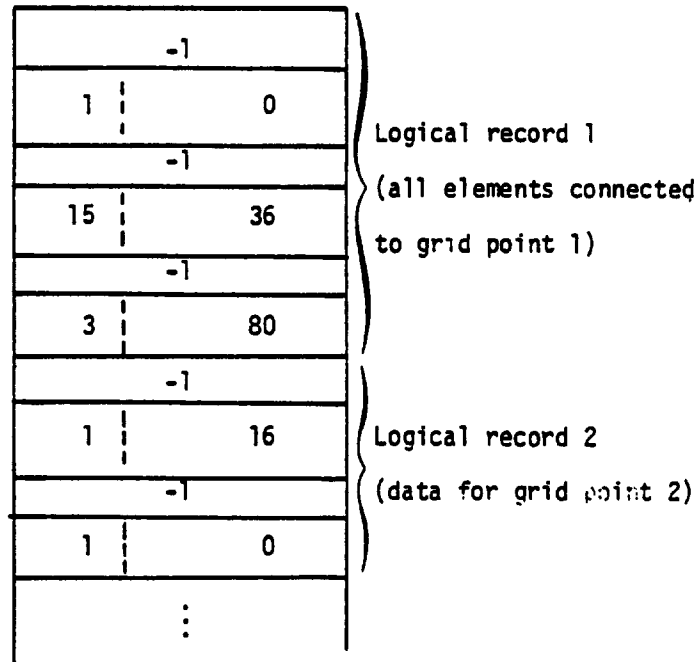
The location in the GPC is incremented by one. When a pass of the ECT is complete, the skeleton ECPT is written, one logical record per point in the band of the current pass. Each

MODULE FUNCTIONAL DESCRIPTIONS

logical record consists of pairs of (-1, element pointer). The number of pairs equals the number of elements connected to the point. Example:

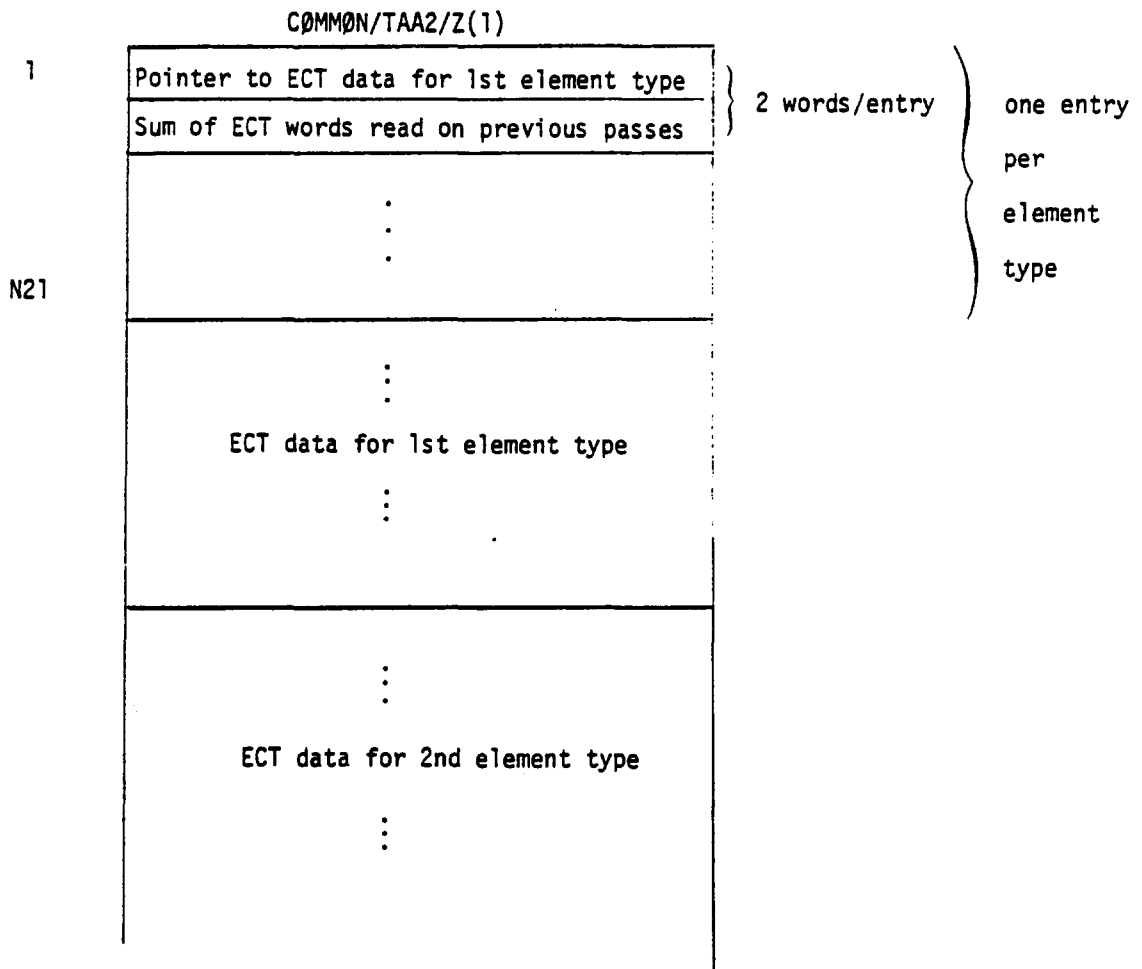


Skeleton ECPT data is written on a scratch file:



FUNCTIONAL MODULE TA1 (TABLE ASSEMBLER)

When all entries of the GPC have been processed, the skeleton ECPT is complete. An area of core equal to two words per element type is set to zero. The ECT is read into core following the above table until either the ECT is exhausted or core is filled. A pointer to the beginning of data for each element type is stored in the table. The skeleton ECPT is now read one pair of words at a time. If the data to which the element pointer points are currently in core, they are written out on a second scratch file. The first word of the entry contains the number of words in the ECT data. If the data are not in core, the pair (-1, element pointer) is written on the second scratch file. This process is repeated until the skeleton has been exhausted. If all the ECT is in core, the new skeleton ECPT is complete. Otherwise, the files for old and new skeleton ECPT are switched, and the process continues by reading more of the ECT data into core. Storage allocation at this point follows:



MODULE FUNCTIONAL DESCRIPTIONS

The remainder of the ECPT assembly is very similar to the EST construction. The principal difference is that the entire EPT is held in core. A pointer table similar to the ECT table is formed (two entries per element type). The EPT is read into core, and the first position of each EPT type is stored in the table along with the number of EPT entries of that type. If the BGPDT and SIL can be held in core with the EPT, the ECPT is assembled in one pass. Otherwise, two passes are made. The skeleton ECPT is read one entry at a time. For each element, the property data are attached by performing a binary search in the associated property table in core. If one pass, the BGPDT, SIL and GPTT data are attached as in TA1A. Otherwise the ECT and EPT data are written on a scratch file, and a second pass is made to attach the BGPDT, SIL and GPTT data. Table 1 contains sample ECPT contents.

During the final pass of the ECPT assembly, the GPCT is constructed. The GPCT is comprised of one logical record per point (same as ECPT). Each logical record consists of the pivot point and all other points connected to the pivot by means of element connections.

FUNCTIONAL MODULE TA1 (TABLE ASSEMBLER)

Table 1. Sample ECPT Data Contents.

Referenced Grid Point First Scalar Index (Pivot Point)	235
1st Element Type	9 (Triangular membrane)
Connected Grid Point First Scalar Indices (From ECT and SIL)	{ 625 235 535
Anisotropic Angle (From ECT)	0.0
Material Number (From EPT)	2
1st Element Properties (From EPT)	0.5 0.0 1.0
Location and Orientation Data for Grid Points (From BGPDT)	{ 5, 10.0, 100.0, 0.0 5, 11.0, 100.0, 0.0 0, 235.0, 50.0, 25.0
Element Temperature (From GPTT)	15.5
2nd Element Type	34 (Bar)
	. etc. .
Last Element Type (for point 235)	1 (Rod)
	. etc. .
End of Logical Record (new grid point)	*****
Reference Grid Point First Scalar Index (Pivot Point)	241
	. etc. .

MODULE FUNCTIONAL DESCRIPTIONS

4.26.7.4 TAIH

The GPECT data block is assembled in TAIH. Each logical record in the GPECT corresponds to a grid or scalar point in the model. For each point, the connection data for each element connected to the point are listed. Each set of element data will be listed in the GPECT "n" times, where "n" is the number of grid points connected by the element. A sample of the GPECT is given in Table 1. The logical phases of the operation are as follows:

A list is formed in core giving the relative locations of the elements in the ECT data in the order which they will be placed in the GPECT data block. Storage is allocated by forming the GPC (Grid Point Counter) and then replacing the GPC by a running sum of elements connected to points. A sample is:

Implied Grid Point internal index	GPC Number of Connected Elements	GPCS Sum of Previous Elements
(1)	5	0
(2)	3	5
(3)	1	8
(4)	2	9
(5)	6	11
.	.	17
.	.	.
.	.	.
.	.	.

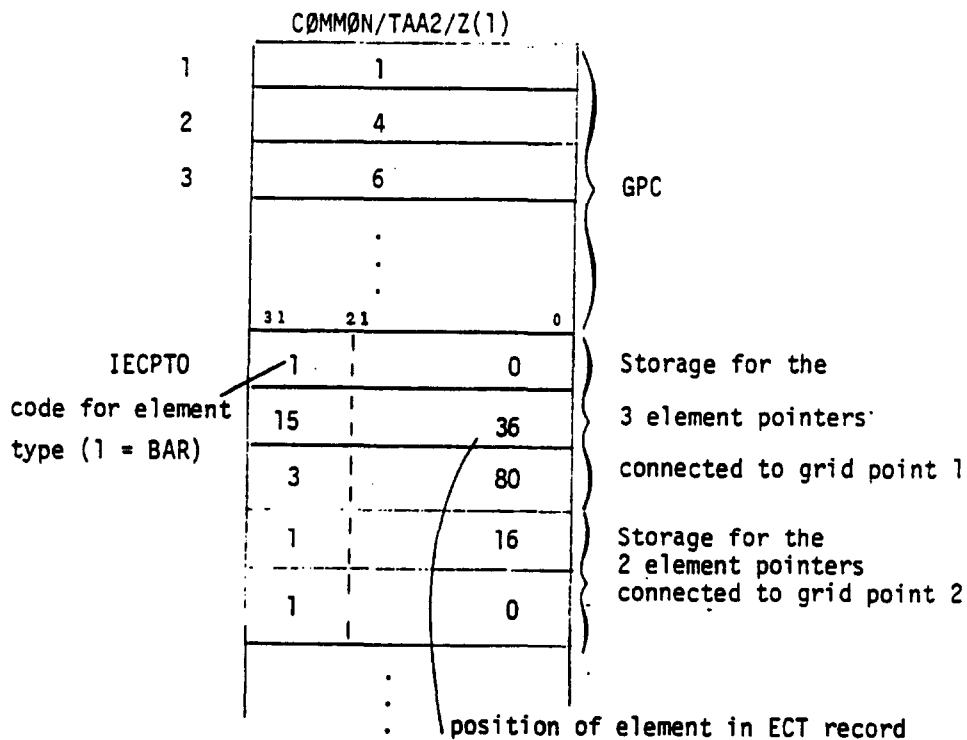
The GPC is formed as follows. An area of core equal to the number of grid and scalar points of the model is set to zero. The ECT is read one element at a time. The storage location corresponding to the internal index of each referenced grid point is incremented by one. When each element in the ECT has been processed, a running sum of the core table is formed.

The contents of each word in the GPC now provide a pointer to the first storage location where a second pointer to the element data will be stored. Since the total number of connected elements may exceed available core storage, spill logic is provided. A band of entries in the GPC is determined. The ECT is read one element at a time. The position of each element of a given element type is determined by summing the number of words for each entry for the element (i.e., if m = number of words per ECT entry, then the position of the element in the ECT record =

FUNCTIONAL MODULE TAA1 (TABLE ASSEMBLER)

$(i-1)*m$ where i = entry number in the ECT record). For each point referenced by the element (which is in the band currently being processed), the contents of the associated position in the GPC is fetched. The element position and its type are stored at the indicated locations.

The location in the GPC is incremented by one. When a pass of the ECT is complete, the skeleton ECPT is written, one logical record per point in the band of the current pass. Each logical record consists of pairs of $(-1, \text{element pointer})$. The number of pairs equals the number of elements connected to the point. Example:



MODULE FUNCTIONAL DESCRIPTIONS

Skeleton ECPT data is written on a scratch file:

-1		
1	:	0
-1		
15	:	36
-1		
3	:	80
-1		
1	:	16
-1		
1	:	0
		⋮

}

Logical record 1

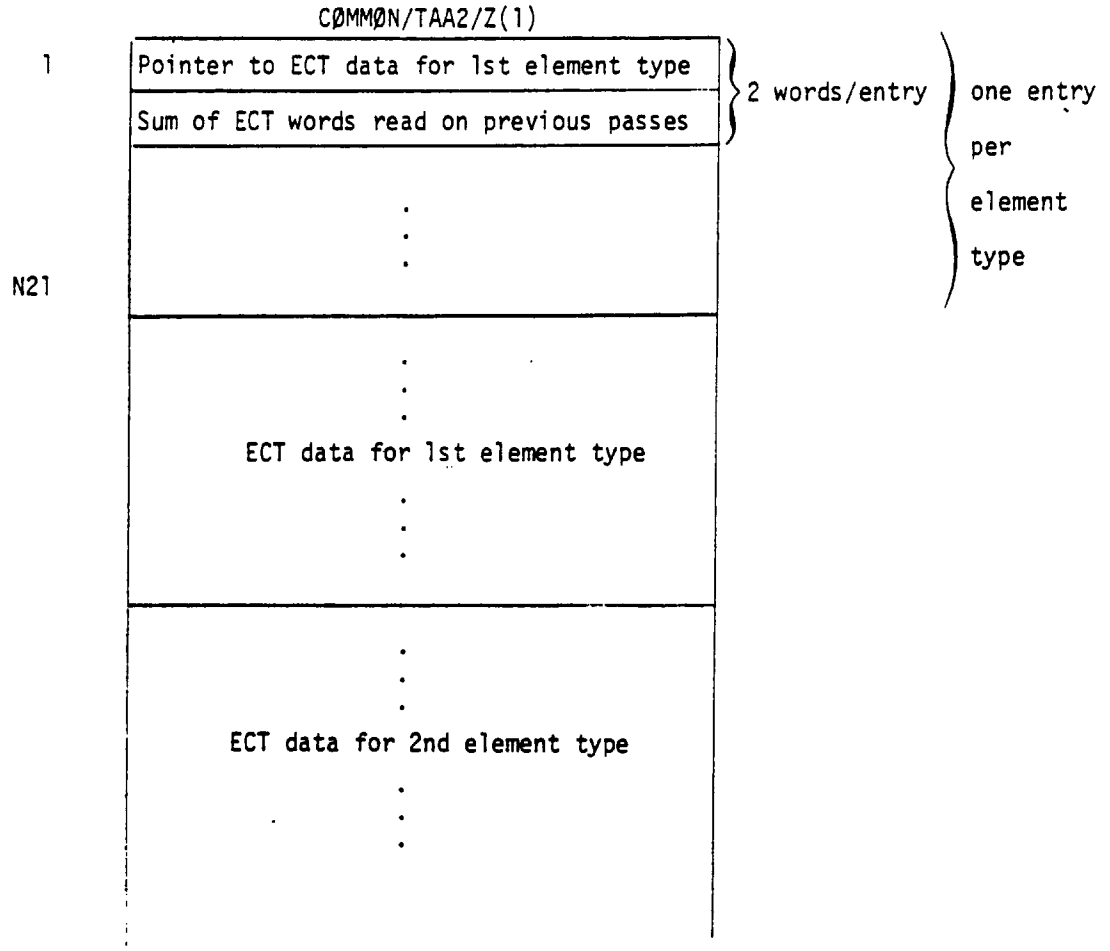
(all elements connected to grid point 1)

Logical record 2

(data for grid point 2)

When all entries of the GPC have been processed, the skeleton GPECT is complete. An area of core equal to two words per element type is set to zero. The ECT is read into core following the above table until either the ECT is exhausted or core is filled. A pointer to the beginning of data for each element type is stored in the table. The skeleton GPECT is now read one pair at a time. If the data to which the element pointer points are currently in core, they are written out on a second scratch file. The first word of the entry contains the number of words in the ECT data. If the data are not in core, the pair (-1, element pointer) is written on the second scratch file. This process is repeated until the skeleton has been exhausted. If all the ECT is in core, the new skeleton GPECT is complete. Otherwise, the files for old and new skeleton GPECT are switched, and the process continues by reading more of the ECT data into core. Storage allocation at this point follows:

FUNCTIONAL MODULE TAA1 (TABLE ASSEMBLER)



The final step in the GPECT assembly is to read the scratch file and convert internal grid numbers to SIL values.

MODULE FUNCTIONAL DESCRIPTIONS

Table 2. Sample GPECT Data Contents.

Reference Grid Point First Scalar Index (Pivot Point)	235
1st Element Type	9 (Triangular membrane)
Connected Grid Point First Scalar Indices (From ECT and SIL)	{ 625 235 535
2nd Element Type	34 (Bar)
	. etc. .
Last Element Type (for point 235)	1 (Rod)
	. etc. .
End of Logical Record (new grid point)	*****
Reference Grid Point First Scalar Index (Pivot Point)	241
	. etc. .

4.26.7.5 TA1C

The General Element flexibility and support matrices are assembled in TA1C. The data, given in the ECT data block, consist of the following sections for each General Element:

1. A list of the independent degrees of freedom, u_i , in terms of grid points and components and/or scalar points.
2. A list of supporting degrees of freedom, u_d , given by grid and scalar points (may be null).
3. An indicator of [K] or [Z] matrix.
4. A stiffness matrix [K] or a flexibility matrix [Z] with rows and columns corresponding to the list of given u_i points.
5. A support matrix [S] with rows corresponding to the u_i points and columns corresponding to the u_d points. (May be null.)

The tasks of TA1C are to (1) convert the lists of u_i and u_d to scalar indices and sort them by increasing scalar index, (2) rearrange the matrices to correspond to the sorted lists of u_i and u_d degrees of freedom, and (3) on user option calculate the support matrix from grid point geometry.

These tasks are accomplished as follows:

1. For each set of coordinates (u_i and u_d) of length n , a $4 \times n$ table is formed where the four entries corresponding to each degree of freedom are:
 - a. The position as given (1, 2, 3 ... n)
 - b. The internal position (zero initially)
 - c. The grid or scalar point I.D. (Internal index)
 - d. The grid point component (1 = x, 2 = y, etc.)
2. The list is sorted on the third and fourth position. If a point I.D. in the third position is duplicated, the duplicates are sorted on the components in the fourth position. The list now corresponds to the desired order of increasing scalar indices.
3. The SIL data block is read, and each of the points in the list is converted to its SIL value. (The position of a SIL number is its internal point index.) The SIL value and the component c_i determine the scalar index, N_i , of a degree of freedom by:

$$N_i = SIL + (c_i - 1) \quad (1)$$

for grid points and

$$N_i = SIL \quad (2)$$

for scalar points.

The list of scalar indices is written on the GEI data block file.

4. In order to rearrange the matrices to correspond to the proper sequence of degrees of freedom the above list is modified as follows:

- a) The internal position number is placed in the second position of each entry in the list. The first entry uses 1, the second, 2, etc.
- b) The list is sorted again to return the original order as given on the input card images. The first position of each entry supplies this order. The internal position of a term in a matrix is now given by the second numbers in each entry.

5. Steps (1) through (4) are repeated for both the u_i and u_d sets.

6. The $[K]$ or $[Z]$ and $[S]$ matrices are rearranged according to the sorted lists of degrees of freedom. The row and column numbers are converted by the algorithm:

- a) For a term of K_{ij} or Z_{ij} , where i and j are the row and column as given by the matrix order, the position i of the u_i internal number list gives ℓ , the new column number.
- b) For a term of S_{ij} , where i and j are the external row and column numbers, the row number i is converted using the u_i list. The column number j uses the u_d list.

If a matrix is small enough to fit in core, the new row and column numbers are used to place the term in its correct position in core. If the matrix will be larger than core, the new row and column indices and the term itself are written on a scratch file. When all terms are processed, the file is sorted to form a sequenced matrix. The terms of the matrix are written on the GEI file in full matrix form.

4.26.7.6 TA1CA

The [S] matrix must be generated if the user inputs a list of U_d points and does not supply an [S] matrix. If the flexibility matrix [Z] was input and [S] is to be internally calculated there must be six U_d points in the list. If the stiffness matrix [K] was input and [S] was not supplied, the number of U_d points must be less than or equal to six. The generation of [S] is accomplished in subroutine TA1CA as follows:

The BGPDT and CSTM data are read into core. (SIL is already in core).

A six by six matrix $[D_0]$ is formed, where each row corresponds to a u_d scalar index (j). $[D_0]$ is a six by six matrix which transforms the three translations and three rotations in the basic coordinate system to the six rigid body u_d degrees of freedom:

$$\{u_d\} = [D_0] \begin{Bmatrix} x_0 \\ y_0 \\ z_0 \\ \theta_{x0} \\ \theta_{y0} \\ \theta_{z0} \end{Bmatrix}_{\text{basic}} \quad (3)$$

The steps for generation of each row of [D] are as follows:

1. X_j, Y_j, Z_j the BGPDT location vector for the grid point containing scalar point j is found.
2. $[T_j]$ the 3x3 global-to-basic transformation matrix for the grid point containing j is fetched using subroutine MAT.
3. If scalar j is a translation, define:

$$[E_j] = \begin{bmatrix} 1 & 0 & 0 & 0 & Z_j & -Y_j \\ 0 & 1 & 0 & -Z_j & 0 & X_j \\ 0 & 0 & 1 & Y_j & -X_j & 0 \end{bmatrix} \quad (4)$$

The column of $[T_j]$ corresponding to the degree of freedom j is defined as the row vector $\{V_j\}^T$. The row vector of $[D_0]$ corresponding to point j is:

$$\{D_{0j}\}^T = \{V_j\}^T [E_j]. \quad (5)$$

4. If scalar j is a rotation, the column of $[T_j]$ corresponding to the degree of freedom j is defined as the row vector $\{V_j\}^T$. The row vector of $[D_0]$ corresponding to point j is:

$$\{D_{0j}\}^T = \{0 \ 0 \ 0: V_j^T\}. \quad (6)$$

When all rows of $[D_0]$ have been generated, the matrix is inverted. However, if the stiffness matrix was input, $[D_0]$ may be singular. If R is the number of U_d terms in the list, then the r by r nonsingular submatrix $[a]$ of $[D_0]$ is extracted using sub-routine DMFGR. The rows of $[a]$ are kept in their original order but the columns may have been rearranged in the extraction process. If a redundant set of rigid body modes was specified, a fatal error exists. The matrix $[H_{do}]$ is defined as:

$$\begin{aligned} [H_{do}] &= [D_0]^{-1} \\ \text{or} \quad [H_{do}] &= [a]^{-1} \end{aligned} \quad (7)$$

$[H_{do}]$ transforms the U_d displacements to rigid body motions about the basic coordinate system, i.e.:

$$\{U\}_{\text{basic}} = [H_{do}] \{U_d\} \quad (8)$$

If the matrix is ill-conditioned, a fatal error exists.

The $[S]$ matrix may now be calculated a row at a time. The list of u_i points, (s_i) , is read one at a time, and the $[S]$ matrix is formed a row at a time. Call each row $\{S_i\}^T$.

The steps for generation of each row are as follows:

1. Using the basic coordinates X_i, Y_i, Z_i for the grid point corresponding to scalar s_i , the global-to-basic transformation matrix $[T_i]$ is fetched.
2. The column of $[T_i]$ corresponding to the scalar coordinate of u_i is defined as the row vector $\{V_i\}^T$.

3. If u_i is a translation, we form

$$[E] = \begin{bmatrix} 1 & 0 & 0 & 0 & Z_i & -Y_i \\ 0 & 1 & 0 & -Z_i & 0 & X_i \\ 0 & 0 & 1 & Y_i & -X_i & 0 \end{bmatrix}, \quad (9)$$

If the stiffness matrix was input and less than six U_d terms were specified, the matrix $\{V_i\}^T [E]$ is changed so that only the columns corresponding to the columns of $[a]$ are retained and these columns are in the same order as the corresponding columns of $[a]$. We then form

$$\{S_i\}^T = \{V_i\}^T [E] [H_{od}] \quad (10)$$

4. If u_i is a rotation:

$$\{S_i\}^T = \{0 \ 0 \ 0: V_j^T\} [H_{od}] \quad (11)$$

However, if the stiffness matrix was input and less than six U_d terms were specified, the matrix $\{000: V_j^T\}$ is modified so that only the columns corresponding to the columns of $[a]$ are retained and these columns are in the same order as the corresponding columns of $[a]$.

4.26.8 Subroutines

4.26.8.1 Subroutine Name: TA1

1. Entry Point: TA1
2. Purpose: Module driver.
3. Calling Sequence: CALL TA1

4.26.8.2 Subroutine Name: TA1A

1. Entry Point: TA1A
2. Purpose: To assemble the EST
3. Calling Sequence: CALL TA1A

4.26.8.3 Subroutine Name: TA1B

1. Entry Point: TA1B
2. Purpose: To assemble the ECPT and GPCT.
3. Calling Sequence: CALL TA1B

4.26.8.4 Subroutine Name: TA1H

1. Entry Point: TA1H
2. Purpose: To assemble the GPECT
3. Calling Sequence: CALL TA1H

4.26.8.5 Subroutine Name: TA1C

1. Entry Point: TA1C
2. Purpose: To assemble the GEI.
3. Calling Sequence: CALL TA1C

4.26.8.6 Subroutine Name: TA1CA

1. Entry Point: TA1CA
2. Purpose: To calculate the general element support matrix [S].
3. Calling Sequence: CALL TA1CA

4.26.8.7 Subroutine Name: DMFGR

1. Entry Point: DMFGR
2. Purpose: To calculate the rank of a matrix.
3. Calling Sequence: CALL DMFGR (IG,JR,JD,IR,ID)

4.26.8.8 Subroutine Name: TA1ETD

1. Entry Point: TA1ETD
2. Purpose: To extract the temperature data for an element from the GPTT data block.

3. Calling Sequence: CALL TA1ETD (ELID, TI, GRIDS)

ELID - ID of the element for which temperature data is desired -integer-input

TI - Array of temperature data - real-output

GRIDS - Number of grid points of the element. If GRIDS=0, average element temperature is returned in TI(1), integer-input

4.26.9 Design Requirements

4.26.9.1 Allocation of Core Storage

TA1A. Step (1): Maximum core storage equals all property data for one element type plus three GINØ buffers.

Step (2): Maximum core storage equals 5* (number of grid and scalar points in model) plus three GINØ buffers.

TA1B. The initial steps of TA1B are open ended. The final assembly of the ECPT requires that all EPT data be held in core at one time. At another time the same storage requirements as TA1A exists plus additional storage to hold a list of the maximum number of points connected to any one point by means of element connections.

TA1H. The initial steps of TA1H are open ended. The same storage requirement as TA1A exists plus additional storage to hold a list of the maximum number of points connected to any one point by means of element connections.

TA1C. The maximum storage requirement equals 5* (number of grid and scalar points in the model) plus the CSTM table plus 4* (number of u_1 + number of u_d points) plus three GINØ buffers.

4.26.9.2 Environment

TA1 is designed to allow each of the major phases of the model to be in a separate overlay segment. Open core for each is defined as follows:

TA1A: /TAA1/

TA1B: /TAA2/

TA1H: /TAA2/

TA1C: /TAC1/

MODULE FUNCTIONAL DESCRIPTIONS

GINØ file names and DMAP parameters are communicated through blank CØMMØN. No block data program is used. Communication between TA1C and TA1CA occurs through /TA1CAX/ and /TAC1/.

4.26.10 Diagnostic Messages

The following messages may be issued by TA1:

2010, 2011, 2013, 2014, 2015, 2044, 2045, 2063, 2082, 3057, 3058, and 4016.

FUNCTIONAL MODULE TA1 (TABLE ASSEMBLER)

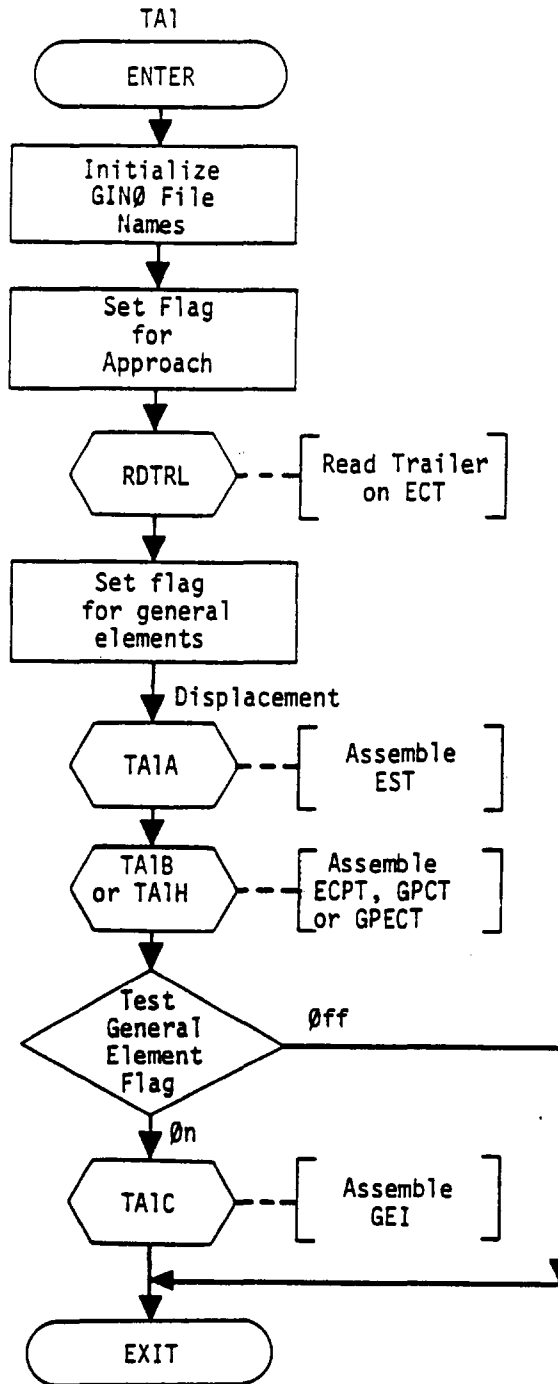


Figure 1. Flowchart for module TA1

MODULE FUNCTIONAL DESCRIPTIONS

THE MATERIAL PREVIOUSLY ON THIS PAGE
HAS BEEN DELETED

FUNCTIONAL MODULE SMA1 (STRUCTURAL MATRIX ASSEMBLER - PHASE 1)

4.27 FUNCTIONAL MODULE SMA1 (STRUCTURAL MATRIX ASSEMBLER - PHASE 1)

4.27.1 Entry Point: SMA1

4.27.2 Purpose

To generate the stiffness matrix exclusive of general elements, $[K_{gg}^x]$, the structural damping matrix, $[K_{gg}^4]$, and the Grid Point Singularity Table, GPST.

4.27.3 DMAP Calling Sequence

SMA1 CSTM,MPT,ECPT,GPCT,DIT/KGGX,K4GG,GPST/V,N,NØGENL/V,N,NØK4GG \$

4.27.4 Input Data Blocks

CSTM - Coordinate System Transformation Matrices.
MPT - Material Properties Table.
ECPT - Element Connection and Properties Table.
GPCT - Grid Point Connection Table.
DIT - Direct Input Tables.

Notes: 1. The CSTM may be purged.
2. The ECPT and the GPCT cannot be purged, or else a fatal error will occur.
3. If some element references a material property, the MPT cannot be purged.
4. If some material property is temperature dependent, DIT cannot be purged.

4.27.5 Output Data Blocks

KGGX - Partition of stiffness matrix exclusive of general elements - g set.
K4GG - Partition of structural damping matrix - g set.
GPST - Grid Point Singularity Table.

Notes: 1. Neither KGGX or GPST may be pre-purged.
2. If K4GG is pre-purged, K4GG will not be generated.
3. If NØGENL > 0 (see below) the GPST will not be generated.

4.27.6 Parameters

NØGENL - Input-integer-no default value. NØGENL is the number of general

elements in the model. If $NØGENL > 0$ then GPST will not be generated.

$NØK4GG$ - Output-integer-no default value. If $K4GG$ has been pre-purged or is the zero matrix, $NØK4GG$ is set equal to -1. Otherwise $NØK4GG$ is set = +1.

4.27.7 Method

Matrix generation modules such as SMA1, SMA2, DSMG1 and PLA4 all use the ECPT (or a variation thereof in the case of PLA4) and its companion data block, the GPCT, as the basic data blocks for generation of stiffness and structural damping matrices (SMA1), mass and viscous damping matrices (SMA2), the differential stiffness matrix (DSMG1), and the non-linear stiffness matrix (PLA4). The central role of the ECPT data block in these modules is discussed in section 1.8.

Subroutine SMA1 is the module driver. Its tasks are: to set up GINØ buffers and matrix control blocks for the output matrices; to determine if the CSTM data block exists, and, if it does, to read it into open core and call the initialization routine PRETRD; to call the material properties initialization routine PREMAT, where material property cards and tables are read into open core; to open and position all files so that input data blocks are ready to be read and output data blocks are ready to be written; to call subroutine SMA1A, the module "workhorse", which will create the output data blocks. Upon return from SMA1A, files are closed and trailers are written. Subroutine descriptions for PRETRD and PREMAT can be found in section 3.4.37 and 3.4.36 respectively.

Subroutine SMA1A consists entirely of a loop in which, during each pass of the loop, a record of the GPCT and a record of the ECPT are processed in a complementary manner. Each pass through this principal loop creates either one or six rows (or columns since $[K_{gg}^x]$ and $[K_{gg}^4]$ are symmetric) of the stiffness matrix, KGGX, (and the structural damping matrix, K4GG, if called for in the DMAP calling sequence). One row will be generated if the pivot point, the first word of both the GPCT record and the ECPT record, is a scalar point; six rows will be generated if the pivot point is a grid point. The latter case holds in the majority of cases. The loop is terminated when an end-of-file is sensed on the file containing the GPCT.

The loop begins by attempting to read the first two words of the current GPCT record. If the second non-standard return from subroutine READ occurs, it implies the current pivot point has no elements connected to it so that one or six null rows must be output for the

FUNCTIONAL MODULE SMA1 (STRUCTURAL MATRIX ASSEMBLER - PHASE 1)

matrices. This non-standard return, it should be noted, does not occur in the majority of cases. A normal return from READ implies a normal path through the principal loop. The remainder of the GPCT record is read into core, and a pointer table is constructed. This pointer table relates the scalar index numbers of the GPCT record, which are the column indices of the (possible) non-zero terms of the generated matrices, to locations in core in which the corresponding submatrices will reside. The pointer table is defined recursively as follows:

$$p_1 = 1 \quad (1)$$

$$p_i = p_{i-1} + q, i > 1 \quad (2)$$

where q is 6 if the point $(i-1)$ is a grid point and q is 1 if the point $(i-1)$ is a scalar point.

It is then determined if all the submatrices corresponding to the current pivot point can be held in open core. If they can, there is no problem. If they cannot, spill logic is provided only if the structural damping matrix is not called for. Rather than compute all submatrices, store them on a scratch file, retrieve them when computations have been completed and sort them, the following approach was adopted. It is determined what the maximum number of rows that can be held in core is: 3, 2 or 1. The element submatrices are computed in their respective routines (i.e. KR0D, KBAR) and passed to the "insertion" subroutine, SMA1B, which "inserts" into the correct open core positions only those rows which can be contained.

When the ECPT record is exhausted, the link vector, LINK, (see discussion below of the LINK variable in /SMA1CL/) is searched to determine if any structural element entries in the ECPT were skipped because the corresponding element subroutine did not reside in the link (element subroutine overlay segment) currently in core. If there did exist such an element, the ECPT file is backspaced and the ECPT record is reprocessed such that all element submatrices not computed on the previous pass(es) of the ECPT record will be computed. If (or when) the LINK array is identically zero, signifying that all element submatrices corresponding to the current pass of the ECPT record have been computed, the DETCK subroutine is called if the input parameter N0GENL ≤ 0 . DETCK generates the GPST by examining the "translational" and "rotational" diagonal 3 X 3 submatrices of the 6 rows of the KGGX matrix currently in core. If the total number of rows to be computed (6 or 1) for the current ECPT record is not in core due to spill problems, DETCK stores those elements of the 3 X 3 sub-

MODULE FUNCTIONAL DESCRIPTIONS

matrices currently in core in local variables and then processes the entire 3x3 submatrices on the last pass of the ECPT record (see definitions of the variables LRØWIC and NRØWSC in /SMA1CL/ below).

After DETCK returns, the number of rows in core are packed onto the KGGX (and, if called for, the K4GG) data block(s) using the standard matrix packing routines BLDPK, ZBLPKI and BLDPKN. If the last row in core is not equal to the total number of rows to be computed, the ECPT file is backspaced and the record is processed again, this time the next set of 3, 2 or 1 rows being output. If the last row in core is equal to the total number of rows to be computed, the processing of the ECPT record is complete and a transfer is made to the top of the "GPCT and ECPT processing" loop to process the next record of the GPCT and ECPT. The loop terminates when an end-of-file is encountered while attempting to read the GPCT. Upon loop termination, SMA1A returns to SMA1.

It should be noted that the most difficult logic of the routine involves the LINK vector and the spill logic. The programmer is advised that the LINK vector logic will not be used on any of the current hardware/software configurations because 1) the routine residing in segment (link) 2, KØØNE, cannot be used in conjunction with any other structural element routine and 2) the axisymmetrical element routines KTRIRG, KTRAPR and KTØRDR cannot (from a mathematical modeling point of view) be used in conjunction with any other structural element routines. The spill logic is very seldom entered since for the majority of cases the geometry of the mathematical model is such that the number of words in any GPCT record - and hence the number of (potentially) non-zero columns in any six rows of the matrix - is generally quite small. A high upper limit for the number of words in any GPCT record would be 40.

4.27.7.1 Determining Grid Point Singularities in Subroutine DETCK

Let the pivot point be a grid point with scalar index p in the following discussion. Let $[Q]$ be the "translational" or "rotational" 3×3 symmetric submatrix along the diagonal of the stiffness matrix, $[K_{gg}^x]$, i.e., the rows and columns of the "translational" $[Q]$ matrix would correspond to scalar index numbers p , $p+1$, and $p+2$; and the rows and columns of the "rotational" $[Q]$ matrix would correspond to scalar index numbers $p+3$, $p+4$, and $p+5$.

The following steps comprise the algorithm for determining the presence or absence of grid point singularities. The discussion assumes $[Q]$ is the "translational" 3×3 matrix but the same algorithm holds for the "rotational" $[Q]$. In this development, a general non-symmetric matrix could be considered. However, only diagonal terms and diagonal 2-by-2 minors are considered in determining the order of the matrix since all present NASTRAN elements theoretically generate symmetric matrices. Thus, in the actual code, the off-diagonal terms are forced to be symmetric. Any non-symmetry due to numerical round-off is corrected by replacing corresponding off-diagonal pairs by their average.

1. The matrix $[Q]$ is scaled by the magnitude of the largest term, Q_{\max} :

$$[B] = \frac{[Q]}{Q_{\max}}. \quad (3)$$

If the largest term is non-positive, the singularity is of order 3, and the scalar index numbers p , $p+1$ and $p+2$ are written on the GPST.

2. The vector magnitudes of 3×1 columns (rows) are calculated:

$$b_1 = \sqrt{B_{11}^2 + B_{12}^2 + B_{13}^2}, \quad (4)$$

$$b_2 = \sqrt{B_{21}^2 + B_{22}^2 + B_{23}^2}, \quad (5)$$

$$b_3 = \sqrt{B_{31}^2 + B_{32}^2 + B_{33}^2}. \quad (6)$$

3. For each $b_i = 0$, the singularity order counter $IORDER$ is increased by one.
4. If two b_i are zero, the order of the singularity is two, and the scalar index numbers j and k corresponding to these two rows of $[B]$ are written on the GPST.

MODULE FUNCTIONAL DESCRIPTIONS

5. If one b_i is zero, and i is the row such that $b_i = 0$, define j and k as the other rows of $[B]$ and calculate:

$$m = \det \begin{bmatrix} B_{jj} & B_{jk} \\ B_{kj} & B_{kk} \end{bmatrix}, \quad (7)$$

$$R = \sqrt{(B_{jj}^2 + B_{kj}^2)(B_{jk}^2 + B_{kk}^2)}. \quad (8)$$

If $\frac{m}{R} < TOL^*$, the order of the singularity is 2 and the GPST contains the paired scalar index values for i, j, k in the order (1) (i, j) if $B_{kk} > 0$ or (2) (i, k) if $B_{jj} > 0$.

If $\frac{m}{R} \geq TOL^*$, the order of the singularity is one and only the SIL value for i is written on the GPST.

6. If all $b_i > 0$, we calculate

$$D = \det [B] \quad (9)$$

If $D > .5 \times TOL^* \times (b_1 b_2 b_3)$, there are no singularities, and DETCK returns if $[Q]$ is the "rotational" matrix. If $[Q]$ is the "translational" matrix, the "rotational" $[Q]$ is input to the algorithm.

7. If $D < .5 \times TOL^* \times (b_1 b_2 b_3)$, one or more singularities exist. The following terms are calculated:

$$m_1 = \det \begin{bmatrix} B_{22} & B_{23} \\ B_{32} & B_{33} \end{bmatrix}, \quad (10)$$

$$m_2 = \det \begin{bmatrix} B_{11} & B_{13} \\ B_{31} & B_{33} \end{bmatrix}, \quad (11)$$

$$m_3 = \det \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}, \quad (12)$$

*The tolerance (TOL) may be supplied by the user on the NASTRAN card by specifying the desired value for SYSTEM(70) or STST. The default is .01.

$$R_1 = \sqrt{(B_{22}^2 + B_{23}^2)(B_{32}^2 + B_{33}^2)} , \quad (13)$$

$$R_2 = \sqrt{(B_{11}^2 + B_{13}^2)(B_{31}^2 + B_{33}^2)} , \quad (14)$$

$$R_3 = \sqrt{(B_{11}^2 + B_{12}^2)(B_{21}^2 + B_{22}^2)} . \quad (15)$$

8. Determine i, j, k such that:

$$\frac{m_i}{R_i} > \frac{m_j}{R_j} > \frac{m_k}{R_k} . \quad (16)$$

9. If $\frac{m_i}{R_i} < TOL$, the singularity is of order 2. Redefine i, j, k such that $B_{ii} < B_{jj} < B_{kk}$. The SIL values for the paired indexes (j,k), (i,k) and (i,j) are written on GPST only if the corresponding B is greater than zero. For instance if B_{kk} is zero, the SIL pair (i,j) is not written on the GPST.

10. If $\frac{m_i}{R_i} \geq TOL$, (see step 8) the singularity is of order 1. The SIL values are written on the GPST in the order

$$i \text{ since } \frac{m_i}{R_i} > TOL , \quad (17)$$

$$j \text{ if } \frac{m_j}{R_j} > TOL , \quad (18)$$

$$k \text{ if } \frac{m_k}{R_k} > TOL . \quad (19)$$

4.27.8 Subroutines

The utility routines PRETRD and PREMAT (or HMAT) are called in SMA1 for initialization purposes so that the structural element subroutines can call the entry points TRANSD of PRETRD and MAT of PREMAT (or HMAT) to fetch Coordinate System Transformation Matrices (CSTM) data and material properties data respectively. GMMATD is used by element routines as a general matrix multiply routine, and INVERD is used for inversion of small in-core (order usually ≤ 12) matrices. It should be noted that all matrices referenced in the structural element subroutines are stored by rows and are double precision. See the subroutine descriptions for these routines in Section 3.

The principal means of communicating an element entry of the ECPT to an element stiffness matrix generation routine is through /SMA1ET/. This fact is not explicitly stated in each of the descriptions of the element routines (e.g. KR0D) given below. Since much of the mathematics needed for generating: (1) element stiffness and structural damping matrices (module SMA1); (2) element mass and damping matrices (module SMA2); (3) element contributions to load vectors (module SSG1); (4) element stress (and force) data recovery (module SDR2); (5) element differential stiffness matrices (module DSMG1); (6) element stress (and force) data recovery for non-linear elements in a Piecewise Linear Analysis Rigid Format problem (module PLA3); and (7) element stiffness matrices for non-linear elements in a Piecewise Linear Analysis problem (module PLA4), is similar or even identical, detailed mathematical algorithms are grouped by element in Section 4.87.

It should be noted that routines DKI, DKINT, DK89, DK100, DK211, R0MBDK, and DMATRX are used only (directly or indirectly) by the axisymmetric shell element routines KTRIRG, KTRAPR and KT0RDR.

4.27.8.1 Subroutine Name: SMA1

1. Entry Point: SMA1
2. Purpose: See discussion above.
3. Calling Sequence: CALL SMA1

4.27.8.2 Subroutine Name: SMA1A

1. Entry Point: SMA1A

2. Purpose: See discussion above.

3. Calling Sequence: CALL SMA1A

4.27.8.3 Subroutine Name: SMA1B

1. Entry Point: SMA1B

2. Purpose: This routine, called by the module's element stiffness matrix generation routines such as KR0D, KBAR, etc., adds a double precision 6×6 or 1×1 matrix, $[K^e]$, to the "submatrix" of $[K_{gg}^x]$ or $[K_{gg}^4]$ corresponding to the current pivot point.

3. Calling Sequence: CALL SMA1B (KE,J,II,IFILE,DAMPC)

- KE - Row-stored double precision 6×6 or 1×1 matrix to be added to the submatrix in core - double precision - input.
- J - The column index of the $[K_{gg}^x]$ or $[K_{gg}^4]$ matrix which corresponds to the first column of the $[K^e]$ matrix - integer - input.
- II - If II is 0, the $[K^e]$ matrix is 6×6 . If II is greater than zero, it is the row index of the $[K_{gg}^x]$ or $[K_{gg}^4]$ matrix corresponding to the 1×1 matrix $[K^e]$ to be added - integer - input.
- IFILE - GIN0 file number of the matrix in core being added to - KGGX or K4GG - integer - input.
- DAMPC - If $[K^e]$ is 6×6 and the $[K_{gg}^4]$ matrix is called for, the input matrix $[K^e]$ is multiplied by DAMPC before being added to the submatrix of $[K_{gg}^4]$ in core.

4.27.8.4 Block Data Program Name: SMA1BD

1. Entry Point: SMA1BD

2. Purpose: Block data program which sets GIN0 file numbers, I/0 parameters, and SMA1 overlay parameters.

3. Calling Sequence: None

4.27.8.5 Subroutine Name: DETCK

1. Entry Point: DETCK

MODULE FUNCTIONAL DESCRIPTIONS

2. Purpose: This routine generates the Grid Point Singularity Table by examining the 3x3 "translational" and "rotational" diagonal submatrices of the KGGX matrix. This routine is called after the submatrix for each pivot point has been completed.
3. Calling Sequence: CALL DETCK (JARG)

$$\text{JARG} - \left\{ \begin{array}{l} \text{If JARG} = 0, \text{ the pivot point has elements connected to it.} \\ \text{If JARG} = -1, \text{ the pivot point is a scalar point and no elements are} \\ \text{connected to it.} \\ \text{If JARG} = 1, \text{ the pivot point is a grid point and no elements are connected} \\ \text{to it.} \end{array} \right.$$

4.27.8.6 Subroutine Name: KRØD

1. Entry Point: KRØD
2. Purpose: To generate the element stiffness matrix for a RØD element.
3. Calling Sequence: CALL KRØD

4.27.8.7 Subroutine Name: KBAR

1. Entry Point: KBAR
2. Purpose: To generate the element stiffness matrix for a BAR element.
3. Calling Sequence: CALL KBAR

4.27.8.8 Subroutine Name: KTUBE

1. Entry Point: KTUBE
2. Purpose: To generate the element stiffness matrix for a TUBE element.
3. Calling Sequence: CALL KTUBE

4.27.8.9 Subroutine Name: KPANEL

1. Entry Point: KPANEL
2. Purpose: To generate the element stiffness matrix for a SHEAR or TWIST panel element.
3. Calling Sequence: CALL KPANEL (IARG)

IARG - $\begin{cases} \text{IARG} = 4 \text{ calls for generation of the matrix for a shear panel;} \\ \text{IARG} = 5 \text{ implies a twist panel.} \end{cases}$

4.27.8.10 Subroutine Name: KTRMEM

1. Entry Point: KTRMEM
2. Purpose: To generate the element stiffness matrix for a TRMEM element.
3. Calling Sequence: CALL KTRMEM (I)

$I = \begin{cases} 0 - \text{Do complete triangular membrane.} \\ 1 - \text{Return 3 transformed } 3 \times 3 \text{ matrices only for pivot point. If } I = 1, \\ \text{KTRMEM is called by KQDMEM.} \end{cases}$

4.27.8.11 Subroutine Name: KQDMEM

1. Entry Point: KQDMEM
2. Purpose: To generate the element stiffness matrix for a QDMEM element.
3. Calling Sequence: CALL KQDMEM

4.27.8.12 Subroutine Name: KTRBSC

1. Entry Point: KTRBSC
2. Purpose: To generate the element stiffness matrix for a basic bending triangle element.
3. Calling Sequence: CALL KTRBSC (I)

$I = \begin{cases} 0 - \text{Do complete element computation for basic bending triangle} \\ 1 - \text{Form only the } [K^U] \text{ } 9 \times 9 \text{ matrix.} \\ 2 - \text{Form only the } [K^U] \text{ } 9 \times 9 \text{ matrix but save the } [H]^{-1} \text{ and } [S] \text{ matrices.} \end{cases}$

4.27.8.13 Subroutine Name: KTRPLT

1. Entry Point: KTRPLT
2. Purpose: To generate the element stiffness matrix for a triangular plate element.
3. Calling Sequence: CALL KTRPLT

4.27.8.14 Subroutine Name: KQDPLT

1. Entry Point: KQDPLT
2. Purpose: To generate the element stiffness matrix for a quadrilateral plate element.
3. Calling Sequence: CALL KQDPLT

4.27.8.15 Subroutine Name: KTRIQQ

1. Entry Point: KTRIQQ
2. Purpose: To generate the element stiffness matrix for any of the following elements: TRIA1, TRIA2, QUAD1, QUAD2.
3. Calling Sequence: CALL KTRIQQ (IARG)

$$IARG = \begin{cases} 1 - \text{TRIA1 element.} \\ 2 - \text{TRIA2 element.} \\ 3 - \text{QUAD1 element.} \\ 4 - \text{QUAD2 element.} \end{cases}$$

4.27.8.16 Subroutine Name: KELAS

1. Entry Point: KELAS
2. Purpose: To generate stiffness matrix contributions from the ELAS1, ELAS2, ELAS3 and ELAS4 elements and structural damping matrix contributions from the ELAS1, ELAS2 and ELAS3 elements.
3. Calling Sequence: CALL KELAS (IARG)

IARG - Indicates the type of element being processed. It can take on the values 1, 2, 3 and 4 denoting the ELAS1, ELAS2, ELAS3 and ELAS4 elements respectively.
Integer-input.

4.27.8.17 Subroutine Names: KCONE, KCONEX

1. Entry Point: KCONE, KCONEX
2. Purpose: To generate the element stiffness matrix for a conical shell problem.
3. Calling Sequence: CALL KCONE and CALL KCONEX

4.27.8.18 Subroutine Name: KTRIRG

1. Entry Point: KTRIRG
2. Purpose: To calculate an element stiffness matrix for a triangular cross-section ring, TRIARG, element.
3. Calling Sequence: CALL KTRIRG

4.27.8.19 Subroutine Name: KTRAPR

1. Entry Point: KTRAPR
2. Purpose: To calculate an element stiffness matrix for a trapezoidal cross-section ring, TRAPRG, element.
3. Calling Sequence: CALL KTRAPR

4.27.8.20 Subroutine Name: KTØRDR

1. Entry Point: KTØRDR
2. Purpose: To calculate an element stiffness matrix for a toroidal thin shell ring, TØRDRG, element.
3. Calling Sequence: CALL KTØRDR

4.27.8.21 Function Name: DKI

1. Entry Point: DKI
2. Purpose: To evaluate integrals in double precision for the triangular and trapezoidal cross-section rings in subroutines KTRIRG and KTRAPR.
3. Calling Sequence: DP = DKI (I,J,K,L,M,N,IP,IQ,R,Z)

I, J - The subscripts of R defining two lines on the limit of integration, integer-input.

K, L - The subscripts of R, Z defining another line on the limit of integration, integer-input.

M, N - The subscripts of R, Z defining the fourth line on the limit of integration, integer-input.

MODULE FUNCTIONAL DESCRIPTIONS

- IP,IQ - Integers that define the power of the r and z variables respectively,
- input.
- R,Z - Vectors of the r and z coordinates of all points used to describe the
area of integration, double precision - input.

4.27.8.22 Function Name: DKINT

1. Entry Point: DKINT
2. Purpose: To evaluate the following function in the FORTRAN function routine DKINT:

$$f_1(A,B) = \frac{1}{W} \sum_{t=0}^W CDEF \cdot A^t \cdot B^{W-t} \cdot AJ \quad (20)$$

where

$$CDEF = \begin{cases} \prod_{s=1}^t \frac{W-s+1}{s} & \text{for } t \neq 0 \\ 1 & \text{for } t = 0 \end{cases} \quad (21)$$

and

$$AJ = \begin{cases} \frac{R(J)^{(W+V-t+1)} - R(I)^{(W+V-t+1)}}{W+V-t+1} & \text{for } (W+V-t+1) \neq 0 \\ \ln\left[\frac{R(J)}{R(I)}\right] & \text{for } (W+V-t+1) = 0. \end{cases} \quad (22)$$

3. Calling Sequence: DP = DKINT (I,J,A,B,V,W,R,Z)

- I, J - The subscripts of R, Z.
- A, B - The arguments of the function f in Equation 20.
- V, W - Integer parameters of the function.
- R, Z - Vectors of the r and z coordinates.

4.27.8.23 Function Name: DK89

1. Entry Point: DK89

2. Purpose: To evaluate the following function in function DK1.

$$DK89(I,A,B) = \frac{1}{B^{M+1}} \sum_{s=0}^M M! (-A)^s \cdot d, \quad (25)$$

where

$$d = \begin{cases} \frac{(A+B \cdot R(I))^{(M+1-N-S)}}{(M-S)! S! (M+1-N-S)} & \text{for } (M+1-N-S) \neq 0 \\ \frac{\ln(|A + B \cdot R(I)|)}{(M+1-N)! (N-1)!} & \text{for } (M+1-N-S) = 0 \end{cases} \quad (26)$$

3. Calling Sequence: DP = DK89 (I,A,B,M,N,R)

- I - The subscript of R.
- A, B - The arguments of the function.
- M, N - Integer parameters of the function.
- R - Vector of the r coordinates.

4.27.8.24 Function Name: DK100

1. Entry Point: DK100

2. Purpose: To evaluate the following function in subroutine DK1.

$$DK100(I,A,B) = \frac{-1}{A^{(M+N-1)}} \sum_{s=0}^{M+N-2} (M+N-2)! \cdot d, \quad (27)$$

where

$$d = \begin{cases} \frac{(A+B \cdot R(I))^{(M-1-S)} \cdot (-B)^S}{(M+N-2-S)! S! (M-1-S) \cdot R(I)^{(M-1-S)}} & \text{for } (M-1-S) \neq 0 \\ \frac{(-B)^{M-1} \cdot \ln\left(\frac{|A+B \cdot R(I)|}{R(I)}\right)}{(M-1)! (N-1)!} & \text{for } (M-1-S) = 0 \end{cases} \quad (28)$$

MODULE FUNCTIONAL DESCRIPTIONS

3. Calling Sequence: DP= DK100 (I,A,B,M,N,R)

- I - The subscript of R.
- A, B - The arguments of the function.
- M, N - Integer parameters of the function.
- R - Vector of the r coordinates.

4.27.8.25 Function Name: DK211

1. Entry Point: DK211
2. Purpose: To evaluate the following function in FØRTRAN function DK1.

$$DK211(I,A,B) = \begin{cases} 0 & \text{for } B \cdot R(I) = A \\ \frac{1}{2} [\ln(|2 \cdot B \cdot R(I)|)]^2 & \text{for } B \cdot R(I) \neq A, [B \cdot R(I)]^2 = A^2 \\ [\ln|A|] \cdot [\ln|R(I)|] - \sum_{t=1}^{\infty} \frac{1}{t^2} \left[\frac{-B \cdot R(I)}{A} \right]^t & \text{for } [B \cdot R(I)]^2 < A^2 \\ \frac{1}{2} [\ln(|B \cdot R(I)|)]^2 + \sum_{t=1}^{\infty} \frac{1}{t^2} \left[\frac{-A}{B \cdot R(I)} \right]^t & \text{for } [B \cdot R(I)]^2 > A^2 \end{cases} \quad (31)$$

3. Calling Sequence: DP = DK211 (I,A,B,R)

- I - The subscript of R.
- A, B - The arguments of the function.
- R - Vector of the r coordinates.

4.27.8.26 Subroutine Name: RØMBDK

1. Entry Point: RØMBDK
2. Purpose: To evaluate integrals in double precision for the toroidal thin shell ring in subroutine KTØRDR.

3. Calling Sequence: CALL RØMBDK (A,B,NØSIG,PRECIS,NUM,ITDØNE,FINTG,KØDE,FUNCT,X)

- A, B - Lower and upper limit of integration respectively.
- NØSIG - Number of correct significant digits desired.
- PRECIS - Actual number of significant digits attained.
- NUM - Maximum number of halvings of the interval [A,B] to be made.
- ITDØNE - Actual number of halvings of the interval [A,B].
- FINTG - Resultant value of integral.
- KØDE - Print control (not used).
- FUNCT - Function subprogram used to evaluate the integral.
- X - Vector of parameters used by function subprogram.

4.27.8.27 Subroutine Name: DMATRIX

1. Entry Point: DMATRIX
2. Purpose: To form the element stiffness matrix in field coordinates for the toroidal thin shell ring in subroutine KTØRDR.
3. Calling Sequence: CALL DMATRIX (D,V,C,CA,CA2,VA,DM,DB,YI)
 - D - Resultant stiffness matrix.
 - V,C,CA,
CA2,VA,
DM,DB - Terms used in the evaluation of the stiffness matrix.
 - YI - Array of integral values.

4.27.8.28 Subroutine Name: KFLUD2

1. Entry Point: KFLUD2
2. Purpose: To form the element psuedo stiffness matrix for the FLUID2 and AXIF2 elements.
3. Calling Sequence: CALL KFLUD2.

4.27.8.29 Subroutine Name: KFLUD3

1. Entry Point: KFLUD3
2. Purpose: To form the element psuedo stiffness matrix for the FLUID3 and AXIF3 elements.
3. Calling Sequence: CALL KFLUD3

MODULE FUNCTIONAL DESCRIPTIONS

4.27.8.30 Subroutine Name: KFLUD4

1. Entry Point: KFLUD4
2. Purpose: To form the element psuedo stiffness matrix for the FLUID4 and AXIF4 elements.
3. Calling Sequence: CALL KFLUD4

4.27.8.31 Subroutine Name: KSLØT

1. Entry Point: KSLØT
2. Purpose: To form the element psuedo stiffness matrix for the SLØT3 and SLØT4 elements.
3. Calling Sequence: CALL KSLØT(IARG)

0 = SLØT3 elements
IARG =
1 = SLØT4 elements

4.27.8.32 Subroutine Name: KTETRA

1. Entry Point: KTETRA
2. Purpose: To calculate and insert element stiffness matrices for the TETRA (solid tetrahedron) element. It is also used for the subelements of the WEDGE, HEXA1, and HEXA2 elements.
3. Calling sequence: CALL KTETRA (IØPT), where:
 - If IØPT = 0, the stiffness is divided by 2.
 - If IØPT = 1, the stiffness is unmodified.
 - If IØPT \geq 100, the element is tested for geometric consistency.

4.27.8.33 Subroutine Name: KSØLID

1. Entry Point: KSØLID
2. Purpose: To perform, on the WEDGE, HEXA1, and HEXA2 elements, the following tasks:
 - a) Check geometric consistency.
 - b) Rearrange the ECPT data into the TETRA format for each subelement and call the KTETRA subroutine.

4.27.8.34 Subroutine Name: HRING

1. Entry Point: HRING
2. Purpose: To calculate thermal conductivity matrix terms for the TRIARG and TRAPRG elements.
3. Calling Sequence: CALL HRING(ITYPE), where
ITYPE = 3 implies a TRIARG element.
ITYPE = 4 implies a TRAPRG element.

4.27.8.35 Subroutine Name: KPLTST

1. Entry Point: KPLTST
2. Purpose: To examine the planarity of quadrilateral elements.
3. Calling Sequence: CALL KPLTST (G1,G2,G3,G4) where
Gi = Grid Point coordinate vectors

4.27.8.36 Subroutine Name: KQDMM1

1. Entry Point: KQDMM1
2. Purpose: Computes element stiffness matrix for the QDMM1 element.
3. Calling Sequence: CALL KQDMM1

4.27.8.37 Subroutine Name: KTRIA

1. Entry Point: KTRIA
2. Purpose: To generate the element stiffness matrix for an axisymmetric triangular ring element.
3. Calling Sequence: CALL KTRIA

4.27.8.38 Subroutine Name: KTPZ

1. Entry Point: KTPZ
2. Purpose: To calculate an element stiffness matrix for an axisymmetric trapezoidal ring element.
3. Calling Sequence: CALL KTPZ

MODULE FUNCTIONAL DESCRIPTIONS

3. Calling sequence: CALL KSOLID (ITYPE), where
ITYPE = 1 implies a WEDGE element (three tetrahedra).
ITYPE = 2 implies a HEXA1 element (five tetrahedra).
ITYPE = 3 implies a HEXA2 element (ten tetrahedra).

4.27.8.39 Subroutine Name: KQUADS

1. Entry Point: KQUADS
2. Purpose: To generate the element stiffness matrix for a quadrilateral thin shell element.
3. Calling Sequence: CALL KQUADS

4.27.8.40 Subroutine Name: KTRIAS

1. Entry Point: KTRIAS
2. Purpose: To generate the element stiffness matrix for a triangular thin shell element.
3. Calling Sequence: CALL KTRIAS

4.27.8.41 Subroutine Name: KIHEx

1. Entry Point: KIHEx
2. Purpose: To generate the element stiffness matrix for an IHEx1, IHEx2, or IHEx3 element.
3. Calling Sequence: CALL KIHEx (ITYPE)

$$\text{ITYPE} = \begin{cases} 1 - \text{IHEx1} \\ 2 - \text{IHEx2} \\ 3 - \text{IHEx3} \end{cases} \quad - \text{integer-input}$$

4.27.8.42 Subroutine Name: IHExSD

1. Entry Point: IHExSD
2. Purpose: To generate isoparametric shape function data.
3. Calling Sequence: CALL IHExSD (ITYPE, SHP, DSHP, JINV, DETJ, EID, XI, ETA, ZETA, CORD)
ITYPE - same as in KIHEx calling sequence - integer-input
SHP - array of values of the shape functions - double precision-output

FUNCTIONAL MODULE SMA1 (STRUCTURAL MATRIX ASSEMBLER - PHASE 1)

DSHP - array of values of the derivatives with respect to element coordinates of the shape functions - double precision-output

JINV - array of the inverse of the Jacobian matrix - double precision-output

DETJ - determinant of the Jacobian matrix - double precision-output

EID - Element ID - integer-input

XI
ETA - Element coordinates at which shape functions are evaluated - double
ZETA precision-input

CORD - array of basic coordinates of the grid points of the element - real-input

4.27.9 Design Requirements

4.27.9.1 Open Core Design

The open core common block for module SMA1 is defined by the following FØRTRAN statements:

1. DOUBLE PRECISION DZ(1)
2. INTEGER IZ(1)
3. COMMON /SMA1X/ Z(2)
4. EQUIVALENCE (Z(1),IZ(1),DZ(1)).

The open core layout is given in Figure 1.

MODULE FUNCTIONAL DESCRIPTIONS

ICSTM	COMMON /SMA1X/
IMAT1	CSTM Data
IGPCT	MPT and DIT to be read into core by MAT or HMAT
IPØINT	GPCT Data
16X6K	Pointer table defined in Equations 1 and 2
	Submatrices for KGGX and K4GG for the current pivot point.
IGGPST	
IGGPCT	GINØ buffer for GPST
IGECPT	GINØ buffer for GPCT
IG4GG	GINØ buffer for ECPT
IGKGG	GINØ buffer for K4GG
	GINØ buffer for KGGX

Figure 1. Open core layout for module SMA1.

The definition of the variables is as follows:

- ICSTM - The zero pointer to the CSTM portion of open core; defined to be zero.
- IMAT1 - The zero pointer to the MPT and DIT data read into core by subroutine PREMAT; defined to be ICSTM + NCSTM, where NCSTM is the length of the CSTM portion of open core.
- IGPCT - The zero pointer to the GPCT portion of open core; defined to be IMAT1 + MATCR, MATCR being the length of open core used by subroutine PREMAT.
- IPØINT - The zero pointer to the pointer table in open core. The pointer table is used as a dictionary to relate the GPCT to the submatrices in core; defined to be IGPCT + NGPCT, NGPCT being the length of the GPCT.
- I6X6K - The zero pointer to the submatrices of KGGX; $I6X6K = (IPØINT + NPØINT - 1) / 2 + 2$ where $NPØINT = NGPCT$ is the length of the pointer table. The extra arithmetic to define I6X6K is necessary because the submatrices are double precision numbers. While the above indices are single precision indices, I6X6K is a double precision index.
- I6X64 - The zero pointer to the submatrices of K4GG; $I6X64 = I6X6K + N6X6K$, where N6X6K is the number of double precision numbers in the submatrices of KGGX. I6X64 is a double precision index.

The pointers for the GINØ buffers, IGGPST, IGGPCT, IGECPCT, IG4GG and IGKGG are, unlike the above, 'one' pointers. It should be noted that the lengths NCSTM and MATCR are constant throughout the module operation, while the length of the GPCT data will vary from pivot point to pivot point as the ECPT and GPCT data blocks are processed serially. (Hence it is probable that for a pivot point with a relatively small number of elements connected to it the entire submatrix may be held in core, while spill logic will be entered only when a pivot point has a great many elements connected to it.)

4.27.9.2 Block Data Subprogram

The block data program SMA1BD sets GINØ file numbers, I/Ø parameters and SMA1 overlay parameters in common blocks /SMA1IØ/ and /SMA1CL/.

4.27.9.3 Common Storage Requirements

Blank common is used only for DMAP parameters. The following common blocks are used throughout the module: /SMA1IØ/, /SMA1BK/, /SMA1CL/, /SMA1ET/ and /SMA1DP/. They are given in detail here since other matrix assembler modules such as SMA2 and DSMG1 are designed similarly. All common block variables are integer except (1) DØDET in /SMA1CL/ which is a logical variable; (2) the array in /SMA1ET/ which is a mixed (integer and real) array; and (3) the double precision array in /SMA1DP/.

MODULE FUNCTIONAL DESCRIPTIONS

1. The SMA1IØ common block is 36 words in length and is used for SMA1 input/output parameters.

<u>Word Number</u>	<u>Variable</u>	<u>Definition</u>
1	IFCSTM	GINØ file number for the CSTM data block.
2	IFMPT	GINØ file number for the MPT data block.
3	IFDIT	GINØ file number for the DIT data block.
4	IDUM1	Undefined.
5	IFECPT	GINØ file number for the ECPT data block.
6	IGECPT	GINØ buffer pointer for the ECPT.
7	IFGPCT	GINØ file number for the GPCT data block.
8	IGGPCT	GINØ buffer pointer for the GPCT.
9-10	IFGEI,IGGEI	Undefined.
11	IFKGG	GINØ file number for the KGGX data block.
12	IGKGG	GINØ buffer pointer for KGGX.
13	IF4GG	GINØ file number for the K4GG data block.
14	IG4GG	GINØ buffer pointer for K4GG.
15	IFGPST	GINØ file number for the GPST data block.
16	IGGPST	GINØ buffer pointer for the GPST.
17	INRW	Input with rewind option for subroutine ØPEN.
18	ØUTRW	Output with rewind option for subroutine ØPEN.
19	CLSNRW	Close without rewind option for subroutine CLØSE.
20	CLSRW	Close with rewind option for subroutine CLØSE.
21	NEØR	No end-of-record indicator for subroutine READ.
22	EØR	End-of-record indicator for subroutine READ.
23-29	MCBKGG	Matrix control block for the KGGX matrix.
30-36	MCB4GG	Matrix control block for the K4GG matrix.

2. The SMA1EK common block is 10 words in length and is used for SMA1 open core bookkeeping parameters. It contains zero pointers and lengths for the various sub-arrays in open core.

FUNCTIONAL MODULE SMA1 (STRUCTURAL MATRIX ASSEMBLER - PHASE 1)

<u>Word Number</u>	<u>Variable</u>	<u>Definition</u>
1	ICSTM	Zero pointer to the CSTM sub-array in open core. For example the first location of this sub-array is referenced as IZ (ICSTM + 1).
2	NCSTM	Length of the CSTM sub-array in open core.
3	IGPCT	Zero pointer to the GPCT sub-array in open core.
4	NGPCT	Length of the GPCT sub-array.
5	IPØINT	Zero pointer to the PØINT sub-array in open core.
6	NPØINT	Length of the PØINT sub-array.
7	I6X6K	Zero pointer to the 6x6 submatrices of KGGX.
8	N6X6K	Number of words allocated to the 6x6 submatrices of KGGX.
9	I6X64	Zero pointer to the 6x6 submatrices of K4GG.
10	N6X64	Undefined.

3. The SMA1CL common block is 133 words in length and is used for module control parameters.

<u>Word Number</u>	<u>Variable</u>	<u>Definition</u>
1	IØPT4	Indicator used by element routines denoting whether or not the K4GG matrix will be generated. IØPT4 = 0, implies no generation; IØPT4 = 1 implies generation.
2	K4GGSW	Indicator set to -1 initially. If IØPT4 = 1, then element routines will set K4GGSW = 1, when a non-zero element structural damping matrix is generated.
3	NPVT	The scalar index which is the pivot point. This is the first word of every record of the ECPT data block.
4	LEFT	The number of words of open core remaining after all sub-arrays in open core have been allocated.
5	FRØWIC	The first row of the submatrices in core. If all six rows of the matrices to be generated cannot be held in core, spill logic is initiated, and 3, 2 or 1 rows of the submatrices are generated during each pass of the ECPT record for the pivot point which causes the spill. FRØWIC can take on the values 1,2,3,4,5 or 6.
6	LRØWIC	The last row of the submatrices in core. LRØWIC is defined as FRØWIC + NRØWSC -1, where NRØWSC is the number of rows in core. If there are no spill problems, then LRØWIC = 6 if the pivot point is a grid point, and LRØWIC = 1 if the pivot point is a scalar point.
7	NRØWSC	The number of rows of the submatrices currently in core.
8	TNRØWS	Total number of rows of the submatrices to be generated. TNRØWS = 6 if the pivot point is a grid point and TNRØWS = 1 if the pivot point is a scalar point. This definition holds whether or not the K4GG matrix is to be generated. (In actuality, if the K4GG is generated the total number of rows

MODULE FUNCTIONAL DESCRIPTIONS

<u>Word Number</u>	<u>Variable</u>	<u>Definition</u>
		generated for any ECPT record is 12 or 2).
9	JMAX	The number of columns of KGGX (and K4GG) to be generated with the current ECPT record.
10	NLINKS	The number of machine links (overlay segments) necessary to contain the module's element routines. Currently NLINKS is defined to be 3. This variable is used in conjunction with the LINK array defined below. For machines with large memories, it is desirable to have all element routines in one link, for when in any ECPT record there are elements which reside in different links, overlay overhead can be very costly (particularly on second generation computing systems).
11-20	LINK	Before the current ECPT record is read for the first time, the LINK array is set to -1 for LINK(I), I = 1, ...NLINKS. When the first element is read from the ECPT, the proper element routine is called, thereby loading the link in which that element routine resides. The variable LINCØR, the link in core, is defined as LINCØR = IØVRLY(ITYPE), where ITYPE is the element's internal number, e.g., RØD = 1, BEAM = 2, etc. For the next element read from the ECPT, it is determined in what link it resides. If it resides in the link which is in core, the element routine is called. If the routine does not reside in the link currently in core, it is determined whether (a) the link has already been processed or (b) the link has not been processed in which case a "to-be-processed-later" flag is set. For case (a) LINK (ITEMP) is 1; for case (b) LINK(ITEMP) is set = 0, where ITEMP = IØVRLY(ITYPE). When an end-of-record is sensed for the ECPT, LINK(LINCØR) is set to 1 and LINK array is searched for zeros. If there are no zeros, the processing of the ECPT record is complete. If there are zeros, that is, links to be processed, the file corresponding to the ECPT is backspaced and processing of the record is repeated.
21	NØGØ	Flag used to indicate if a user fatal error message occurred in the processing of any element. NØGØ = 1 indicates an error. Execution is termination upon completion of the processing of the GPCT. NØGØ = 0 indicates no error. Continue execution.
22	IDETCK	Used as a first pass indicator in the DETCK subroutine. There will be multiple passes through the DETCK routine, for each ECPT record, only if there are spill problems, i.e., the total number of rows to be generated for the ECPT record will not fit in core.

23 DØDET Logical variable which if true implies the DETCK routine will be called and if false will not be called. If the input parameter, NØGENL, is greater than zero, implying general elements exist, then DØDET is set false. Otherwise DØDET is true.

4. The common block SMA1ET is 100 words in length and is used as the means of communicating the element data from the ECPT data block to the element subroutines.

5. The common block SMA1DP defines an array of 300 double precision words. This block is used as "scratch" storage by element routines. Those variables which in most FØRTRAN programs would be local subroutine variables are defined in /SMA1DP/ by the module's element routines in order to preserve core storage and hence increase open core.

4.27.9.4 Arithmetic Considerations

All floating point arithmetic operations are carried out in double precision. Both $[K_{gg}^x]$ and $[K_{gg}^4]$ are double precision matrices.

4.27.10 Diagnostic Messages

The module has a variety of "fail-safe" error checks. If any of these checks fails, it implies an obscure program design error or a computer operating system/hardware failure. Diagnostic messages 2022, 2023, 2034 are of this type.

User fatal error messages 2025, 2026, 2031, 2032, 2033, 2035, 2036, 2037, 2038, 2039, 2040, 5001, 5002, 5003, and 5004 occur when one of the structural element routines encounters some user data which makes generation of an element matrix impossible. Examples would include a user defining a RØD or BAR element of zero length; a user defining the four points of a SHEAR panel element not in the proper cyclical order; a user defining TRPLT data so that a matrix in a algorithm is singular; etc. It should be noted the first time this type of user data is encountered a fatal error occurs, that is the module does not continue to process data for uncovering any more errors.

The following additional user fatal error messages may be issued by the isoparametric solid element routines: 3301, 3302 and 3306.

Detailed descriptions of these error messages can be found in Section 6 of the User's Manual.

4.28 FUNCTIONAL MODULE SMA2 (STRUCTURAL MATRIX ASSEMBLER - PHASE 2)

4.28.1 Entry Point: SMA2

4.28.2 Purpose

To generate the mass matrix $[M_{gg}]$ and the damping matrix $[B_{gg}]$.

4.28.3 DMAP Calling Sequence

SMA2 CSTM,MPT,ECPT,GPCT,DIT/MGG,BGG/V,Y,WTMASS=1.0/V,N,NØMGG/V,N,NØBGG/V,Y,
CØUPMASS/V,Y,CPBAR/V,Y,CPRØD/V,Y,CPQUAD1/V,Y,CPQUAD2/V,Y,CPTRIA1/V,Y,CPTRIA2/
V,Y,CPTUBE/V,Y,CPQDPLT/V,Y,CPTRPLT/V,Y,CPTRBSC/ \$

4.28.4 Input Data Blocks

CSTM - Coordinate System Transformation Matrices.

MPT - Material Properties Table.

ECPT - Element Connection and Properties Table.

GPCT - Grid Point Connection Table.

DIT - Direct Input Tables.

Notes:

1. The CSTM may be purged.
2. If some element references a material property, the MPT cannot be purged.
3. Neither the ECPT nor the GPCT may be purged.
4. If some material property is temperature dependent, DIT cannot be purged.

4.28.5 Output Data Blocks

MGG - Partition of mass matrix - g set.

BGG - Partition of damping matrix - g set.

Notes:

1. MGG cannot be pre-purged.
2. BGG can be pre-purged.

MODULE FUNCTIONAL DESCRIPTIONS

4.28.6 Parameters

- WTMASS - Input-real-default value (in DMAP calling sequence) = 1.0. WTMASS is the scalar value by which the generated mass matrix will be multiplied before the columns are packed onto the output file. WTMASS is the ratio of mass to weight.
- NØMGG - Output-integer-no default value. NØMGG is set equal to -1 if the generated mass matrix is the zero matrix. Otherwise it is set = +1.
- NØBGG - Output-integer-no default value. If the BGG matrix is either pre-purged or is generated as the zero matrix, NØBGG is set = -1. Otherwise it is set = +1.
- CØUPMASS - Input-integer-default value = -1. If CØUPMASS = -1, "consistent" mass matrices for all elements will not be generated; if CØUPMASS > 0, "consistent" mass matrices for all elements will be generated. If CØUPMASS = 0 "consistent" mass matrices will be generated for element types depending on the values of CPBAR, CPRØD, CPQUAD1, CPQUAD2, CPTRIA1, CPTRIA2, CPTUBE, CPQDPLT, CPTRPLT, and CTRBSC.
- CPBAR, CPRØD, CPQUAD1, CPQUAD2, CPTRIA1, CPTRIA2, CPTUBE, CPQDPLT, CPTRPLT and CTRBSC - Input - integer-default value = -1.

These parameters choose between "consistent" mass and normal mass algorithms for their respective element types as given below.

<u>Parameter</u>	<u>Element Types</u>
CPBAR	BAR
CPRØD	RØD, CØNRØD
CPQUAD1	QUAD1
CPQUAD2	QUAD2
CPTRIA1	TRIA1
CPTRIA2	TRIA2
CPTUBE	TUBE
CPQDPLT	QDPLT
CPTRPLT	TRPLT
CTRBSC	TRBSC

These parameters function in conjunction with CØUPMASS and have no effect if CØUPMASS ≠ 0. If CØUPMASS = 0 a negative value for these parameters will cause the "normal" mass matrix to be generated. A value equal to or greater than zero will cause the "consistent" mass matrices to be generated for all element types controlled by this parameter.

4.28.7 Method

SMA2 is structured similarly to module SMA1. A separate module was written to generate the mass and damping matrices in order to maximize the amount of open core space available for element matrices during matrix generation. This core space was especially critical on the development computer, the IBM 7094-7040 DCS. Since SMA2 is so similar to SMA1, the details of the similarities will not be repeated here; the differences will be pointed out. The reader is referred to the Module Functional Description (MFD) for SMA1 (Section 4.27).

When all rows (or, in the case of spill, the number of rows in core) for each pivot point have been computed, each matrix element of $[M_{gg}]$ is multiplied by WTMASS before being packed onto the output data block MGG.

If the heat transfer option is being used, the following differences occur. Subroutine HMAT is called, rather than subroutine PREMAT to set up the material thermal coefficient tables. The element routines will insert thermal capacity terms in the BGG matrix. The WTMASS parameter is not used.

4.28.8 Subroutines

SMA2, like SMA1, uses the utility routines PRETRD, PREMAT and GMMATD.

The principal means of communicating an element entry of the ECPT to an element mass or damping matrix generation routine is through /SMA2ET/. This fact is not explicitly stated in each of the descriptions of the element routines (e.g., MRØD) given below.

The following list gives a correspondence between SMA1 and SMA2 routines that are used only (directly or indirectly) by the axisymmetric shell element routines TRIARG and TRAPRG. All of the SMA2 routines are the same as their SMA1 counterparts except for name. The reason for duplicating these routines with different names was to maximize open core for element matrices in SMA1 and SMA2, which both reside in the same NASTRAN link. For details on each of the routines, see the corresponding SMA1 counterpart (Section 4.27.8).

<u>SMA1</u>	<u>SMA2</u>
DKI	DMI
DKINT	DMINT
DK89	DM89
DK100	DM100
DK211	DM211

4.28.8.1 Subroutine Name: SMA2

1. Entry Point: SMA2
2. Purpose: The module driver which parallels SMA1. For further details see the method section of the MFD for SMA1 (section 4.27.7).
3. Calling Sequence: CALL SMA2

4.28.8.2 Subroutine Name: SMA2A

1. Entry Point: SMA2A
2. Purpose: To generate $[M_{gg}]$ and $[B_{gg}]$. This routine parallels SMA1A. See the MFD for SMA1 for details on SMA1A (section 4.27.8).
3. Calling Sequence: CALL SMA2A.

4.28.8.3 Subroutine Name: SMA2B

1. Entry Point: SMA2B
 2. Purpose: To add a double precision 6 by 6 or 1 by 1 matrix $[K^e]$ to the "submatrix" of $[M_{gg}]$ or $[B_{gg}]$ corresponding to the current pivot point.
 3. Calling Sequence: CALL SMA2B (KE,J,II,IFILE,DUMDP).
- KE,J,II are as defined for subroutine SMA1B (see section 4.27.8).
- IFILE - GINØ file number of the matrix in core being added to $[M_{gg}]$ or $[B_{gg}]$ -integer-input.
- DUMDP - A dummy double precision argument added so that the calling sequence to SMA2B would conform to that of SMA1B.

4.28.8.4 Block Data Program Name: SMA2BD.

1. Purpose: To set GINØ file numbers, I/Ø parameters and SMA2 overlay parameters in /SMA2IØ/ and /SMA2CL/.

FUNCTIONAL MODULE SMA2 (STRUCTURAL MATRIX ASSEMBLER - PHASE 2)

4.28.8.5 Subroutine Name: MRØD

1. Entry Point: MRØD
2. Purpose: To generate the element mass matrix for a RØD element.
3. Calling Sequence: CALL MRØD.

4.28.8.6 Subroutine Name: MTUBE

1. Entry Point: MTUBE
2. Purpose: To generate the element mass matrix for a TUBE element.
3. Calling Sequence: CALL MTUBE

4.28.8.7 Subroutine Name: MASSTQ

1. Entry Point: MASSTQ
2. Purpose: To generate an element mass matrix for any of the two-dimensional structural elements listed under the Calling Sequence.
3. Calling Sequence: CALL MASSTQ(IARG)

IARG =	{	4	=	TRMEM
		1	=	QDNEM
		3	=	TRBSC
		3	=	TRPLT
		7	=	QDPLT
		5	=	TRIA1
		4	=	TRIA2
		2	=	QUAD1
		1	=	QUAD2
		6	=	SHEAR
6	=	TWIST		

4.28.8.8 Subroutine Name: MBAR

1. Entry Point: MBAR
2. Purpose: To generate the "diagonal" (uncoupled) element mass matrix for a BAR element.
3. Calling Sequence: CALL MBAR

4.28.8.9 Subroutine Name: MCBAR

1. Entry Point: MCBAR
2. Purpose: To generate the "consistent" (coupled) element mass matrix for a BAR element.
3. Calling Sequence: CALL MCBAR

4.28.8.10 Subroutine Name: MCØNMX

1. Entry Point: MCØNMX
2. Purpose: To generate an element mass matrix for either of the two concentrated-mass-elements listed under Calling Sequence.
3. Calling Sequence: CALL MCØNMX(IARG)

$$IARG = \begin{cases} 1 & = CØNM1 \\ 2 & = CØNM2 \end{cases}$$

4.28.8.11 Subroutine Name: MCØNE

1. Entry Point: MCØNE
2. Purpose: To generate an element mass matrix for the axisymmetric conical shell element (CØNE).
3. Calling Sequence: CALL MCØNE

4.28.8.12 Subroutine Name: MASSD

1. Entry Point: MASSD
 2. Purpose: To generate 1 by 1 element mass matrices for scalar elements MASS_i, $i = 1, 2, 3, 4$; and 1 by 1 element damping matrices for scalar damping elements DAMP_i, $i = 1, 2, 3, 4$.
 3. Calling Sequence: CALL MASSD(I)
- I - {
 - 1 - Generate element mass matrix for a MASS1 element,
 - 2 - Generate element mass matrix for a MASS2 element,
 - 3 - Generate element mass matrix for a MASS3 element,
 - 4 - Generate element mass matrix for a MASS4 element,
 - 5 - Generate element damping matrix for a DAMP1 element,
 - 6 - Generate element damping matrix for a DAMP2 element,
 - 7 - Generate element damping matrix for a DAMP3 element,
 - 8 - Generate element damping matrix for a DAMP4 element.

4.28.8.13 Subroutine Name: MTRIRG

1. Entry Point: MTRIRG
2. Purpose: To generate an element mass matrix for a triangular cross-section ring, TRIARG, element.
3. Calling Sequence: MTRIRG.

4.28.8.14 Subroutine Name: MTRAPR

1. Entry Point: MTRAPR
2. Purpose: To generate an element mass matrix for a trapezoidal cross-section ring, TRAPRG, element.
3. Calling Sequence: CALL MTRAPR.

4.28.8.15 Subroutine Name: MTØRDR

1. Entry Point: MTØRDR

MODULE FUNCTIONAL DESCRIPTIONS

2. Purpose: To generate an element mass matrix for a toroidal thin shell ring, TØRDRG, element.

3. Calling Sequence: CALL MTØRDR.

4.28.8.16 Subroutine Name: BVISC

1. Entry Point: BVISC

2. Purpose: To generate an element damping matrix for a VISC element.

3. Calling Sequence: Call BVISC

4.28.8.17 Subroutine Name: MCRØD

1. Entry Point: MCRØD

2. Purpose: To generate the "consistent" (coupled) element mass matrix for any of the elements listed under calling sequence.

3. Calling Sequence: CALL MCRØD (IARG)

$$IARG = \begin{cases} 1 - RØD \\ 1 - CØNRØD \\ 3 - TUBE \end{cases}$$

4.28.8.18 Subroutine Name: MTRBSC

1. Entry Point: MTRBSC

2. Purpose: To generate the "consistent" (coupled) element mass matrix for a basic bending triangle element.

3. Calling Sequence: CALL MTRBSC

4.28.8.19 Subroutine Name: MTRPLT

1. Entry Point: MTRPLT

2. Purpose: To generate the "consistent" (coupled) element mass matrix for a triangular plate element.

3. Calling Sequence: CALL MTRPLT

FUNCTIONAL MODULE SMA2 (STRUCTURAL MATRIX ASSEMBLER - PHASE 2)

4.28.8.20 Subroutine Name: MQDPLT

1. Entry Point: MQDPLT
2. Purpose: To generate the "consistent" (coupled) element mass matrix for a quadrilateral plate element.
3. Calling Sequence: CALL MQDPLT

4.28.8.21 Subroutine Name: MTRIQD

1. Entry Point: MTRIQD
2. Purpose: To generate the "consistent" (coupled) element for any of the following elements: TRIA1, TRIA2, QUAD1, QUAD2.
3. Calling Sequence: CALL MTRIQD (IARG)

IARG = $\left\{ \begin{array}{l} 1 - \text{TRIA1 element.} \\ 2 - \text{TRIA2 element.} \\ 3 - \text{QUAD1 element.} \\ 4 - \text{QUAD2 element.} \end{array} \right.$

4.28.8.22 Subroutine Name: MFLUD2

1. Entry Point: MFLUD2
2. Purpose: To generate the psuedo mass matrix terms for the FLUID2 and AXIF2 elements.
3. Calling Sequence: CALL MFLUD2

4.28.8.23 Subroutine Name: MFLUD3

1. Entry Point: MFLUD3
2. Purpose: To generate the pseudo mass matrix terms for the FLUID3 and AXIF3 elements.
3. Calling Sequence: CALL MFLUD3

4.28.8.24 Subroutine Name: MFLUD4

1. Entry Point: MFLUD4
2. Purpose: To generate the psuedo mass matrix terms for the FLUID4 and AXIF4 elements.
3. Calling Sequence: CALL MFLUD4

MODULE FUNCTIONAL DESCRIPTIONS

4.28.8.25 Subroutine Name: MFREE

1. Entry Point: MFREE
2. Purpose: To generate the psuedo mass matrix terms for the free surface element. This element is internally generated in IFP4 from FSLIST data.
3. Calling Sequence: CALL MFREE

4.28.8.26 Subroutine Name: MSLØT

1. Entry Point: MSLØT
2. Purpose: To generate psuedo mass matrix terms for the SLØT3 and SLØT4 elements.
3. Calling Sequence: CALL MSLØT (IARG)

$$IARG = \begin{cases} 0 & = \text{SLØT3} \\ 1 & = \text{SLØT4} \end{cases}$$

4.28.8.27 Subroutine Name: MSØLID

1. Entry Point: MSØLID
2. Purpose: To generate the mass matrix terms for the three-dimensional solid elements.
3. Calling Sequence: CALL MSØLID(I)

<u>I</u>	<u>Element</u>
1	TETRA
2	WEDGE
3	HEXA1
4	HEXA2

4.28.8.28 Subroutine Name: MHBDY

1. Entry Point: MHBDY
2. Purpose: To generate heat capacity terms in a heat transfer problem.
3. Calling Sequence: CALL MHBDY

4.28.8.29 Subroutine Name: MRING

1. Entry Point: MRING
2. Purpose: To generate heat capacity terms for the TRIARG and TRAPRG elements.

FUNCTIONAL MODULE SMA2 (STRUCTURAL MATRIX ASSEMBLER - PHASE 2)

3. Calling Sequence: CALL MRING (NPØINTS), where NPØINTS = 3 implies the TRIARG and NPØINTS = 4 implies the TRAPRG element.

4.28.8.30 Subroutine Name: DELTKL

1. Entry Point: DELTKL
2. Purpose: To evaluate integrals in double precision for the axisymmetric triangular and trapezoidal ring elements in subroutines MSTRIA and MPZDA.
3. Calling Sequence: DELTKL (A,R,Z,K)
A - Double precision array of 15 locations containing the results
R,Z - Double precision arrays of four locations each. The arrays contain the R and Z coordinates of all points to describe the area of integration
K = 0 for triangular ring
1 for trapezoidal ring

4.28.8.31 Subroutine Name: MSTRIA

1. Entry Point: MSTRIA
2. Purpose: To generate the consistent or lump element mass matrix for an axisymmetric triangular ring element.
3. Calling Sequence: CALL MSTRIA (ICMBAR)

$$\text{ICMBAR} = \begin{cases} < 0 & \text{Lump Mass} \\ \geq & \text{Consistent Mass} \end{cases}$$

4.28.8.32 Subroutine Name: MPZDA

1. Entry Point: MPZDA
2. Purpose: To generate the consistent or lump element mass matrix for an axisymmetric trapezoidal ring element.
3. Calling Sequence: CALL MPZDA

$$\text{ICMBAR} = \begin{cases} < 0 & \text{Lump Mass} \\ \geq 0 & \text{Consistent Mass} \end{cases}$$

4.28.8.33 Subroutine Name: MQUADS

1. Entry Point: MQUADS
2. Purpose: To generate the element mass matrix for a quadrilateral thin shell element

MODULE FUNCTIONAL DESCRIPTIONS

3. Calling Sequence: CALL MQUADS (ICMBAR)

$$\text{ICMBAR} = \begin{cases} <0, \text{ generate lumped mass matrix} \\ \geq 0, \text{ generate consistent mass matrix} \end{cases}$$

4.28.8.34 Subroutine Name: MTRIAS

1. Entry Point: MTRIAS
2. Purpose: To generate the element mass matrix for a triangular thin shell element
3. Calling Sequence: CALL MTRIAS (ICMBAR)

$$\text{ICMBAR} = \begin{cases} <0, \text{ generate lumped mass matrix} \\ \geq 0, \text{ generate consistent mass matrix} \end{cases}$$

4.28.8.35 Subroutine Name: MIHEX

1. Entry Point: MIHEX
2. Purpose: To generate the coupled mass matrix for an IHEX1, IHEX2, or IHEX3 element
3. Calling Sequence: CALL MIHEX (ITYPE)

$$\text{ITYPE} = \begin{cases} 1 - \text{IHEX1} \\ 2 - \text{IHEX2} \\ 3 - \text{IHEX3} \end{cases} - \text{integer-input}$$

4.28.8.36 Subroutine Name: IHEXSD

See Section 4.27.8.42.

4.28.9 Design Requirements

4.28.9.1 Open Core Design

The open core design for SMA2 is the same as that in SMA1 with the exception that /SMA2X/ defines the beginning of open core and only four buffers are needed, one each for MGG, BGG, ECPT and GPCT.

4.28.9.2 Common Storage Requirements

The common storage requirements for SMA2 are similar to those in SMA1. The common blocks /SMA2IØ/, /SMA2BK/, /SMA2CL/, /SMA2ET/ and /SMA2DP/ of SMA2 correspond to /SMA1IØ/, /SMA1BK/, /SMA1CL/, /SMA1ET/ and /SMA1DP/ of SMA1. See the MFD for SMA1 (see section 4.27.9). The following differences are worthy of note.

1. In /SMA2IØ/, words 15 and 16 are undefined and words 23 through 36 define matrix control blocks for MGG and BGG.
2. /SMA2CL/ is only 131 words in length, the last two words of /SMA1CL/ being reserved for variables unique to SMA1.

4.28.9.3 Arithmetic Considerations

Floating point arithmetic operations are carried out in double precision. Both $[M_{gg}]$ and $[B_{gg}]$ are real symmetric double precision matrices.

4.28.10 Diagnostic Messages

See the diagnostic message section in the MFD for SMA1 (section 4.27.10).

FUNCTIONAL MODULE GPWG (GRID POINT WEIGHT GENERATOR)

4.29 FUNCTIONAL MODULE GPWG (GRID POINT WEIGHT GENERATOR)

4.29.1 Entry Point: GPWG

4.29.2 Purpose

To compute the center of mass of the structure relative to a given point and find the principal inertias about the center of gravity.

4.29.3 DMAP Calling Sequence

GPWG BGPDT,CSTM,EQEXIN,MGG/ØGPWG/V,Y,GRDPNT/V,Y,WTMASS \$

4.29.4 Input Data Blocks

BGPDT - Basic Grid Point Definition Table.
CSTM - Coordinate System Transformation Matrices.
EQEXIN - Equivalence between external grid or scalar numbers and internal numbers.
MGG - Partition of mass matrix - g set.

Notes: 1. BGPDT,EQEXIN and MGG cannot be purged.
2. CSTM must be present if some grid point of the model is not in basic coordinates.

4.29.5 Output Data Blocks

ØGPWG - Grid Point Weight Generator Output Table.

Notes: This data block cannot be purged.

4.29.6 Parameters

GRDPNT - Input-integer-default = -1. GRDPNT selects the grid point about which the inertias will be calculated. If GRDPNT is not the external ID of a geometric grid point, the basic origin is used.
WTMASS - Input-real-default = 1.0. WTMASS gives the ratio of mass to weight for the structure. All output quantities are in weight units.

4.29.7 Method

The Grid Point Weight Generator module calculates the masses, centers of gravity, and inertias of the general mathematical model of the structure. The data are extracted from the $[M_{gg}]$ matrix by using a rigid body transformation calculation. The transformation is defined by the global coordinate displacements resulting from unit translations and rotations of the whole body about a reference point.

Because of the scalar mass effects, the total mass may have directional properties, and the center of gravity may not be a unique location. This effect is shown in the output by giving for each of the three masses its own direction and center of gravity. The inertia terms are calculated by using the directional mass effects. The axes about which the inertia terms are calculated may not intersect. However, these axes are those which provide uncoupled rotation and translation effects. This is the significance of the term "center of gravity". If the structural model has been constructed using only real masses, the three masses printed out will be equal, the center of gravity will be unique, and the axes of the inertia terms will intersect at the center of gravity.

The actual computation proceeds in 4 parts (see Figure 1).

1. Computation of the $[D]^T$ matrix takes place in subroutine GPWG1A. Six vectors are formed which will describe the six motions about the reference point. The matrix $[D]$ formed from the vectors which describe rigid body displacements in global coordinates in terms of the six unit displacements and rotations in basic coordinates at the reference point:

$$\{\dot{u}_g\} = [D] \{\dot{u}_0\} \quad (\text{reference point}). \quad (1)$$

The method of generation is as follows:

EQEXIN is placed in core and searched for GRDPNT to obtain its internal sequence number. If the value of the parameter GRDPNT is not the ID number of a physical grid point, the basic origin is used. Assuming GRDPNT is a physical grid point, BGPDT is read to obtain $\{R_0\}$ for the reference point. The BGPDT file is then rewound. CSTM (if present) is placed in core and readied for use by subroutine PRETRS. Each grid point in the BGPDT is considered in order. If it is a scalar point, zero is stored in each of the six columns of $[D]$.

If it is a grid point,

$$\{r_i\} = \{R_i\} - \{R_0\} = \begin{Bmatrix} r_1 \\ r_2 \\ r_3 \end{Bmatrix}, \quad (2)$$

FUNCTIONAL MODULE GPWG (GRID POINT WEIGHT GENERATOR)

is computed where $\{R_i\}$ is the basic coordinate location of the i^{th} grid point given in the BGPDT table and $\{R_0\}$ is the location of the reference point.

The transformation matrix to the grid point,

$$[T_r] = \begin{bmatrix} 0 & r_3 & -r_2 \\ -r_3 & 0 & r_1 \\ r_2 & -r_1 & 0 \end{bmatrix}, \quad (3)$$

is formed. Subroutine TRANSS calculates the 3x3 transformation matrix $[T_i]$ from global coordinates to basic coordinates for the grid point. The matrix

$$[d] = \begin{bmatrix} T_i^T & \vdots & T_i^T T_r \\ -T_i^T & \vdots & -T_i^T T_r \\ 0 & \vdots & T_i^T \end{bmatrix}, \quad (4)$$

is computed. The rows of $[d]$ form the columns of $[D]^T$. The matrix $[D]^T$ is generated a column at a time and is packed out onto a scratch file.

2. If all points were scalar points, GPWG returns; otherwise subroutine TRANP1 is called to form $[D]$ from $[D]^T$.

3. $[M_0]$ is computed by two calls to subroutine SSG2B,

$$[M_0] = [D]^T [M_{gg}] [D]. \quad (5)$$

4. Output quantities are computed as follows:

M_0 is unpacked, output and partitioned as follows:

$$[M_0] \Rightarrow \begin{bmatrix} \bar{M}^t & \vdots & \bar{M}^{tr} \\ -\bar{M}^{rt} & \vdots & \bar{M}^r \end{bmatrix} \quad (6)$$

(The matrix is symmetric, where the superscripts t and r refer to translation and rotation respectively.)

A check is made for inconsistent scalar masses. Let

$$\delta = \sqrt{\sum (\bar{M}_{ij}^t)^2} \quad (7)$$

and

$$\epsilon = \sqrt{\sum (\bar{M}_{ij}^t)^2} \quad i \neq j. \quad (8)$$

MODULE FUNCTIONAL DESCRIPTIONS

If $\frac{\epsilon}{\delta} > 10^{-3}$, the coordinates should be rotated. Otherwise $[S] = [I]$. If rotation is necessary, the eigenvectors of $[\bar{M}^t]$, $\{e_1\}$, $\{e_2\}$, and $\{e_3\}$, are determined by the Jacobi technique. Define

$$[S] = [\{e_1\}, \{e_2\}, \{e_3\}]. \quad (9)$$

The $[S]$ matrix is output. $[M^t]$, $[M^r]$ and $[M^{tr}]$ are computed as follows:

$$[M^t] = [S]^T [\bar{M}^t] [S]. \quad (10)$$

$$[M^{tr}] = [S]^T [\bar{M}^{tr}] [S]. \quad (11)$$

$$[M^r] = [S]^T [\bar{M}^r] [S]. \quad (12)$$

The following terms, defined in the principal axis system $\{e_1\}$, $\{e_2\}$, and $\{e_3\}$, are calculated and output: The mass terms are:

$$M_x = M_{11}^t, \quad (13)$$

$$M_y = M_{22}^t, \quad (14)$$

$$M_z = M_{33}^t, \quad (15)$$

the "centers of gravity" are:

$$x_x = \frac{M_{11}^{tr}}{M_x}, \quad y_x = \frac{-M_{13}^{tr}}{M_x}, \quad z_x = \frac{M_{12}^{tr}}{M_x}, \quad (16)$$

$$x_y = \frac{M_{23}^{tr}}{M_y}, \quad y_y = \frac{M_{22}^{tr}}{M_y}, \quad z_y = \frac{-M_{21}^{tr}}{M_y}, \quad (17)$$

$$x_z = \frac{-M_{32}^{tr}}{M_z}, \quad y_z = \frac{M_{31}^{tr}}{M_z}, \quad z_z = \frac{M_{33}^{tr}}{M_z}. \quad (18)$$

and the inertias at the center of gravity relative to the principal mass axis are determined from:

$$I_{11}^{(S)} = M_{11}^r - M_y z_y^2 - M_z y_z^2, \quad (19)$$

$$I_{12}^{(S)} = I_{21}^{(S)} = -M_{12}^r - M_z x_z y_z, \quad (20)$$

$$I_{13}^{(S)} = I_{31}^{(S)} = -M_{13}^r - M_y x_y z_y, \quad (21)$$

$$I_{22}^{(S)} = M_{22}^r - M_z x_z^2 - M_x z_x^2, \quad (22)$$

FUNCTIONAL MODULE GPWG (GRID POINT WEIGHT GENERATOR)

$$I_{32}^{(S)} = I_{23}^{(S)} = -M_{23}^r - M_x Y_x Z_x, \quad (23)$$

$$I_{33}^{(S)} = M_{33}^r - M_x Y_x^2 - M_y X_y^2. \quad (24)$$

These terms form the symmetric matrix [I].

For principal inertias eigenvalues and eigenvectors are found such that:

$$\begin{bmatrix} I_{11}^p & 0 & 0 \\ 0 & I_{22}^p & 0 \\ 0 & 0 & I_{33}^p \end{bmatrix} = [Q]^T [I] [Q]. \quad (25)$$

[Q] contains the normalized eigenvectors (the directions of the principal inertias), and the I_{ii}^p terms are the eigenvalues. The matrices [S] and [Q] are actually coordinate rotation matrices and show the directions of the principal masses and inertias.

4.29.8 Subroutines

Utility Subroutines PRETRS,TRANSS,TRANP1,SSG2B and GMMATS are used. See subroutine descriptions, section 3.

4.29.8.1 Subroutine Name: GPWG1A

1. Entry Point: GPWG1A
2. Purpose: To form the $[D]^T$ matrix.
3. Calling Sequence: CALL GPWG1A(PØINT,BGPDT,CSTM,EQEXIN,DT,NØGØ)
 - PØINT - External grid point id of reference point - integer - input.
 - BGPDT - GINØ file number of ØGPWG - integer - input.
 - CSTM - GINØ file number of CSTM - integer - input.
 - EQEXIN - GINØ file number of EQEXIN - integer - input.
 - DT - GINØ file number of file on which $[D]^T$ will be written - integer - input.
 - NØGØ - Flag for all scalar problem - integer - output. NØGØ = 0 implies all scalars.

4.29.8.2 Subroutine Name: GPWG1B

1. Entry Point: GPWG1B

MODULE FUNCTIONAL DESCRIPTIONS

2. Purpose: To form output quantities as given in paragraph 4 of section 4.29.7.

3. Calling Sequence: CALL GPWG1B (MØ,ØGPWG,WTMASS,IP)

- MØ - GINØ file number of [M_o] matrix - integer - input.
- ØGPWG - GINØ file number of ØGPWG - integer - input.
- WTMASS - Mass to weight ratio parameter - real - input.
- IP - External grid point id of reference point (=0 if basic origin was used)
- integer - input.

4.29.8.3 Subroutine Name: GPWG1C

1. Entry Point: GPWG1C

2. Purpose: To compute eigenvectors and values of a 3 by 3 matrix by the classical Jacobi method.

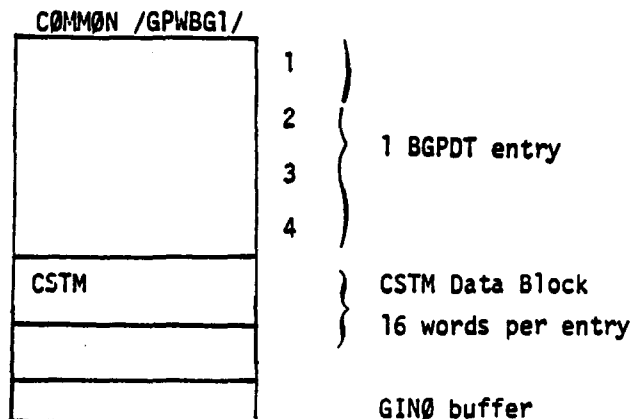
3. Calling Sequence: CALL GPWG1C (B,E,EV,IFLAG)

- B - 3 by 3 input matrix - real - input.
- E - 3 by 3 matrix of eigenvectors - real - output.
- EV - 3 eigenvalues - real - output.
- IFLAG - Error termination flag - integer - output. If IFLAG > 0, GPWG1C could not converge.

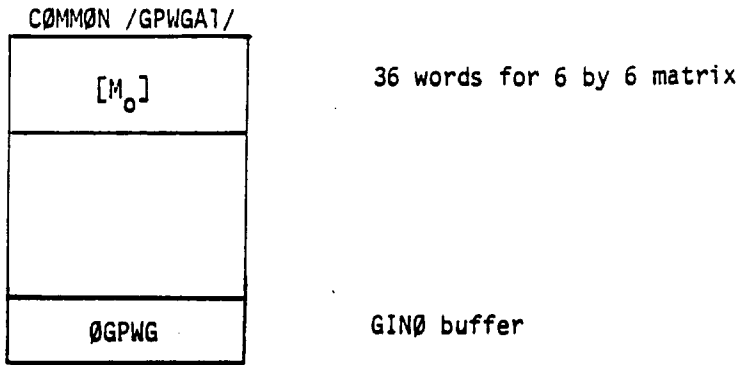
4.29.9 Design Requirements

GPWG requires four scratch files. Open core for GPWG1A is defined at /GPWGA1/. Open core for GPWG1B is defined at /GPWGB1/.

The layout of open core is as follows:



FUNCTIONAL MODULE GPWG (GRID POINT WEIGHT GENERATOR)



4.29.10 Diagnostic Messages

The following fatal error messages may occur: 3007, 3008.

MODULE FUNCTIONAL DESCRIPTIONS

FLOW CHART OF GPWG
WITH A "g" POINT IN CYLINDRICAL COORDS

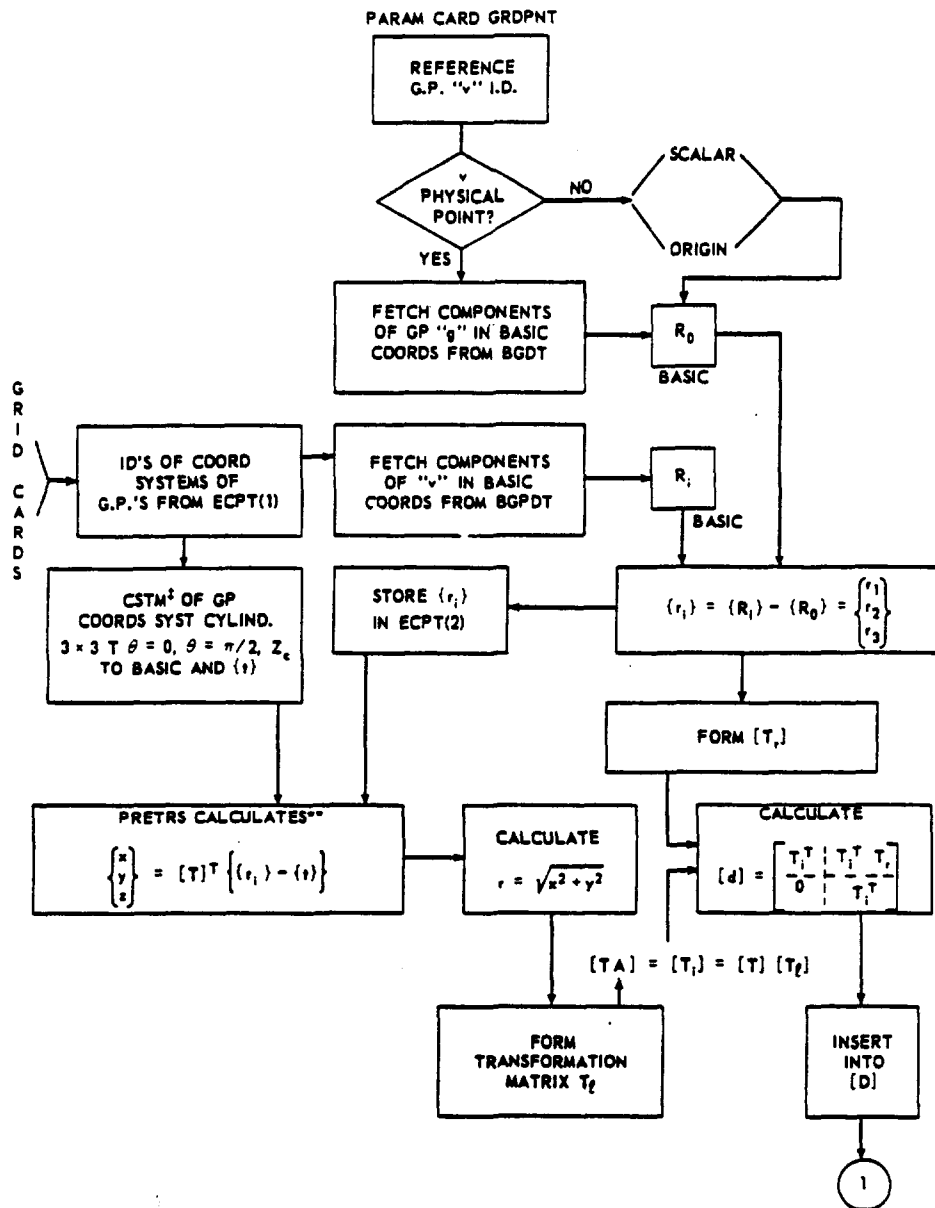


Figure 1.(a) Flowchart of GPWG Module Processing

Functional Module GPWG (Grid Point Weight Generator)

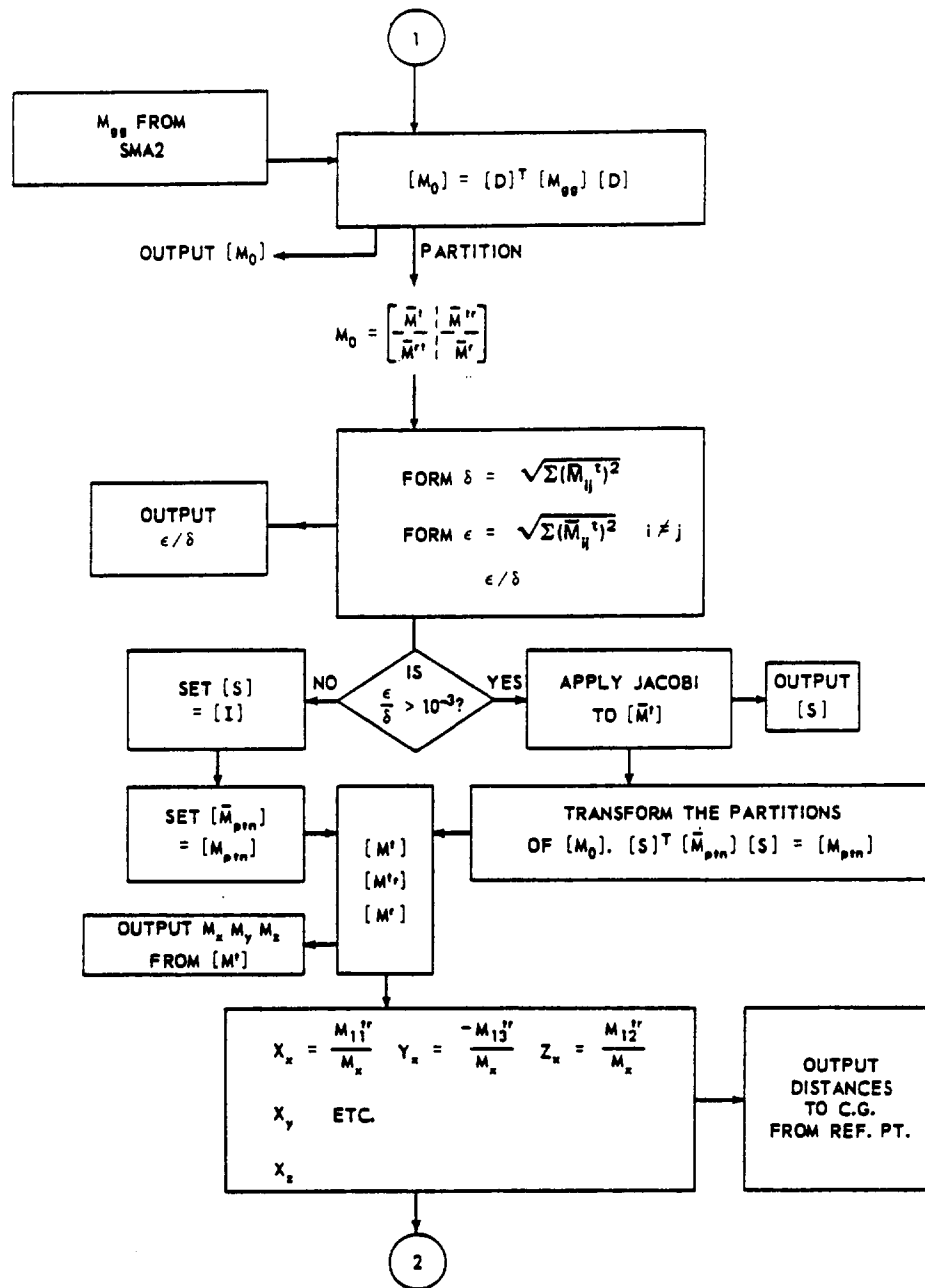


Figure 1.(a) Flowchart of GPWG Module Processing

MODULE FUNCTIONAL DESCRIPTIONS

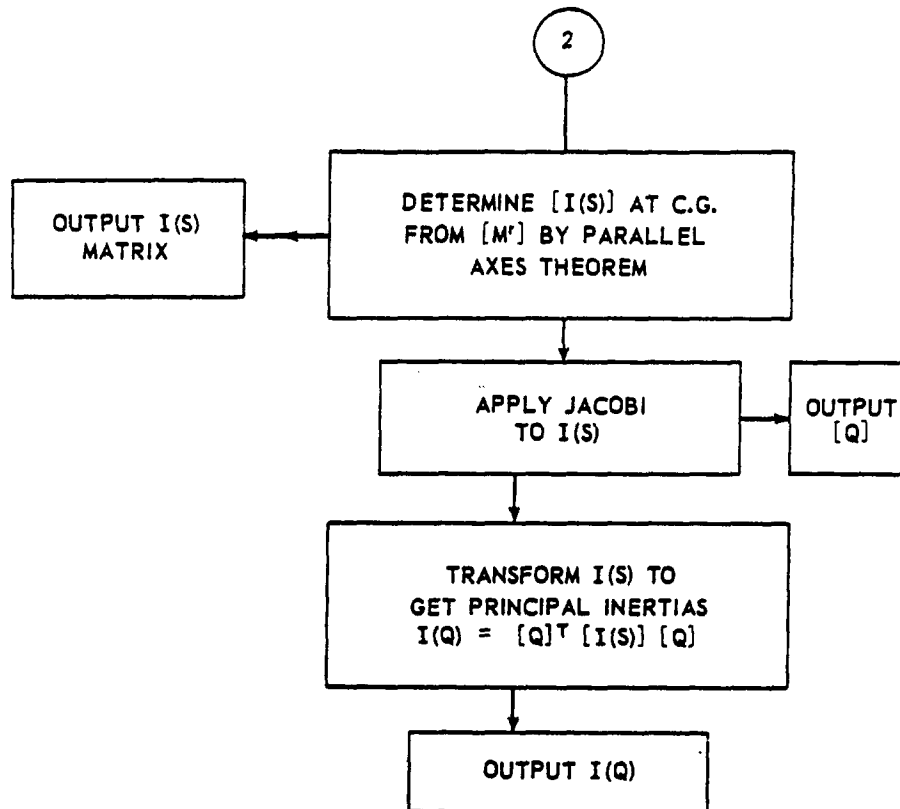


Figure 1.(a) Flowchart of GPWG Module Processing

4.30 FUNCTIONAL MODULE SMA3 (STRUCTURAL MATRIX ASSEMBLER - PHASE 3)

4.30.1 Entry Point: SMA3

4.30.2 Purpose

To generate the final stiffness matrix, $[K_{gg}]$, by generating a matrix of order g for each general element in the model, and successively adding this matrix to $[K_{gg}^x]$, the stiffness matrix exclusive of general elements.

4.30.3 DMAP Calling Sequence

SMA3 GEI,KGGX/KGG/V,N,LUSET/V,N,NØGENL/V,N,NØSIMP \$

4.30.4 Input Data Blocks

GEI - General Element Input.

KGGX - Partition of stiffness matrix exclusive of general elements - g set.

Note:

1. GEI cannot be pre-purged.
2. KGGX may be pre-purged. This implies that the model consists entirely of general elements (i.e., there are no simple elements).

4.30.5 Output Data Blocks

KGG - Partition of stiffness matrix - g set. Contains contributions from all elements in the model, including general elements.

Note: KGG may not be pre-purged.

4.30.6 Parameters

LUSET - Input-integer-no default value. LUSET is the total number of degrees of freedom in the g displacement set. It is the order of the $[K_{gg}^x]$ and $[K_{gg}]$ matrices.

NØGENL - Input-integer-no default value. NØGENL is the number of general elements in the GEI data block.

NØSIMP - Input-integer-no default value. If NØSIMP < 0, $[K_{gg}^x]$ does not exist, i.e., all elements of the model are general elements. If NØSIMP \leq 0, $[K_{gg}^x]$ does exist.

4.30.7 Method

4.30.7.1 Mathematical Considerations

Two methods are available to generate the stiffness matrix for the general element. The method used depends on the type of input supplied.

A. Flexibility Method

Two matrices can be used to form a stiffness matrix for each general element in the GEI data block: a flexibility influence coefficient matrix, $[Z]$, and a rigid body matrix $[S]$. The former must be present and must be non-singular; the latter may or may not be present. The set degrees of freedom (scalar index numbers) used by $[Z]$ is designated the " u_i " set; the set of degrees of freedom used by $[S]$ is designated the " u_d " set. Call the length of the u_i set m , and call the length of the u_d set n . $[Z]$ is m by m , and $[S]$ is m by n .

For each general element in the model, the stiffness matrix corresponding to the general element, $[K^{ge}]$, is made up of four partitions if the u_d set exists. They are:

$$[K_{ii}] = [Z]^{-1} , \quad (1)$$

$$[K_{id}] = -[Z]^{-1}[S] , \quad (2)$$

$$[K_{di}] = -[S]^T[Z]^{-1} = [K_{id}]^T , \quad (3)$$

$$[K_{dd}] = [S]^T[Z]^{-1}[S] . \quad (4)$$

The four matrices must be merged such that 1):

$$[K^{ge}] = \begin{bmatrix} K_{ii} & | & K_{id} \\ \hline K_{di} & | & K_{dd} \end{bmatrix} ; \quad (5)$$

and 2) the rows and columns of $[K^{ge}]$ must correspond to a merged list of both u_i and u_d coordinates in order of ascending scalar index numbers.

If the u_d set does not exist, then

$$[K^{ge}] = [K_{ii}] = [Z]^{-1} . \quad (6)$$

B. Stiffness Method

The stiffness method is identical to the method used for the flexibility matrix except that the stiffness matrix $[K]$ replaces $[Z]^{-1}$ in the above equations.

4.30.7.2 Initialization

The GEI data block is opened, and the header record is skipped. It is determined whether the number of general elements, f , is even or odd. This is done to insure that the result of the final matrix addition,

$$[K_{gg}] = [K^{rs}] + [K^{gef}] \quad , \quad (7)$$

where

$$[K^{rs}] = [K_{gg}^x] + \sum_{i=1}^{f-1} [K^{gei}] \quad , \quad (8)$$

and $[K^{gef}]$ is the final general element matrix, will be written on the output data block, KGG.

If $[K_{gg}^x]$ does not exist, and there is only one general element, then the GINØ file number (201) for the KGG data block is stored in IFA, the variable which contains the GINØ file number of the file onto which the current $[K^{ge}]$ matrix will be packed.

The principal logic of the module driver, SMA3, is carried out in a loop in which, during each pass of the loop, a $[K^{ge}]$ matrix is generated and added, using subroutine SSG2C, to the running sum matrix, $[K^{rs}]$.

4.30.7.3 Generation and Addition of a General Element Matrix

The steps involved in the principal loop of the program are as follows:

1. The loop counter is incremented.
 2. The first three words of the next logical record are read from GEI: the element id; the length of the u_i set, m ; and the length of the u_d set, n .
 3. The matrix control blocks for the scratch files IFB and IFC are interchanged provided that:
 - a. This is not the first pass through the loop;
 - b. this is not the second pass through the loop and the number of general element is odd and there are only general elements in the model;
- and
- c. this is not the second time through the loop and $[K_{gg}^x]$ exists.

MODULE FUNCTIONAL DESCRIPTIONS

4. It is determined whether the orders of the [Z] and [S] matrices are such that the in-core matrix routines GMMATD and INVERD can be used. This is accomplished as follows. Define

$$p = 2(m + n + m^2 + n^2 + 2mn) \quad , \quad (9)$$

$$q = 2(m + n + m^2) + 3m \quad , \quad (10)$$

and

$$r = \max(p, q) \quad . \quad (11)$$

p is the number of computer words needed to store: 1) the u_i and u_d sets in two different sorts; 2) the double precision m by m [K] or [Z] matrix; 3) the double precision n by n $[S]^T[K][S]$ or $[S]^T[K][S]$ or $[S]^T[Z]^{-1}[S]$ matrix; and 4) the double precision m by n [S] matrix and the double precision m by n $[K][S]$ or $[Z]^{-1}[S]$ matrix.

q is the number of computer words needed to store: 1) the u_i and u_d sets in two different sorts; 2) the double precision m by m [K] or [Z] matrix; and 3) 3m cells of scratch storage to be used by subroutine INVERD.

If r is less than the available amount of core, the in-core routine, SMA3A, is called to compute $[K^{ge}]$. Otherwise: 1) SMA3B generates [K] or [Z] and $-[S]^T$; 2) if [Z] is input, FACTØR decomposes [Z] into its triangular factors; 3) if Z is input, SSG3A computes $[Z]^{-1}$; 4) SSG2B computes either $-[S]^T[K]$ or $-[S]^T[Z]^{-1}$; 5) TRANP1 transposes $-[S]^T$; 6) SSG2B computes either $-[K][S]^T$ or $-[Z]^{-1}[S]^T$; 7) SSG2B computes either $[S]^T[K][S]$ or $[S]^T[Z]^{-1}[S]$; and 8) SMA3C builds the final $[K^{ge}]$ matrix of order g by g.

5. The matrix $[K^{ge}]$ having been generated as in step 4, SSG2C is called to add $[K^{ge}]$ to $[K^{rs}]$.

4.30.8 Subroutines

Utility routines GMMATD, INVERD, FACTØR, SSG3A, SSG2B, TRANP1 and SSG2C are used in this module.

4.30.8.1 Subroutine Name: SMA3A

1. Entry Point: SMA3A

2. Purpose: To build a g by g general element matrix, $[K^{ge}]$, using the in-core matrix routines GMMATD and INVERD.

3. Calling Sequence: CALL SMA3A (MCBA)

COMMON/GENELY/ - see description below (Section 4.30.9.2).

MCBA - The matrix control block corresponding to $[K^{ge}]$. Word 1 is input; words 2 through 7 are output.

4. Method: The u_i set and the u_d set (if present) of scalar index numbers are read into core, and a list L is formed of length $m + n$, such that $L(k) = \ell$ implies the ℓ^{th} entry of the string: $\{u_{i1}, u_{i2}, \dots, u_{im}, u_{d1}, u_{d2}, \dots, u_{dn}\}$ is the k^{th} smallest. The m^2 single precision elements of $[K]$ or $[Z]$ are read and stored at double precision locations. $[Z]^{-1}$, if required, is computed using INVERD. $[S]$, if present, is read and stored at double precision locations. GMMATD is called twice to compute either $[K][S]$ and $[S]^T[K][S]$ or $[Z]^{-1}[S]$ and $[S]^T[Z]^{-1}[S]$. The elements of $[K^{ge}]$, as defined in Equation 5, are output to the GINØ file corresponding to MCBA(1) with non-zero terms in the row and column positions specified by the u_i and u_d sets. The list L determines the sequence of elements to be output for any one column.

4.30.8.2 Subroutine Name: SMA3B

1. Entry Point: SMA3B

2. Purpose: To create $[K]$ or $[Z]$ and $[S]$ from the GEI data.

3. Calling Sequence: CALL SMA3B (IFLAG, IZK)

COMMON/GENELY/ - see description below (Section 4.30.9.2).

IFLAG - IFLAG = -1 implies $[S]$ does not exist. - integer - output.
IFLAG = 1 implies $[S]$ exists.

IZK - IZK = 1 if $[Z]$ is input. - integer - output.
IZK = 2 if $[K]$ is input.

MODULE FUNCTIONAL DESCRIPTIONS

4. Method: The GEI data block is read for the row numbers and non-zero terms of $[K]$ or $[Z]$. These are output in standard NASTRAN matrix format by subroutine BLDPK. $-[S]^T$ is generated in a similar manner. ($-[S]^T$ is created rather than $[S]$ for computational ease in subsequent calculations - see paragraph 4 in Section 4.30.7.3 above).

THIS PAGE HAS BEEN LEFT BLANK INTENTIONALLY.

4.30.8.3 Subroutine Name: SMA3C

1. Entry Point: SMA3C

2. Purpose: To create the $[K^{ge}]$ matrix from $[K]$, $-[K][S]$, $-[S]^T[K]$, and $[S]^T[K][S]$ or $[Z]^{-1}$, $-[Z]^{-1}[S]$, $-[S]^T[Z]^{-1}$ and $[S]^T[Z]^{-1}[S]$.

3. Calling Sequence: CALL SMA3C (IFLAG,KE)

COMMON/GENELY/ - see description below (Section 4.30.9.2).

COMMON//LUSET - Size of problem.

IFLAG is as described in SMA3B.

KE - matrix control block for $[K^{ge}]$ - integer - input/output.

4. Method: A matrix of g size is created from $[K]$, $-[K][S]$, $-[S]^T[K]$ and $[S]^T[K][S]$ or $[Z]^{-1}$, $-[Z]^{-1}[S]$, $-[S]^T[Z]$ and $[S]^T[Z]^{-1}[S]$ with the non-zero terms in the row and column positions specified by the u_i and u_d lists. This matrix can be added to the existing $[K^{rs}]$ to reflect the stiffness terms of this general element.

4.30.8.4 Block Data Subprogram: SMA3BD

Purpose: To initialize the GINØ file numbers and GINØ options indicators in /GENELY/, which is discussed below.

4.30.9 Design Requirements

4.30.9.1 Open Core Design

The open core common block for the module driver SMA3 and subroutine SMA3A is defined by the following FORTRAN statements:

1. DOUBLE PRECISION DQ(1)
2. INTEGER IQ(1)
3. DIMENSION Q(1)
4. COMMON /GENELX/ Q
5. EQUIVALENCE (IQ(1),DQ(1),Q(1))

SMA3 uses open core only for one GINØ buffer, which is reserved for the GEI data block while SMA3A, SMA3B, or SMA3C is executing, and which is reserved for use by SSG2C when this routine adds $[K^{ge}]$ to $[K^{rs}]$ at the end of the principal loop in the driver.

FUNCTIONAL MODULE SMA3 (STRUCTURAL MATRIX ASSEMBLER - PHASE 3)

SMA3A uses low order open core as outlined in paragraph 4 in section 4.30.7 above.

The open core for subroutine SMA3B is defined at /SMAB3/ and is used for two GINØ buffers in high order open core.

The open core for subroutine SMA3C is defined at /SMAC3/ and is used for: 1) the u_i and u_d sets in low order open core and 2) six GINØ buffers in high order open core.

4.30.9.2 Common Storage Requirements

The common block /GENELY/ is used for: 1) GINØ file numbers; 2) GINØ option indicators; 3) matrix control blocks; and 4) zero pointers to sub-arrays in /GENELX/ when SMA3A executes. It is defined as follows:

COMMON /GENELY/ IFGEI,IFKGGX,IFØUT,IFA,IFB,IFC,IFD,IFE,IFF,INRW,ØUTRW,CLSRW,CLSNRW,EØR,
NEØR,MCBA(7),MCBB(7),MCBC(7),MCBE(7),MCBF(7),MCBKGG(7),IUI,IUD,IZI,IS,IZIS,ISTZIS,IBUFF3(3),
LEFT

<u>Variable</u>	<u>Definition</u>
IFGEI,IFKGGX,IFØUT	GINØ file numbers for the two input data blocks and the output data block respectively.
IFA	GINØ file number for the current $[K^{ge}]$ being computed.
IFB,IFC	GINØ file numbers for $[K^{rs}]$ and $[K^{rs}] + [K^{ge}]$ matrices. They are "flip-flopped" such that $IFC = IFØUT$ for the final matrix addition.
IFD,IFE,IFF	GINØ file numbers for scratch files which are used in subroutine SMA3C.
INRW,ØUTRW,CLSRW,CLSNRW,EØR,NEØR	GINØ option indicators as defined in section 4.27.9.3.
MCBA,MCBB,...,MCBF,MCBKGG	Matrix control blocks for the matrices corresponding to IFA, IFB, ..., IFF, and IFKGGX.
IUI,IUD,IZI,IS,IZIS,ISTZIS	Zero pointers to the sub-arrays in /GENELX/ corresponding to: 1) u_i set; 2) u_d set; 3) $[Z]^{-1}$; 4) $[S]$; 5) $[Z]^{-1}[S]$ and 6) $[S]^T[Z]^{-1}[S]$. Note that IZI, IZIS, ISTZIS are zero pointers into double precision arrays.
IBUFF3(3)	Three word buffer which contains the general element id, m and n.
LEFT	The number of computer words currently remaining in /GENELX/.

4.30.9.3 Arithmetic Considerations

All floating point arithmetic operations are carried out in double precision.

4.30.10 Diagnostic Messages

In SMA3A, system fatal error 2028 can occur. See section 6 of the User's Manual for details.

4.31 FUNCTIONAL MODULE GP4 (GEOMETRY PROCESSOR - PHASE 4)

4.31.1 Entry Point: GP44.31.2 Purpose

GP4 assembles the various displacement sets and builds the displacement set definition table (USET). Additionally, for statics problems, GP4 analyzes subcases based on single-point and multi-point constraint sets, and sets parameters to control execution of the Rigid Format.

4.31.3 DMAP Calling Sequence

```
GP4    CASECC,GEOM4,EQEXIN,GPD,T,BGPD,T,CSTM / RG,YS, { HUSET } , { ASET } / V,N,LUSET /
        USET      HASET
        V,N,MPCF1 / V,N,MPCF2 / V,N,SINGLE / V,N,ØMIT / V,N,REACT / V,N,NSKIP / V,N,REPEAT /
        V,N,NØSET / V,N,NØL / V,N,NØA / C,N,SSID $
```

4.31.4 Input Data Blocks

CASECC - Case Control Data Table.
 GEØM4 - Displacement set definition.
 EQEXIN - Equivalence between external grid or scalar and internal numbers.
 GPD,T - Grid Point Definition Table.
 BGPD,T - Basic Grid Point Definition Table.
 CSTM - Coordinate System Transformation Matrix Table.

Note: Only GEØM4 and CSTM may be purged.

4.31.5 Output Data Blocks

RG - Multipoint and rigid element constraint equations matrix
 YS - Constrained displacement vector(s) set.
 USET - Displacement set definition table.
 HUSET - Temperature set definition table for heat problems.
 ASET
 HASET - Not used.

Note: YS may be purged.

4.31.6 Parameters

LUSET - Input-integer-no default. Degrees of freedom in the g-displacement set.
 MPCF1 - Output-integer-no default. +1 if the current subcase contains multipoint constraints, -1 otherwise.

MODULE FUNCTIONAL DESCRIPTIONS

- MPCF2 - Output; integer, no default. +1 if the current subcase contains a different multipoint constraint set from the last subcase, -1 if no new multipoint constraint set or no multipoint constraints in the current subcase.
- SINGLE - Output, integer, no default. +1 if the current subcase contains single-point constraints, -1 otherwise.
- ØMIT - Output, integer, no default. +1 if the model contains omitted coordinates, -1 otherwise.
- REACT - Output, integer, no default. +1 if the model contains supports, -1 otherwise.
- NSKIP - Input and output, integer, default = 0. Number of records to skip to reach the first record in the Case Control Data Block for the next subcase. (NSKIP = 0 for the first subcase).
- REPEAT - Output, integer, no default. -1 if the current subcase is the last subcase in the problem, +1 otherwise.
- NØSET - Output, integer, no default. -1 if MPCF1 = -1 and SINGLE = -1 and ØMIT = -1 and REACT = -1, +1 otherwise.
- NØL - Output, integer, default = +1. -1 if all degrees of freedom in the model belong to dependent displacement sets (i.e., no degree of freedom belongs to an independent set), +1 otherwise.
- NØA - Output, integer, default = +1. -1 if MPCF1 = -1 and SINGLE = -1 and ØMIT = -1, +1 otherwise.
- SSID - Input, integer, default = 0. Reserved for future use.

4.31.7 Method

GP4 first determines the multi-point constraint (MPC) set and the single-point constraint (SPC) set for the current subcase by reading CASECC. For the first subcase, these sets are extracted from the first non-header record of CASECC. For subsequent subcases, GP4 examines each data record on CASECC that follows the data record used for the previous subcase until a different MPC or SPC set is found. After the MPC and SPC sets are known, GP4 reads EQEXIN to set up a table of external grid or scalar numbers and the corresponding internal number, SIL value, and

code (1 for grid point; 2 for scalar point) associated with each external grid of scalar number. See Figure 1 for where these values are kept in open core.

If rigid elements exist, GP4 calls subroutine CRIGGP to compute the coefficients of the constraint equations directly. See The NASTRAN Theoretical Manual, Section 3.5.6 for a discussion of the methods used to calculate the coefficients of the constraint equations. After calculating the coefficients, CRIGGP writes a two-word entry onto RG for each independent degree of freedom that affects a dependent degree of freedom. The first word is a coded column-and-row number containing the SIL value of the independent degree of freedom multiplied by $2^{**}16$ and the result added to the SIL value of the dependent degree of freedom. This storage algorithm implies a 65,535 ($2^{**}16 - 1$) degree of freedom limitation for any NASTRAN problem. The second word of the two-word entry contains the associated coefficient. These two-word entries are written as one logical record on RG. A second logical record is written by CRIGGP containing the list of dependent SIL values for the rigid elements. After the RG table is written, CRIGGP returns control to GP4.

GP4 determines whether the MPC set is on a MPCADD card image and, if it is not, a simulated MPCADD card image is defined referencing the MPC set. GP4 then reads all MPC card images referenced by the MPCADD card image and writes, in open core, two-word entries containing a coded column-and-row number, and the corresponding coefficient in the same manner as described for rigid elements above (see Figure 1). However, for MPC's the coefficients are obtained directly from the MPC card image rather than being calculated as in the case of rigid elements. A list of all dependent SIL values associated with MPC's is saved in the upper end of open core. Once all MPC card images have been processed by GP4, the list of two-word entries and dependent SIL values is expanded to include those values saved on RG that were calculated by CRIGGP for the rigid elements. The dependent SIL values are then sorted, thus forming the U_m set, and written on SCRI. Based on the two-word entries described above, GP4 writes the RG matrix.

GP4 then reads DMIT (U_0) card images. The grid and component numbers are converted to SIL values, sorted, and written to SCRI. Similarly, SUPORT (U_p) card images are processed. Figure 2 shows open core allocation for processing DMIT and SUPORT card images.

GP4 then reads GPDIT and forms a list of single-point constraints (U_{sg}) that are identified on GRID bulk data card images. The list is written to SCRI. Next, GP4 determines whether the SPC set is on an SPCADD card image and, if it is not, a simulated SPCADD referencing the SPC set is

MODULE FUNCTIONAL DESCRIPTIONS

generated. SPCADD, SPC, and SPC1 bulk data card images identify the sets of single-point constraints used as boundary points (U_{sb}) and include any possible constrained displacement values (Y_s). GP4 reads all SPC and SPC1 card images referenced by the SPCADD card image, converts the grid and component numbers to SIL values and writes the SIL values and their corresponding displacement values on SCRI. The non-zero Y_s values obtained in this step form a packed vector (the constrained displacement vector Y_s) that is packed relative to the U_s set. Figure 3 shows open core allocation for processing SPC card images.

In a similar manner as described for the OMIT card images, the ASET and ASET1 card images are processed by GP4. After processing the ASET and ASET1 card images, SCRI contains a logical record for each of the U_m , U_o , U_r , U_{sg} , U_{sb} , and U_a sets, if data exists for each of these sets.

GP4 forms a list of all SILs in open core and creates a mask for each SIL of bits that are turned on or off for various displacements sets by examining each logical record of SCRI (see The NASTRAN Programmer's Manual Section 2.3.13). This list is written by GP4 to USET. While reading the U_{sb} set from SCRI, GP4 also writes Y_s . The following conventions are followed by GP4 for degrees of freedom not specifically included or omitted:

1. If ASET or ASET1 card images are present, unspecified degrees of freedom are omitted.
2. If ASET or ASET1 card images are not present and OMIT or OMIT1 card images are present, unspecified degrees of freedom are included in the analysis set.
3. If there are no ASET, ASET1, OMIT, or OMIT1 card images present, unspecified degrees of freedom are included in the analysis set.
4. If both ASET or ASET1 card images and OMIT or OMIT1 card images are present, unspecified degrees of freedom are omitted.

The next operation is to process each degree of freedom in USET to insure that the displacement set definitions are consistent. The governing rule is that each degree of freedom may belong at most to one dependent subset. If any inconsistent definitions are found, they are written on SCRI. When each point has been analyzed, and if inconsistent definitions are found, then SCRI is read, and, for each entry, an error message is queued. After all error messages have been queued, GP4 abnormally terminates.

The final operation for GP4 is to process any SPCD card images. GP4 reads USET into open core and establishes a table of LOAD set IDs by examining all records of CASECC up to the current

FUNCTIONAL MODULE GP4 (GEOMETRY PROCESSOR - PHASE 4)

subcase record. YS is then read into open core. See Figure 4 for open core allocation when processing SPCD card images. GP4 reads the SPCD card images specified by the LOAD set IDs and writes two-word entries containing those SILs to SCR2, and corresponding constrained displacement values that are modified by the SPCD card. There will be one logical record for each LOAD set on SCR2. Once all LOAD sets have been processed, GP4 reads SCR2, updates the YS displacement vector, and rewrites YS. There will be a column written to YS for each load set specified.

When axisymmetric elements are used, the total number of degrees of freedom for the problem and the number of MPC's and SPC's are greater than one might expect by examination of the input data. Table 1 shows the number of GRID, SPC, and MPC card images that are generated by the axisymmetric cards MPCADD, MPCAX, OMITAX, POINTAX, RINGAX, SECTAX, SPCADD, SPCAX, and SUPAX. As can be seen from Table 1, axisymmetric problems involving many harmonics generate many GRID, SPC, and MPC card images and by examination of Figures 1, 2, 3, and 4, the core requirements of GP4 go up accordingly. Users of axisymmetric cards should be familiar with how NASTRAN processes these cards (see The NASTRAN Programmers Manual Section 4.6).

MODULE FUNCTIONAL DESCRIPTIONS

Relative address to /GP4CØR/

1	External Grid or Scalar Number	}	Repeated for each grid or scalar point in the model
2	Internal Number		
	⋮		
KN+1	External Grid or Scalar Number	}	Repeated for each grid or scalar point in the model
	10*SIL + Code		
	⋮		
KM	Table of sorted SILs		
KM + KN/2 + 1	Integer value of the Number of Degrees of Freedom in the g-Displacement set (LUSET)		
KNKL1	MPCSET ID		
IMPC	Coded column-and-row number	}	Repeated for each independent SIL that affects a dependent SIL
	Coefficient		
	⋮		
	⋮		
	List of Dependent SIL Values		
4 I/O Buffers			

Figure 1. Open core allocation for GP4 when processing MPC card images.

FUNCTIONAL MODULE GP4 (GEOMETRY PROCESSOR - PHASE 4)

Relative address to /GP4CØR/

1	External Grid of Scalar Number		}	Repeated for each grid or scalar point in the model
2	Internal Number			
	⋮			
KN+1	External Grid or Scalar Number 10*SIL + Code		}	Repeated for each grid or scalar point in the model
	⋮			
KM	Table of sorted SILs			
KM + KN/2 + 1	Integer Value of the Number of Degrees of Freedom in the g-Displacement set (LUSET)			
KNKL1	SIL Values for ØMIT and ØMIT1 Card Images	SIL Values for SUPØRT Card Images	SIL Values for ASET and ASET1 Card Images	
	4 I/Ø Buffer			

Figure 2. Open core allocation for GP4 when processing ØMIT, SUPØRT, and ASET card images.

MODULE FUNCTIONAL DESCRIPTIONS

Relative address to /GP4CØR/

1	External Grid or Scalar Number	}	Repeated for each grid or scalar point in the model
2	Internal Number		
	⋮		
KN+1	External Grid or Scalar Number 10*SIL + Code	}	Repeated for each grid or scalar point in the model
	⋮		
KM	Table of sorted SILs		
KM + KN/2 + 1	Integer Value of the Number of Degrees of Freedom in the g-Displacement set (LUSET)		
KNKL1	SIL Value of SPCs Constrained Displacement for SPC	}	Repeated for all SPCs
4 I/O Buffers			

Figure 3. Open core allocation for GP4 when processing SPC card Images.

FUNCTIONAL MODULE GP4 (GEOMETRY PROCESSOR - PHASE 4)

Relative address to /GP4CØR/

1	External Grid or Scalar Number	}	Repeated for each grid or scalar point in the model
2	Internal Number		
	⋮		
KN+1	External Grid or Scalar Number 10*SIL + Code	}	Repeated for each grid or scalar point in the model
	⋮		
KM+1	Table of sorted SILs		
KM + KN/2 + 1	Integer Value of the Number of Degrees of Freedom in the g-Displacement Set (LUSET)		
KNKL1	USET - One Coded Word for Each SIL		
ILOAD	Load Set ID Record Number in SCR2 containing Displacement Value	}	Repeated for all LOAD sets
	⋮		
IØYS	Old YS Constrained Displacement Vector		
INYS	New YS Constrained Displacement Vector		
4 I/Ø Buffers			

Figure 4. Open core allocation for GP4 when processing SPCD card images.

MODULE FUNCTIONAL DESCRIPTIONS

Axisymmetric Card	Cards Generated
MPCADD (open ended)	2 MPCADD cards
MPCAX (open ended)	2 MPCADD cards 1 MPC card
ØMITAX	1 ØMIT card
POINTAX	N x 12 (3-word groups) MPC cards 1 GRID card
RINGAX	N x 2 SPC cards N GRID cards
SECTAX	N x 6 (3-word groups) MPC cards 1 GRID card
SPCADD (open ended)	2 SPCADD cards
SPCAX	2 SPCADD cards 1 SPC card
SUPAX	1 SUPØRT card

Note: N = HARM + 1 where HARM is obtained from the AXIC card.

Table 1. Card images generated by axisymmetric cards.

4.31.7.1 Definitions of the Sets Defined in USET

- u_g - All structural degrees of freedom defined by grid and scalar points.
- u_m - Dependent coordinates used in the multipoint constraint equations (defined as the first degree of freedom of an MPC card) and rigid elements.
- u_n - All structural degrees of freedom except u_m .
- u_s - All fixed points. The u_{sg} points are defined by the GRID cards and have displacements of zero. The u_{sb} points are defined by the SPC cards and may have constrained displacements.
- u_f - All degrees of freedom in the structure except u_m and u_s .
- u_o - These are "omitted coordinates" defined by OMIT and OMIT1 cards. In statics, the structural matrix is partitioned, and these degrees of freedom are solved separately. In dynamics, the displacements of these points are approximated by their static displacements under mass loads.
- u_a - These are the unconstrained degrees of freedom of the system. They include rigid body modes in dynamics.
- u_r - These are fictitious supports defined by the SUPORT cards. In dynamics and inertia relief, the elastic displacements are measured relative to these points.
- u_l - This set includes all degrees of freedom not defined by the u_m , u_s , u_o and u_r points. The stiffness matrix defined by these points is used for the solution of displacements versus loads.

4.31.8 Subroutines

4.31.8.1 Subroutine Name: GP4PRT

1. Entry Point: GP4PRT
2. Purpose:
 - a. Prints, at user request via DIAG 21, a list of degrees of freedom. For each degree of freedom, an indication is made identifying the sets to which it belongs.
 - b. Prints, at user request via DIAG 22, the contents of selected displacement sets. For each set, a list of all degrees of freedom belonging to the set is given.
3. Calling Sequence: CALL GP4PRT (BUF)

MODULE FUNCTIONAL DESCRIPTIONS

4. Method: The DIAG flags are tested and local variables set. Table EQEXIN is then read into open core and sorted. If DIAG 21 is on, table USET (already in open core) is examined and the external degree of freedom is extracted from EQEXIN and printed along with the set indications. If DIAG 22 is on, transpose process takes place.

4.31.8.2 Subroutine Name: SCALEX

1. Entry Point: SCALEX
2. Purpose: Decodes packed component codes.
3. Calling Sequence: CALL SCALEX (I,C,L)
I = Value to be decoded.
C = If non-positive, return after loading I into L(1).
L = Array into which the decoded values are placed.

4.31.8.3 Subroutine Name: FØRMGG - single precision FØRMG2 - double precision

1. Entry Point: FØRMGG
FØRMG2
2. Purpose: To form the [GG] matrix.
3. Calling Sequence: CALL FØRMGG (IG,JR,JD,IR,ID)
CALL FØRMG2 (IG,JR,JD,IR,ID)
IG = Start of row stored [GG] matrix -1.
JR = Start of [TA] matrix -1.
JD = Start of [TB] matrix -1.
IR = Start of BGPDT information for reference point.
ID = Start of BGPDT information for dependent point.

4.31.8.4 Subroutine Name: GP4BD

1. Entry Point: Blockdata
2. Purpose: Initializes COMMONs /GP4FIL/ and /GP4PRM/
COMMON/GP4FIL/CASECC,GEØMP,EQEXIN,SIL,GPDT,BGPDT,
CSTM,RGT,YS,USET,SCR1,SCR2,SCRF

where

CASECC	Input file number of CASECC
GEØMP	Input file number of GEØM4

FUNCTIONAL MODULE GP4 (GEOMETRY PROCESSOR - PHASE 4)

EQEXIN	Input file number of EQEXIN
SIL	Input file number of SIL (not used)
GPDT	Input file number of GPDT
BGPDT	Input file number of BGPDT
CSTM	Input file number of CSTM
RGT	Output file number of RG
YS	Output file number of YS
USET	Output file number of USET
SCR1	Scratch files
SCR2	
SCR3	

COMMON/GP4PRM/BUF, BUF1, BUF2, BUF3, BUF4, KNKL1, MULTI,
 TW016, MPCSET, N0G0, DUP, MCBYS, MCB,
 IRGD1, IRGD2, MPCAX1, MPCAX2, MPC, USGSET,
 SPCSET, SPC0LD, GP0INT, KN, MPCADD

where

BUF	Temporary storage array
BUF1	Indexes for I/O buffers in /GP4C0R/
BUF2	
BUF3	
BUF4	
KNKL1	Index to /GP4C0R/ for storage of SIL values
MULTI	Total number of MPCs (including RIGID elements)
TW016	2**16
MPCSET	MPC set ID
N0G0	Abnormal termination flag
DUP	Flag for dublicately defined grid point component
MCBYS	Matrix control block for YS
MCB	General purpose matrix control block
IRGD1	Flag for CRIGD1 elements
IRGD2	Flag for CRIGD2 elements
MPCAX1	Identification of MPCs generated by a MPCAX card
MPCAX2	
MPC	Identification numbers to be used by subroutine LOCATE to find MPC card images
USGSET	Flag for constrained points obtained from GRID card images
SPCSET	SPC set ID
SPC0LD	SPC set ID for previous subcase
GP0INT	Grid point ID
KN	Index to /GP4C0R/ for storage of external grid or scalar numbers and coded SIL (see Figure 2)
MPCADD	Identification number to be used by subroutine LOCATE to find MPCADD card images

MODULE FUNCTIONAL DESCRIPTIONS

4.31.8.5 Subroutine Name: CRIGGP

1. Entry Point: CRIGGP
2. Purpose: Generates coefficients of constraint equations for rigid elements CRIGD1, CRIGD2, and CRIGD3, and calls CRDRD or CRDRD2 (depending on precision) for the rigid element CRIGDR.
3. Calling Sequence:
CALL CRIGGP

4.31.8.6 Subroutine Name: CRDRD Single precision CRDRD2 Double precision

1. Entry Point: CRDRD
CRDRD2
2. Purpose: Generates coefficients of constraint equations for the rigid element CRIGDR
3. Calling Sequence:
CALL CRDRD(MU, INDCØM), RETURNS(n1,n2)
CALL CRDRD2(MU, INDCØM), RETURNS(n1,n2)

where

MU	Index into /GP4CØR/ to store dependent SILs
INDCØM	Dependent component of the dependent grid point
n1	Statement number to return to if the length of the CRIGDR is zero
n2	Statement number to return to if CRIGDR is not properly defined

4.31.9 Design Requirements

See Figures 1, 2, 3, and 4 for storage requirements.

4.31.10 Diagnostic Messages

The following messages may be issued by GP4:

2045, 2047, 2048, 2049, 2050, 2051, 2052, 2053, 2101, 2110, 2132, 2152, 2153, 2403, 2423, 3001, 3002, 3003, 3056, 3059, 3060, 3061, 3145, 3146, 3147, 3148, 3133, 3134

MODULE FUNCTIONAL DESCRIPTIONS

4.32 FUNCTIONAL MODULE GPSPC (GRID POINT SINGULARITY PROCESSOR)

4.32.1 Entry Point: GPSPC

4.32.2 Purpose

The GPST data block contains data on possible stiffness matrix singularities. These singularities may have been removed through the application of single or multipoint constraints. The GPSPC module checks each singularity against the list of constraints, and if the singularity is not thereby removed, writes a warning for the user and on user's option automatically constrains the singularity.

4.32.3 DMAP Calling Sequence

```
GPSPC  GPL,GPST,USET,SIL / ØGPST,USETC / V,N,NØGPST / V,Y,SINCØN / V,N,SINGLE  
      V,N,ØMIT / V,N,REACT / V,N,NØSET / V,N,NØL / V,N,NØA $
```

4.32.4 Input Data Blocks

GPL - Grid Point List

GPST - Grid Point Singularity Table

USET - Displacement Set Definitions Table

SIL - Scalar Index List

Note: No input data block can be purged.

4.32.5 Output Data Blocks

ØGPST - Tabular list of grid point singularities not removed by user. This data block will be processed by the ØFP (Output File Processor) module.

USETC - Displacement Set Definition Table with singularities constrained.

4.32.6 Parameters

NØGPST - Output, integer, default = 1. If positive, ØGPST was created.

SINCØN - Input-output, integer, default = -1. If SINCØN is negative on input, remaining singularities are automatically constrained. On output, same negative value if singularities existed, zero otherwise.

SINGLE	}	-	Input-output, integer, no default. See description of GP4 parameters of the same name in Section 4.31. Values are corrected only if singularities were constrained.
ØMIT			
REACT			
NØSET			
NØL			
NØA			

MODULE FUNCTIONAL DESCRIPTIONS

4.32.7 Method

USET is read into core. The USET data block contains one word for each degree of freedom in the structural model. This word identifies the displacement coordinate sets to which the coordinate belongs. Each entry in the GPST data block contains the order of the singularity and the scalar index numbers of the degrees of freedom involved. The logic of the algorithm depends on the order of the singularity. For each type, the logic is:

1. If order = 1, the contents of the GPST are:

- a. ORDER = 1
- b. Number of words following
- c. N_1
- d. N_2 } may not appear
- e. N_3 }

N_1 , N_2 , and N_3 are the scalar indices of the degrees of freedom which will remove the singularity if constrained. If the singularity is not removed, the GPST data is output. If SINCØN is negative the singularity is removed as an SPC.

2. If order = 2:

- a. ORDER = 2
- b. Number of words following
- c. N_{11}
- d. N_{21}
- e. N_{12} } may not appear
- f. N_{22} }
- g. N_{13} }
- h. N_{23} }

Each pair of indices identifies two degrees of freedom which cause the singularity. If both indices for any pair belong to the u_m or u_s set on USET, the singularity is removed. If only one of the degrees of freedom in a pair is constrained by a u_s or u_m coordinate, the singularity order is now ORDER = 1. The numbers listed are not unique, and more than one of the N_{ij} indices may belong to the u_s or u_m sets. Keeping the same sequence, the unconstrained scalar indices in each partially constrained pair is listed in the GPST in the form for ORDER = 1.

3. If order = 3:

- a. ORDER = 3
- b. Number of words following
- c. N_1
- d. N_2
- e. N_3

All three indices (N_1 , N_2 , and N_3) must belong to the u_s or u_m sets to remove the singularity. If one or two of the coordinates is constrained, the order is two or one, and the remaining scalar indices or index is listed in the ØGPST.

If any singularities are not constrained by SPC or MPC input, a message is given to this effect. Also, the SIL values must be converted to external grid point-component notation for user readability. The first time an unremoved singularity is detected, the SIL is read into core beneath the USET, and the GPL is placed beneath the SIL. The external grid point identification number is determined from the SIL and the GPL. Scalar points are differentiated from grid points. Data is output to the ØGPST in the following order: grid point ID, type, singularity order, and components. Subroutine GPSPD is called to create new USETC entries after the message is created for each point. When all entries in the GPST have been processed, the routine creates USETC if SINCØN is negative on input and singularities were removed. If no singularity existed, SINCØN is set to zero before exiting. Only the strongest combination (weakest stiffness) is constrained automatically. The parameters from GP4 are then corrected to reflect the new constraints. If no singularity existed, SINCØN is set to zero before exiting.

4.32.8 Subroutines

4.32.8.1 Subroutine Name: GPSPD

- 1. Entry Point: GPSPD
- 2. Purpose: To constrain the "strongest combination" singularities through the application of single point constraints.
- 3. Calling Sequence: CALL GPSPD (ISIL,INDEX)

ISIL - Three-word array of singularities

INDEX - Pointer to first SIL value of the point

MODULE FUNCTIONAL DESCRIPTIONS

COMMON/GPSPB/NSING,CORE(1)

NSING - Counter of singularities constrained

CORE - USET entries

4. Method: The bits for USB, US, UN, UP, and UG are reset for each USET entry pointed to by ISIL(i) + INDEX. Note that this change may override any omits, multipoint constraints, or support reactions applied by the user. The next singularity checked by GPSPC will use this corrected USET.

4.32.9 Design Requirements

Open core is defined at / GPSPB / .

COMMON / GPSPB /	
USET	} LUSET (Length of USET)
SIL	} LSIL+1 (Length of SIL+1) Last cell has LUSET stored
GPL	} LGPL (Length of GPL)
GPSP Buffer	} GINØ buffer
GPSP Buffer	} GINØ buffer
Scratch Buffer- USET,GPL,SIL	} GINØ buffer

4.32.9.1 Allocation of Core Storage

If no singularities exist, USET plus two GINØ buffers must be held in core. If singularities exist, USET, SIL, GPL plus three GINØ buffers must be held in core.

4.32.10 Diagnostic Messages

The following diagnostic messages may occur: 3007 (if the GPST does not contain legal SIL numbers, indicating a programming error); 3008 (if the core storage requirements given in Section 4.32.9.1 are not met).

FUNCTIONAL MODULE MCE1 (MULTIPOINT CONSTRAINT ELIMINATOR - PHASE 1)

4.33 FUNCTIONAL MODULE MCE1 (MULTIPOINT CONSTRAINT ELIMINATOR - PHASE 1)

4.33.1 Entry Point: MCE1

4.33.2 Purpose

MCE1 extracts the $[G_{m1}]$ matrix for rigid elements and matrix $[R_g]$ for multipoint constraints from table RG. MCE1 partitions $[R_g]$ into $[R_m]$ and $[R_n]$ and then solves the matrix equation $[R_m][G_{m2}] = -[R_n]$ for $[G_{m2}]$. MCE1 then forms matrix $[G_m]$ as a combination of $[G_{m1}]$ and $[G_{m2}]$.

4.33.3 DMAP Calling Sequence

MCE1 USET,RG/GM \$

4.33.4 Input Data Blocks

USET - Displacement set definitions table.

RG - Multipoint constraint equations and rigid element data table.

Note: Neither USET nor RG may be purged.

4.33.5 Output Data Blocks

GM - Multipoint constraint transformation matrix - m set.

Note: GM may not be purged.

4.33.6 Parameters

None

4.33.7 Method

The first time MCE1A is called from MCE1, MCE1A determines if multipoint constraint equations were used. If they were not control returns to MCE1. If multipoint constraint equations were used, the RG file is opened. If rigid elements are also present, it is necessary to form a new USET file on scratch file, SCR3, which will contain only the dependent SIL values for multipoint constraints in the UM set. This is accomplished by transferring the data currently on USET to SCR3. In the process, the UM set for rigid elements is read from RG and whenever the corresponding location is found on USET, the value is changed to represent the UG set when it is written on SCR3.

MODULE FUNCTIONAL DESCRIPTIONS

If rigid elements are not used, the USET table can be used as it is.

The RG table is then positioned to read the dependent set for multipoint constraints. These SIL values are read and stored in core. A RG matrix for multipoint constraints is formed and written on file SCR2 in standard packed matrix form. The next operation of MCE1 is to partition $[R_g]$ into $[R_m]$ and $[R_n]$. MCE1A performs this operation by initializing /PARMEG/ and calling PARTN (see Section 3.5.6 for PARTN details).

The next operation of MCE1 is to solve the matrix equation

$$[R_m] [G_m] = -[R_n] \quad (1)$$

for $[G_m]$. If $[R_m]$ is diagonal, the operation is straightforward. In this case MCE1D is called. The terms $-r_{ij}$ are stored in core, the $[R_n]$ matrix is read interpretively by INTPK, and the terms of $[G_m]$ are formed from the equation

$$g_{ij} = - \frac{r_{ij}}{r_{ii}}, \quad (2)$$

where the terms in the numerator belong to $[R_n]$ and those in the denominator belong to $[R_m]$. If $[R_m]$ is not diagonal, Equation 1 is solved by decomposition and forward-backward substitution. In this case, MCE1B is called. MCE1B performs an unsymmetric decomposition of $[R_m]$ by initializing /DCOMPX/ and calling DECØMP. MCE1C is then called by MCE1. MCE1C performs a forward-backward substitution to solve for $[G_m]$ by initializing /GFBSX/ and calling GFBS. See Section 3.5.15 and 3.5.19 for further details on DECØMP and GFBS, respectively.

The second time MCE1A is called from MCE1, the GM data for rigid elements is read into core from the RG table and sorted on the coded column-row number. This sorted list is written on scratch file SCR2. If multipoint constraint equations were also used, the GM matrix on SCR1 must be unpacked, the rows expanded to the full UM set, the corresponding coefficients for rigid elements added to it, and finally repacked on file GM. To accomplish this, the dependent SIL values for rigid elements are read from file RG. Each SIL value is assigned three consecutive storage locations. The first location is used to store the SIL value, the second location is used to store its position in the list, and the third location will later contain its position in the merged UM list. The dependent SIL values for the multipoint constraint set are read into core and stored as above except that the second location is its position in the original list plus 65536 (10000 in hexadecimal). The two lists are merged by sorting on the SIL number. The

FUNCTIONAL MODULE MCE1 (MULTIPOINT CONSTRAINT ELIMINATOR - PHASE 1)

third location is filled using this order. The merged list is separated into the two original lists again by sorting on the value in the second location. The expanded GM matrix is formed using the new row numbers assigned after the merge and stored in the third location of the above table. A column of the GM matrix for multipoint constraints is unpacked and expanded to the full UM set. The rigid element coefficients belonging to this column are added and the column is then repacked on file GM.

If multipoint constraints were not used, the coefficients for the rigid elements are read and packed in standard packed matrix form on file GM.

4.33.8 Subroutines

4.33.8.1 Subroutine Name: MCE1A

1. Entry Point: MCE1A
2. Purpose: To perform the functions described in 4.33.7
3. Calling Sequence: CALL MCE1A

4.33.8.2 Subroutine Name: MCE1B

1. Entry Point: MCE1B
2. Purpose: To decompose $[R_m]$ into lower and upper triangular factors.
3. Calling Sequence: CALL MCE1B

4.33.8.3 Subroutine Name: MCE1C

1. Entry Point: MCE1C
2. Purpose: Performs a forward-backward substitution to solve $[G_m]$.
3. Calling Sequence: CALL MCE1C

4.33.8.4 Subroutine Name: MCE1D

1. Entry Point: MCE1D
2. Purpose: To solve the matrix equation $[R_m] [G_m] = [R_n]$ for $[G_m]$ where $[R_m]$ is diagonal.
3. Calling Sequence: CALL MCE1D

MODULE FUNCTIONAL DESCRIPTIONS

4.33.9 Design Requirements

4.33.9.1 Allocation of Core Storage

The maximum core storage requirement in the module is one double precision vector in the u_m displacement set plus three GINØ buffers.

4.33.9.2 Environment

Communication of GINØ file names to each of the phases of MCE1 occurs through blank COMMON. The four phases are designed so that each may be in a separate overlay segment. Open core for each of the phases is as follows:

MCE1A: /MCEA1/

MCE1B: /MCEB1/

MCE1C: /MCEC1/

MCE1D: /MCED1/.

4.33.10 Diagnostic Messages

The following messages may be issued by MCE1:

3005, 3016

FUNCTIONAL MODULE MCE2 (MULTIPOINT CONSTRAINT ELIMINATOR - PHASE 2)

4.34 FUNCTIONAL MODULE MCE2 (MULTIPOINT CONSTRAINT ELIMINATOR - PHASE 2)

4.34.1 Entry Point: MCE2

4.34.2 Purpose

MCE2 partitions the stiffness matrix $[K_{gg}]$ into $[\bar{K}_{nn}]$, $[\bar{K}_{mn}]$ and $[\bar{K}_{mm}]$ and then performs the matrix reduction $[K_{nn}] = [\bar{K}_{nn}] + [G_m]^T [\bar{K}_{mn}] + [\bar{K}_{mn}]^T [G_m] + [G_m]^T [\bar{K}_{mm}] [G_m]$. Similar partitions and reductions are performed on $[M_{gg}]$, $[B_{gg}]$ and $[K_{gg}^4]$ if these matrices are not purged.

4.34.3 DMAP Calling Sequence

MCE USET,GM, $\begin{Bmatrix} KDGG \\ KGG \\ HKGGX \\ HKGG \end{Bmatrix}$, $\begin{Bmatrix} MGG \\ HRGG \end{Bmatrix}$, $\begin{Bmatrix} BGG \\ HBGG \end{Bmatrix}$, K4GG/ $\begin{Bmatrix} KDNN \\ KNN \\ HKNN \end{Bmatrix}$, $\begin{Bmatrix} MNN \\ HRNN \end{Bmatrix}$, $\begin{Bmatrix} BNN \\ HBNN \end{Bmatrix}$, K4NN \$

4.34.4 Input Data Blocks

USET - Displacement set definitions table.
 GM - Multipoint constraint transformation matrix - m set.
 KDGG - Partition of differential stiffness matrix - g set.
 KGG - Partition of stiffness matrix - g set.
 MGG - Partition of mass matrix - g set.
 BGG - Partition of damping matrix - g set.
 K4GG - Partition of structural damping matrix - g set.
 $\begin{Bmatrix} HKGGX \\ HKGG \end{Bmatrix}$ - Partition of conductivity matrix - g set.
 HRGG - Partition of radiation matrix - g set.
 HBGG - Partition of capacity matrix - g set.

Note: MGG, BGG and K4GG may be purged.

4.34.5 Output Data Blocks

KDNN - Partition of differential stiffness matrix - g set.
 KNN - Partition of stiffness matrix - n set.
 MNN - Partition of mass matrix - n set.
 BNN - Partition of damping matrix - n set.
 K4NN - Partition of structural damping matrix - n set.
 HKNN - Partition of conductivity matrix - n set.
 HRNN - Partition of radiation matrix - n set.
 HBNN - Partition of capacity matrix - n set.

Note: MNN, BNN or K4NN may be purged only if MGG, BGG or K4GG is purged.

4.34.6 Parameters

None

4.34.7 Method

Using subroutine UPART to generate row and column partitioning vectors and subroutine MPART to perform the actual partitioning, $[K_{gg}]$ is partitioned as follows:

$$[K_{gg}] \Rightarrow \begin{bmatrix} \bar{K}_{nn} & \bar{K}_{nm} \\ \bar{K}_{mn} & \bar{K}_{mm} \end{bmatrix}. \quad (1)$$

Subroutine ELIM is called to perform the following matrix reduction:

$$[K_{nn}] = [\bar{K}_{nn}] + [G_m]^T [\bar{K}_{mn}] + [\bar{K}_{mn}]^T [G_m] + [G_m]^T [\bar{K}_{mm}] [G_m]. \quad (2)$$

For each of the data blocks corresponding to the matrices $[M_{gg}]$, $[B_{gg}]$, $[K_{gg}^4]$ which is not purged, the above partitioning and matrix reductions are performed.

4.34.8 Subroutines

Calls are made to the following matrix utility routines:

UPART see section 3.5.9 for details

MPART see section 3.5.9 for details

ELIM see section 3.5.22 for details

4.34.9 Design Requirements

4.34.9.1 Allocation of Core Storage

The maximum storage requirement for MCE2 is four times the number of degrees of freedom in the u_n displacement set plus one GINØ buffer.

4.34.9.2 Environment

The module MCE2 consists of one subroutine, MCE2. Calls are made to the matrix utility routines as indicated above. Six scratch files are used.

FUNCTIONAL MODULE SCE1 (SINGLE-POINT CONSTRAINT ELIMINATOR)

4.35 FUNCTIONAL MODULE SCE1 (SINGLE-POINT CONSTRAINT ELIMINATOR)

4.35.1 Entry Point: SCE1

4.35.2 Purpose

To reduce the n set matrices to f set matrices by removing the single-point constraints.

4.35.3 DMAP Calling Sequence

SCE1 {USET } , {KDNN } , {MNN } , {BNN } , K4NN / {KDFF } , {KFS } , {KSS } , {MFF } , {BFF } ,
 {HUSET } , {KNN } , {HRNN } , {HBNN } , {KFF } , {HKFS } , {HKSS } , {HRFF } , {HBFF } , K4FF / \$

4.35.4 Input Data Blocks

{HUSET }
{USET } - Displacement set definitions table.

KDNN - Partition of differential stiffness matrix - n set.

KNN - Partition of stiffness matrix - n set

MNN - Partition of mass matrix - n set.

BNN - Partition of damping matrix - n set.

K4NN - Partition of the structural damping matrix - n set.

HKNN - Partition of conductivity matrix - n set.

HRNN - Partition of radiation matrix - n set.

HBNN - Partition of capacity matrix - n set.

- Notes: 1. USET cannot be purged
2. KNN, MNN, BNN and K4NN can be purged.
3. At least one degree of freedom must belong to the f and s sets.

4.35.5 Output Data Blocks

KDFF - Partition of differential stiffness matrix after single-point constraints have been removed - f set.

KFF - Partition of stiffness matrix after single-point constraints have been removed - f set.

KDFS - Partition of differential stiffness matrix after single-point constraints have been removed.

MODULE FUNCTIONAL DESCRIPTIONS

- KFS - Partition of stiffness matrix after single-point constraints have been removed.
- KDSS - Partition of differential stiffness matrix after single-point constraints have been removed - s set.
- KSS - Partition of stiffness matrix after single-point constraints have been removed - s set.
- MFF - Partition of mass matrix after single-point constraints have been removed - f set.
- BFF - Partition of damping matrix after single-point constraints have been removed - f set.
- K4FF - Partition of structural damping matrix with single-point constraints removed - f set.
- HKFF - Partition of conductivity matrix after single-point constraints have been removed - f set.
- HKFS - Partition of conductivity matrix after single-point constraints have been removed.
- HKSS - Partition of conductivity matrix after single-point constraints have been removed - s set.
- HRFF - Partition of radiation matrix after single-point constraints have been removed - f set.
- HBFF - Partition of capacity matrix after single-point constraints have been removed - f set.

4.35.6 Parameters

None

4.35.7 Method

The matrices are partitioned using USET(UN,UF,US) as follows (see section 1.7 for details):

1. If $[K_{nn}]$ exists:

$$[K_{nn}] \Rightarrow \left[\begin{array}{c|c} K_{ff} & K_{fs} \\ \hline K_{sf} & K_{ss} \end{array} \right]. \quad (1)$$

The $[K_{ff}]$, $[K_{fs}]$ and $[K_{ss}]$ partitions are generated and saved.

2. If $[M_{nn}]$ exists:

$$[M_{nn}] \Rightarrow \left[\begin{array}{c|c} M_{ff} & M_{fs} \\ \hline M_{sf} & M_{ss} \end{array} \right]. \quad (2)$$

3. If $[B_{nn}]$ exists:

$$[B_{nn}] \Rightarrow \left[\begin{array}{c|c} B_{ff} & B_{fs} \\ \hline B_{sf} & B_{ss} \end{array} \right]. \quad (3)$$

4. If $[K_{nn}^4]$ exists:

$$[K_{nn}^4] \Rightarrow \left[\begin{array}{c|c} K_{ff}^4 & K_{fs}^4 \\ \hline K_{sf}^4 & K_{ss}^4 \end{array} \right]. \quad (4)$$

For the $[M_{nn}]$, $[B_{nn}]$ and $[K_{nn}^4]$ matrices, only the "ff" partition is generated and saved.

One call to UPART followed by 4 calls to MPART accomplishes the above tasks.

4.35.8 Subroutines

UPART and MPART are called. See subroutine description in section 3.5.9.

4.35.9 Design Requirements

One scratch file is necessary.

FUNCTIONAL MODULE SMP1 (STRUCTURAL MATRIX PARTITIONER - PHASE 1)

4.36 FUNCTIONAL MODULE SMP1 (STRUCTURAL MATRIX PARTITIONER - PHASE 1)

4.36.1 Entry Point: SMP1

4.36.2 Purpose

SMP1 partitions $[K_{ff}]$ into $[\bar{K}_{aa}]$, and $[K_{oa}]$ and $[K_{oo}]$. The matrix equations $[K_{oo}] [G_o] = -[K_o]$ is solved for $[G_o]$. $[K_{ff}]$ is then reduced by the matrix equation $[K_{aa}] = [\bar{K}_{aa}] + [K_{oa}]^T [G_o]$. If $[M_{ff}]$ is not purged, it is reduced by the equation $[M_{aa}] = [\bar{M}_{aa}] + [G_o]^T [M_{oa}] + [M_{oa}]^T [G_o] + [G_o]^T [M_{oo}] [G_o]$. Similarly, $[B_{ff}]$ and $[K_{ff}^4]$ are reduced.

4.36.3 DMAP Calling Sequence

SMP1 {HUSET},{HKFF}, MFF,BFF,K4FF/{HG0},{HKAA},{HK00},{HL00},MAA,M00,M0A,BAA,K4AA / \$
 {USET},{KFF},{G0},{KAA},{K00},{L00}

4.36.4 Input Data Blocks

HUSET - Displacement set definitions table.
 USET
 KFF - Partition of stiffness matrix - f set.
 MFF - Partition of mass matrix - f set.
 BFF - Partition of damping matrix - f set.
 K4FF - Partition of structural damping matrix - f set.
 HKFF - Partition of conductivity matrix - f set.

Note: MFF, BFF or K4FF may be purged.

4.36.5 Output Data Blocks

G0 - Structural matrix partitioning transformation matrix.
 KAA - Partition of stiffness matrix - a set.
 K00 - Partition of stiffness matrix - o set.
 L00 - Lower triangular factor of K00B - o set.
 MAA - Partition of mass matrix - a set.
 M00 - Partition of mass matrix - a set.
 M0A - Partition of mass matrix.
 BAA - Partition of damping matrix - a set.

MODULE FUNCTIONAL DESCRIPTIONS

- K4AA - Partition of structural damping matrix - a set.
- HGØØ - Heat matrix partitioning transformation matrix.
- HKAA - Partition of conductivity matrix - a set.
- HKØØ - Partition of conductivity matrix - o set.
- HLØØ - Lower triangular factor of HKØØ - o set.

Note:

1. UØØ and LØØ are not standard form matrices. Their format is compatible only for input to subroutine FBS.
2. MAA, MØØ, MØA, BAA or K4AA may be purged only if MFF, BFF or K4FF are purged.

4.36.6 Parameters

None.

4.36.7 Method

Using subroutine UPART to generate row and column partitioning vectors and subroutine MPART to perform the actual partitioning, $[K_{ff}]$ is partitioned as follows:

$$[K_{ff}] \Rightarrow \left[\begin{array}{c|c} \bar{K}_{aa} & K_{ao} \\ \hline K_{oa} & K_{oo} \end{array} \right] \quad (1)$$

Subroutine FACTØR is called to decompose $[K_{oo}]$ into triangular factors. Subroutine SØLVER is called to perform a forward-backward substitution solving for $[G_o]$ in the matrix equation

$$[K_{oo}] [G_o] = -[K_{oa}] , \quad (2)$$

and computing $[K_{aa}]$ from the equation

$$[K_{aa}] = [\bar{K}_{aa}] + [K_{oa}] [G_o] . \quad (3)$$

For each of the data blocks $[M_{ff}]$, $[B_{ff}]$, $[K_{ff}^4]$ which is not purged, the above partitioning operation is performed and the matrix reductions:

$$[M_{aa}] = [\bar{M}_{aa}] + [G_o]^T [M_{oa}] + [M_{oo}]^T [G_o] + [G_o]^T [M_{oo}] [G_o] , \quad (4)$$

$$[B_{aa}] = [\bar{B}_{aa}] + [G_o]^T [B_{oa}] + [B_{oa}]^T [G_o] + [G_o]^T [B_{oo}] [G_o] , \quad (5)$$

$$[K_{aa}^4] = [\bar{K}_{aa}^4] + [G_o]^T [K_{oa}^4] + [K_{oa}^4]^T [G_o] + [G_o]^T [K_{oo}^4] [G_o] , \quad (6)$$

are performed by subroutine ELIM.

4.36.8 Subroutines

Calls are made to the following matrix utility routines:

UPART See subroutine descriptions - Section 3.5.9 for details
 MPART See subroutine descriptions - Section 3.5.9 for details
 FACTØR See subroutine descriptions - Section 3.5.23 for details
 SØLVER See subroutine descriptions - Section 3.5.20 for details
 ELIM See subroutine descriptions - Section 3.5.22 for details

4.36.9 Design Requirements

4.36.9.1 Allocation of Core Storage

The maximum storage requirement for SMP1 is four times the number of degrees of freedom in the u_a displacement set plus one GINØ buffer.

4.36.9.2 Environment

The module SMP1 consists of one subroutine, SMP1. Calls are made to the matrix utility routines indicated above. Six scratch files are used.

The matrix multiply-add routine is used by ELIM to perform the matrix reductions described by equations 3, 4, 5 and 6. For equations 4, 5 and 6, the reduction is done in three phases as shown below for the mass matrix.

$$[A] = [M_{oo}] [G_o] + [M_{oa}] \quad (7)$$

$$[B] = [M_{oa}]^T [G_o] + [\bar{M}_{aa}] \quad (8)$$

$$[M_{aa}] = [G_o]^T [A] + [B] \quad (9)$$

FUNCTIONAL MODULE RBMG1 (RIGID BODY MATRIX GENERATOR - PHASE 1)

4.37 FUNCTIONAL MODULE RBMG1 (RIGID BODY MATRIX GENERATOR - PHASE 1)

4.37.1 Entry Point: RBMG1

4.37.2 Purpose

RBMG1 partitions $[K_{aa}]$ into $[K_{\ell\ell}]$, $[K_{\ell r}]$ and $[K_{rr}]$. If $[M_{aa}]$ is not purged, it is partitioned similarly.

4.37.3 DMAP Calling Sequence

RBMG1 {HUSER}, {HKAA}, MAA / {KLL} {KLR} {KRR}, MLL, MLR, MRR \$
 {USER}, {KAA}

4.37.4 Input Data Blocks

USER - Displacement set definitions table.
HUSER - Temperature set definitions table.
KAA - Partition of stiffness matrix - a set.
MAA - Partition of mass matrix - a set.
HKAA - Partition of conductivity matrix - a set.

Note: USER may not be purged.

4.37.5 Output Data Blocks

KLL - Partition of stiffness matrix - ℓ set.
KLR - Partition of stiffness matrix.
KRR - Partition of stiffness matrix - r set.
MLL - Partition of mass matrix - ℓ set.
MLR - Partition of mass matrix.
MRR - Partition of mass matrix - r set.
HKLL - Partition of conductivity matrix - ℓ set.
HKLR - Partition of conductivity matrix.
HKRR - Partition of conductivity matrix - r set.

Note: Output data blocks may be purged only if the corresponding input data block is purged.

MODULE FUNCTIONAL DESCRIPTIONS

4.37.6 Parameters

None

4.37.7 Method

Using subroutine UPART to generate row and column partitioning vectors and subroutine MPART to perform the actual partitioning, $[K_{aa}]$ is partitioned as follows:

$$[K_{aa}] \rightarrow \left[\begin{array}{c|c} K_{ll} & K_{lr} \\ \hline K_{rl} & K_{rr} \end{array} \right]. \quad (1)$$

Similarly, if $[M_{aa}]$ is not purged, it is partitioned.

4.37.8 Subroutines

RBMG1 calls the following matrix utility routines:

UPART }
MPART } see section 3.5.9 for details.

4.37.9 Design Requirements

4.37.9.1 Allocation of Core Storage

Storage requirements for RBMG1 are minimal since no unpacked vectors are held in core.

4.37.9.2 Environment

The module RBMG1 consists of one subroutine, RBMG1. Calls are made to the matrix utility routines indicated above. One scratch file is used.

FUNCTIONAL MODULE RBMG2 (RIGID BODY MATRIX GENERATOR - PHASE 2)

4.38 FUNCTIONAL MODULE RBMG2 (RIGID BODY MATRIX GENERATOR - PHASE 2)

4.38.1 Entry Point: RBMG2

4.38.2 Purpose

RBMG2 decomposes $[K_{\ell\ell}]$ into its triangular factors $[L_{\ell\ell}]$ and $[U_{\ell\ell}]$.

4.38.3 DMAP Calling Sequence

$$\text{RBMG2} \quad \begin{pmatrix} \text{HKLL} \\ \text{KAA} \\ \text{KBLL} \\ \text{KLL} \\ \text{KKK} \end{pmatrix} / \begin{pmatrix} \text{HLLL} \\ \text{LBLL} \\ \text{LLL} \\ \text{LKK} \end{pmatrix} / \text{V,N,PØWER/V,N,DET \$}$$

4.38.4 Input Data Blocks

$\begin{pmatrix} \text{KAA} \\ \text{KBLL} \\ \text{KLL} \\ \text{KKK} \end{pmatrix}$ - Partition of stiffness matrix - 2 set.

HKLL - Partition of conductivity matrix - 2 set.

Note: KLL may not be purged.

4.38.5 Output Data Blocks

$\begin{pmatrix} \text{LBLL} \\ \text{LKK} \\ \text{LLL} \\ \text{HLLL} \end{pmatrix}$ - Lower triangular factor of KLL - 2 set.

Notes

1. LLL and ULL may not be purged.
2. ULL is not a standard upper triangular matrix. Its format is compatible only for input to subroutine FBS.

4.38.6 Parameters

PØWER - Output-integer-default = 1. Power of 10 in the determinant of KLL.

DET - Output-real-default = 1.0. Magnitude of determinant of KLL, i.e.,

$$\det [K_{\ell\ell}] = \text{DET} \cdot 10^{\text{PØWER}}.$$

MODULE FUNCTIONAL DESCRIPTIONS

4.38.7 Method

RBMG2 calls subroutine FACTØR to perform the deomcposition of $[K_{\ell\ell}]$ into $[L_{\ell\ell}]$ and $[U_{\ell\ell}]$.

4.38.8 Subroutines

RBMG2 calls the matrix utility routine FACTØR (see section 3.5.23 for FACTØR details).

4.38.9 Design Requirements

For allocation of core storage, see subroutine SDCØMP (section 3.5.14). The module RBMG2 consists of one subroutine, RBMG2. Three scratch files are used.

4.38.10 Diagnostic Messages

Message number 3005 may be issued by RBMG1.

FUNCTIONAL MODULE RBMG3 (RIGID BODY MATRIX GENERATOR - PHASE 3)

4.39 FUNCTIONAL MODULE RBMG3 (RIGID BODY MATRIX GENERATOR - PHASE 3)

4.39.1 Entry Point: RBMG3

4.39.2 Purpose

RBMG3 solves for the rigid body transformation matrix $[D]$ from the equation

$$[K_{\ell\ell}] [D] = -[K_{\ell r}] \quad (1)$$

The rigid body error ratio, ϵ , is computed from

$$\epsilon = \frac{||[K_{rr}] + [K_{\ell r}]^T [D]||}{||[K_{rr}]||} \quad (2)$$

Note: The absolute value $|| \quad ||$ is the square root of the sum of the squares (this is not a determinant).

4.39.3 DMAP Calling Sequence

RBMG3 $\left\{ \begin{array}{l} LLL \\ HLLL \end{array} \right\} \cdot \left\{ \begin{array}{l} HKLR \\ KLR \end{array} \right\} \cdot \left\{ \begin{array}{l} HKRR \\ KRR \end{array} \right\} / \left\{ \begin{array}{l} HDM \\ DM \end{array} \right\} \$$

4.39.4 Input Data Blocks

LLL - Lower triangular factor of KLL - ℓ set.

HLLL - Lower triangular factor of HKLL - ℓ set.

KLR - Partition of stiffness matrix.

HKLR - Partition of conductivity matrix.

KRR - Partition of stiffness matrix - r set.

HKRR - Partition of conductivity matrix - r set.

Note: Input data blocks may not be purged.

4.39.5 Output Data Blocks

$\left. \begin{array}{l} DM \\ HDM \end{array} \right\}$ - Rigid body transformation matrix.

Note: The DM data block corresponds to the matrix $[D]$ and may not be purged.

MODULE FUNCTIONAL DESCRIPTIONS

4.39.6 Parameters

None.

4.39.7 Method

Subroutine SØLVER is called to perform the operations in Equation 1 and 2.

4.39.8 Subroutines

RBMG3 calls the matrix utility routine SØLVER and has no auxiliary subroutines. See section 3.5.20 for SØLVER details.

4.39.9 Design Requirements

For allocation of core storage, see subroutines FBS (section 3.5.17) and MPYAD (section 3.5.12). Two scratch files are used.

FUNCTIONAL MODULE RBMG4 (RIGID BODY MATRIX GENERATOR - PHASE 4)

4.40 FUNCTIONAL MODULE RBMG4 (RIGID BODY MATRIX GENERATOR - PHASE 4)

4.40.1 Entry Point: RBMG4

4.40.2 Purpose

RBMG4 computes the rigid body mass matrix $[m_r]$ from the matrix equation

$$[m_r] = [M_{rr}] + [D]^T [M_{lr}] + [M_{lr}]^T [D] + [D]^T [M_{ll}] [D]. \quad (1)$$

4.40.3 DMAP Calling Sequence

RBMG4 DM,MLL,MLR,MRR/MR \$

4.40.4 Input Data Blocks

DM - Rigid body transformation matrix.

MLL - Partition of mass matrix - l set.

MLR - Partition of mass matrix.

MRR - Partition of mass matrix - r set.

Notes:

1. No input data block may be purged.
2. The DM data block corresponds to the matrix $[D]$ in Equation 1.

4.40.5 Output Data Blocks

MR - Rigid body mass matrix - r set.

4.40.6 Parameters

None

4.40.7 Method

Subroutine ELIM is called to compute $[m_r]$ as in Equation 1 (see section 3.5.22 for ELIM details).

4.40.8 Subroutines

RBMG4 consists of one subroutine, RBMG4.

Matrix utility routine ELIM (section 3.5.22) is called by RBMG4.

4.40.9 Design Requirements

For allocation of core storage, see subroutine MPYAD (section 3.5.12). Three scratch files are used.

FUNCTIONAL MODULE SSG1 (STATIC SOLUTION GENERATOR - PHASE 1)

4.41 FUNCTIONAL MODULE SSG1 (STATIC SOLUTION GENERATOR - PHASE 1)

4.41.1 Entry Point: SSG1

4.41.2 Purpose

To compute the static loads, thermal loads, and enforced deformation loads selected by the user.

4.41.3 DMAP Calling Sequence

SSG1 {HSLT},BGPDT,CSTM,{HSIL},{HEST},MPT,GPTT,EDT,MGG,CASECC,DIT/{PG1}/V,N,LUSET/V,N,NSKIP \$
SLT }EST }PG

4.41.4 Input Data Blocks

HLT - Static Loads Table (heat problems).
SLT - Static Loads Table.
BGPDT - Basic Grid Point Definition Table.
CSTM - Coordinate System Transformation Matrices.
HSIL - Scalar Index List (heat problems).
SIL - Scalar Index List.
HEST - Element Summary Table (heat problems).
EST - Element Summary Table.
MPT - Material Property Table.
GPTT - Grid Point Temperature Table.
EDT - Element Deformation Table.
MGG - Partition of mass matrix - g set.
CASECC - Case Control Data Table.
DIT - Direct Input Tables.

Notes:

1. SLT, BGPDT, SIL cannot be purged if external static loads or LOAD cards are selected in CASECC.
2. CSTM cannot be purged if any grid point or load references a coordinate system other than basic.

MODULE FUNCTIONAL DESCRIPTIONS

3. EST, MPT cannot be purged if thermal or element deformation loads are selected.
4. GPTT cannot be purged if thermal loads are applied.
5. EDT cannot be purged if element deformation loads are selected.
6. MGG cannot be purged if GRAVITY or RFØRCE loads are applied.
7. CASECC cannot be purged.
8. DIT cannot be purged if temperature dependent materials are loaded.

4.41.5 Output Data Blocks

HPG }
PG } - Static load vector matrix giving static loads - g set.
PG1 }

Note: PG can never be purged.

4.41.6 Parameters

LUSET - Input-integer-no default. LUSET defines length of PG.

NSKIP - Input-integer-no default. One static load is built for each CASECC record starting with NSKIP + 1 as long as the boundary conditions are constant.

4.41.7 Overview of the Method Used in SSG1

The purpose of the first phase of static solution calculation (module SSG1) is the generation of the load vectors on the whole structure. The structure may be loaded in three different ways:

1. Simple applied loads and moments may be given to grid and scalar points. Pressure loads may be applied to an area defined by three or four grid points or to a face of an isoparametric solid element. Centrifugal force fields may also be defined.
2. Thermal and enforced deformation loads are generated by using the structural element characteristics. The loads on the connected grid points are equivalent to fixing the displacements and replacing the element by the load it would apply to the points.
3. Gravity loads are dependent on the mass characteristics of the structure. A gravity load is produced by generating a vector of accelerations on all grid points in the structure and pre-multiplying the vector by the structural mass matrix.

FUNCTIONAL MODULE SSG1 (STATIC SOLUTION GENERATOR - PHASE 1)

The details on load vector generation for these three different types of loading are given in Sections 4.41.8, 4.41.9 and 4.41.10 below. The function of module SSG1 is to read the case control data and extract the necessary load set data, calculate load vectors for each requested load set, and combine these sets to produce the loads requested for all subcases using the same boundary condition.

4.41.7.1 Module Initialization

Some of the simple load cards will reference elements, rather than grid points. This is especially true of the heat transfer loads QBDY1, QBDY2, QVECT, and QVØL. Subroutine SSGSLT processes these cards to produce loads at grid points. The SLT file is copied onto a scratch file (NEWSLT), with the converted load cards replacing the old load cards.

Common block /LØADX/, which contains GINØ file numbers for input data files and position pointers, is initialized.

A list of all external load sets is extracted from the SLT. (This must be less than 101.) CASECC is skipped forward NSKIP records (in case several boundary conditions are being solved in one run). For each succeeding record which is not an eigenvalue record, not a symmetry record, not a differential stiffness record, and for which the boundary conditions are those for the current loop (SPC and MPC sets), a list is made of each thermal or enforced deformation load. The external loads selected are marked in the above list. If a selected external load is not in the above list, the LØAD cards are read in and their component id's searched. A LØAD card may cause additional members to be selected. A composite list is created which contains:

	} External load id's selected ones marked ≤ 360	} Thermal load id's ≤ 360	} Enforced deformation load id's ≤ 360	} < 360

If there is no record which allows construction of a load, SSG1 aborts. If a selected external load id does not exist either as a LØAD card or simple load set, SSG1 aborts. (Subroutine SSG1A).

MODULE FUNCTIONAL DESCRIPTIONS

4.41.7.2 Individual Load Vector Generation

Each requested set of loads is used to generate a $\{p_g^j\}$ load vector. The vectors are generated one at a time in core and written on the PG temporary file, a scratch file. Files PG temporary, SLT (or NEWSLT), BGPDT, CSTM, and SIL are opened. The vector generation depends on the type of load and type of input data. Details are given in Sections 4.41.8, 4.41.9, and 4.41.10 below.

4.41.7.3 Subcase Load Vector Generation

Each simple load set, j , produces a $\{p_g^j\}$ load vector, and each subcase may be a combination of various simple load sets. As each load set vector is formed, it is written on PG temporary. When all sets have been generated, the CASECC data block and the LOAD card images are read again. A table is formed for each subcase consisting of the required set number and the scale factor for each set if given on a LOAD card. The file containing the load vectors for the sets is read for each subcase, c , and added to a $\{p_g^c\}$ load vector. The $\{p_g^c\}$ load vectors are packed and written as the PG data block in standard NASTRAN form.

4.41.8 Direct Applied Loads

Direct loads are applied to the structural model by means of FORCE, FORCE1, FORCE2, GRAV, MOMENT, MOMENT1, LOAD, LOAD2, LOAD3, RFORCE, and SLLOAD Bulk Data Cards and the PRESAX card which is used for the axisymmetric conical shell problem only.

4.41.8.1 FORCE and MOMENT Card Processing

The data described by a FORCE and MOMENT data card are given as follows:

- N_p = Grid point index;
- N_c = Coordinate system number;
- S = Scalar factor; and

$$\{P\} = \begin{pmatrix} A_1 \\ A_2 \\ A_3 \end{pmatrix} \quad (1)$$

FUNCTIONAL MODULE SSG1 (STATIC SOLUTION GENERATOR - PHASE 1)

The BGPDT data for the point are determined. If the global coordinate number, N_g , for the point equals N_c , the vector is

$$\{P_g\} = S P \quad , \quad (2)$$

where the row index is determined from SIL. If $N_g \neq N_c$, $[T_g]$ and $[T_c]$ are calculated using the location coordinates and the two local coordinate systems (subroutines GLBBAS and BASGLB). The load in global coordinates is:

$$\{P_g\} = S [T_g]^T [T_c] \{P\} . \quad (3)$$

If a FØRCE card is used, the loads are added to the first three positions for the grid point in the load vector. If a MØMENT card is used, the loads are added to the last three positions. (Subroutine DIRECT).

4.41.8.2 FØRCE1 and MØMENT1 Card Processing

The data described by FØRCE1 or MØMENT1 card are given as follows:

- N_p = Application point number;
- S = Load magnitude;
- N_1, N_2 = Grid point numbers describing the vector direction of the load.

The basic coordinates of the points N_1 , N_2 and N_p are found in the BGPDT. (Subroutines PERMUT and FNDPNT). If $\{R_1\}$ and $\{R_2\}$ are the vectors corresponding to N_1 and N_2 , the load direction is:

$$\{d\} = \frac{\{R_2\} - \{R_1\}}{|\{R_2\} - \{R_1\}|} . \quad (4)$$

The coordinate transformation $[T_g]$ for point N_p is calculated (Subroutine BASGLB). The load vector in global coordinates is:

$$\{P_g\} = S [T_g]^T \{d\} . \quad (5)$$

If a FØRCE1 card was used the values are added to the first three coordinates, starting with the SIL number, in the load vector. If a MØMENT1 card was used, the values are added to the last three (subroutine TPØNT).

MODULE FUNCTIONAL DESCRIPTIONS

4.41.8.3 FØRCE2 and MØMENT2 Card Processing

The data on a FØRCE2 or MØMENT2 card are as follows:

N_p = Application point number;

S = Load magnitude;

N_1, N_2, N_3, N_4 are such that the direction of the force is determined by Equation 6 below.

The algorithm is similar to the one for the FØRCE1 and MØMENT1 case except that four basic coordinate system vectors, $\{R_1\}$, $\{R_2\}$, $\{R_3\}$, $\{R_4\}$, are formed for the four points and:

$$\{d\} = \frac{(\{R_2\} - \{R_1\}) \times (\{R_4\} - \{R_3\})}{|(\{R_2\} - \{R_1\}) \times (\{R_4\} - \{R_3\})|} \quad (6)$$

(Subroutine FPØNT).

4.41.8.4 PLØAD and PLØAD2 Card Processing

The data contents for a PLØAD card are (a PLØAD2 card is transformed into a PLØAD card by GP3):

p = Pressure value;

N_1, N_2, N_3, N_4 = Points describing area over which pressure load is acting.
(N_4 is optional.)

For each of the four points, N_i , the basic coordinate system vector, $\{R_i\}$, is formed.

If $N_4 = 0$, the load on each point is:

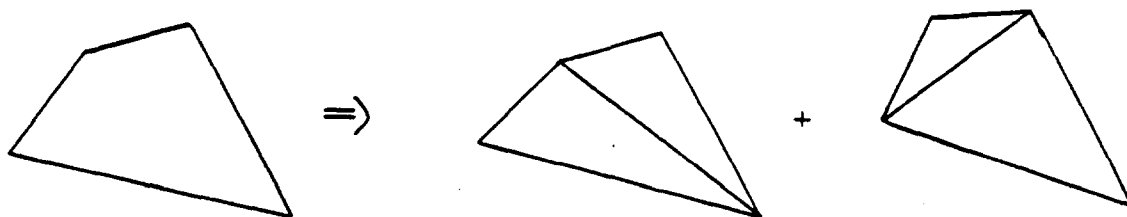
$$\{F\} = -\frac{p}{6} [(\{R_1\} - \{R_2\}) \times (\{R_3\} - \{R_2\})] \quad (7)$$

The load vector in global coordinates for each point is:

$$\{P_{g_i}\} = [T_i]^T \{F\} \quad (8)$$

FUNCTIONAL MODULE SSG1 (STATIC SOLUTION GENERATOR - PHASE 1)

If $N_4 \neq 0$, the quadrilateral is subdivided into four triangles as shown.



For each triangle $\{P_{g_i}\}$ is calculated for the three connected points using a pressure value $p_T = 1/2 p$. The equations are the same as the previous case except that the points are interchanged each time the triangle calculation is done. (Subroutine PLØAD.)

4.41.8.5 SLØAD Card Processing

The data contents for a SLØAD card are:

N_p = Scalar point id and
 S = Load on point.

The scalar index is computed by subroutine FNDSIL, and S is added in. (Subroutine SLØAD.)

4.41.8.6 RFØRCE Card Processing

The data contents for an RFØRCE card are:

N_p = Index of grid point through which
rotation vector passes;
 N_c = Coordinate system number defining
the rotation vector;
 A = Factor for vector;

R_x, R_y, R_z are components of rotation vector in cps.

The following sequence of operations comprises RFØRCE card processing which is carried out in subroutine RFØRCE.

1. The local to basic coordinate transformation matrix, $[T_c]$, for the reference coordinate system N_c is extracted from the CSTM data block.

MODULE FUNCTIONAL DESCRIPTIONS

2. The rotation vector in basic coordinates, and in radians per second, is:

$$\{\omega_b\} = 2 \pi A [T_c] \begin{Bmatrix} R_x \\ R_y \\ R_z \end{Bmatrix} \quad (9)$$

3. Define the basic location vector of the reference point N_p as $\{r_a\}$. If $N_p = 0$, set $\{r_a\} = \{0\}$.
4. Extract the basic coordinate system vector $\{r_i\}$ for each point from BGPDT.
5. Using $\{r_i\}$ and the local coordinate system referenced by the point, calculate the global-to-basic transformation matrix $[T_i]$.
6. For the six columns of the mass matrix $[M_{gg}]$ corresponding to the grid point i , the 6x6 matrix partition on the diagonal is extracted. Define this as $[M^i]$.
7. Partition the 6x6 matrix into 3x3 matrices

$$[M^i] \Rightarrow \begin{bmatrix} M_t^i & M_{tr}^i \\ M_{rt}^i & M_r^i \end{bmatrix}, \quad (10)$$

and transform the rotational velocity vector to global coordinates

$$\{\omega_g\} = [T_i]^T \{\omega_b\}. \quad (11)$$

8. Calculate the forces and moments on the grid point by the equations:

$$\{F\} = -\{\omega_g\} \times [M_t^i] [T_i]^T (\{\omega_b\} \times [\{r_i\} - \{r_a\}]) - \{\omega_g\} \times [M_{tr}^i] \{\omega_g\} \quad (12)$$

$$\{M\} = -\{\omega_g\} \times [M_{rt}^i] [T_i]^T (\{\omega_b\} \times [\{r_i\} - \{r_a\}]) - \{\omega_g\} \times [M_r^i] \{\omega_g\} \quad (13)$$

9. The load vector partition in global coordinates is:

$$\{P_i\} = \begin{Bmatrix} F \\ M \end{Bmatrix} \quad (14)$$

4.41.8.7 PRESAX Card Processing

The data contents for a PRESAX card (which applies only to pressure loading of an AXISYMMETRIC shell) are:

FUNCTIONAL MODULE SSG1 (STATIC SOLUTION GENERATOR - PHASE 1)

- P = Pressure;
 N_a = Index value of harmonic Ring A;
 N_b = Index value of harmonic Ring B;
 ϕ_1 = (degrees);
 ϕ_2 = (degrees);
 n = Harmonic number of harmonic being added.

The algorithm given in the following steps is performed in subroutine PRESAX.

1. ϕ_1 and ϕ_2 are converted to radians.
2. BGPDT data are extracted for both RINGA and RINGB, giving $\{r\}$ and $\{z\}$.
3. The calculations for harmonic n are:

$$\ell = \sqrt{(r_b - r_a)^2 + (z_b - z_a)^2}, \quad (15)$$

$$\sin \psi = \frac{r_b - r_a}{\ell}, \quad (16)$$

$$\cos \psi = \frac{z_b - z_a}{\ell}. \quad (17)$$

For the cosine case, if $n = 0$, we calculate:

$$P_{r0}^i = P \ell \left(\frac{r_i}{3} + \frac{r_j}{6} \right) (\phi_2 - \phi_1) \cos \psi, \quad (18)$$

$$P_{z0}^i = -P \ell \left(\frac{r_i}{3} + \frac{r_j}{6} \right) (\phi_2 - \phi_1) \sin \psi. \quad (19)$$

If $n > 0$

$$P_{rn}^i = P \frac{\ell}{n} \left(\frac{r_i}{3} + \frac{r_j}{6} \right) (\sin(n\phi_2) - \sin(n\phi_1)) \cos \psi, \quad (20)$$

$$P_{zn}^i = -P \frac{\ell}{n} \left(\frac{r_i}{3} + \frac{r_j}{6} \right) (\sin(n\phi_2) - \sin(n\phi_1)) \sin \psi. \quad (21)$$

For the sine case, if $n > 0$,

$$p_{rn}^i = -p \frac{\ell}{n} \left(\frac{r_i}{3} + \frac{r_j}{6} \right) (\cos(n\phi_2) - \cos(n\phi_1)) \cos \psi, \quad (22)$$

$$p_{zn}^i = p \frac{\ell}{n} \left(\frac{r_i}{3} + \frac{r_j}{6} \right) (\cos(n\phi_2) - \cos(n\phi_1)) \sin \psi. \quad (23)$$

4. The above equations are solved for $i = a, j = b$ and $i = b, j = a$. The loads are added to the corresponding grid point location in the PG load vector

$$\left\{ p_g^i \right\} = \left\{ p_g^i \right\} + \begin{pmatrix} p_r^i \\ 0 \\ p_z^i \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad (24)$$

(Subroutine PRESAX).

4.41.8.8 GRAV Card Processing

Each GRAV input card describes a uniform acceleration field with the following parameters:

- N = Coordinate system ID;
- G = Scale factor;
- {V} = Vector of load in coordinate system N.

The gravity vector in basic coordinates is:

$$\{g_b\} = G [T_{ON}] \{V\}. \quad (25)$$

where $[T_{ON}]$ is the 3x3 orientation matrix of coordinate system N. (Subroutines GRAV, FDCSTM, MPYL). This vector $\{g_b\}$ is saved for later processing. Subroutine EXTERN then returns, noting the number of gravity loads listed.

MODULE FUNCTIONAL DESCRIPTIONS

4.41.8.9 PLØAD3 Card Processing

The data contents from a PLØAD3 card, as converted in GP3, are:

P_1, P_2, \dots, P_6 = Normal pressure value on faces 1 to 6

$N_1^g, N_2^g, \dots, N_{32}^g$ = Internal grid point numbers defining element on which pressure acts (N_9^g to N_{32}^g may be zero depending on element type).

The pressure load vector on the i^{th} element grid point is:

$$\{F_i\} = \sum_{k=1}^6 P_k \sum_{m=1}^2 \sum_{n=1}^2 N_i^s \{B_k\}$$

Where P_k is the normal pressure on the k^{th} face, N_i^s is the isoparametric shape function, and B_k is an area factor times a vector of direction cosines defining the outward normal to the k^{th} face at each integration point. The summation on m and n are for integration over the area of the k^{th} face using the method of Gaussian Quadrature. The load vector transformed to global coordinates for the i^{th} point is:

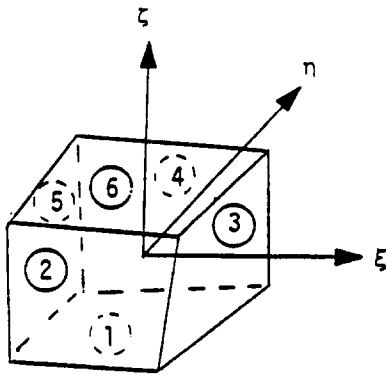
$$\{P_{g_i}\} = [T_i]^T \{F_i\}$$

where $[T_i]$ is the basic to global transformation matrix.

The computational method using in computing the load vector is as follows: Each element of a vector $\{F\}$ of length three times the number of grid points is set to zero. The following steps are repeated at each integration point for each face which has a non-zero applied pressure:

1. Isoparametric utility routines are called to compute N_i^s 's and $[J]^{-1}$ (see Section 4.87.16.2).
2. For each grid point i , and element face k , $\{B_k\}$ is multiplied by $N_i^s P_k$ and added to $\{F_i\}$. $\{B_k\}$ is simply ± 1 times $\det [J]$ times a column of $[J]^{-1}$. The column and sign are dependent on the face numbers as follows:

MODULE FUNCTIONAL DESCRIPTIONS



Face	Sign	Column Number of $[J]^{-1}$
1	-	3
2	-	2
3	+	1
4	+	2
5	-	1
6	+	3

The $\{F\}$ vector is then transformed to global coordinates and added to the global static load vector $\{P_g\}$.

4.41.9 Thermal and Enforced Deformation Loads

The thermal and enforced deformation loads are calculated using the stiffness properties of the structural elements. The EDT data for each load set and the MPT, DIT and SIL data blocks are placed in core. The EST data block and the GPTT data for the selected set are read one element at a time. The loads produced by that element are placed in the PG load vector. The actual algorithms for generating element loads are given in Section 4.87.

4.41.10 Gravity Loads

Acceleration vectors are computed for each gravity load by two means, one for an axisymmetric shell problem, the other for non-shell problems.

4.41.10.1 Gravity Loads for an Axisymmetric Shell Problem

m (number of rings) and n (number of harmonics) are extracted from the /SYSTEM/ common block. The first m points in the BGPDT define the "zero" harmonic. The second m entries define the "one" harmonic etc. The acceleration vectors are calculated by the formulae:

$$g = \sqrt{g_x^2 + g_y^2 + g_z^2}, \quad (26)$$

$$g_{xy} = \sqrt{g_x^2 + g_y^2}, \quad (27)$$

$$\cos \theta_g = \frac{g_z}{g}, \quad (28)$$

$$\sin \theta_g = \frac{g_{xy}}{g}, \quad (29)$$

$$\sin \phi_g = \frac{g_y}{g_{xy}}, \quad (30)$$

$$\cos \phi_g = \frac{g_x}{g_{xy}}. \quad (31)$$

The vectors $\{a\}$ for harmonics $n = 0$ and $n = 1$ are defined for load set cosine by:

$$\{a_0^C\} = g \cos \theta_g \begin{Bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{Bmatrix} \text{ (all rings),} \quad (32)$$

$$\{a_1^C\} = g \sin \theta_g \cos \phi_g \begin{Bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix}; \quad (33)$$

(all rings)

and for load set sine by:

$$\{a_1^S\} = g \sin \theta_g \sin \phi_g \begin{Bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix}. \quad (34)$$

These vectors are used as in the normal case. They are merged into $\{a_g\}$ which in turn is pre-multiplied by $[M_{gg}]$ to give the $\{P_g\}$ vector (subroutine GRAVL3).

4.41.10.2 Gravity Loads for Non-Shell Problems

The acceleration vector must be transferred to the global coordinate system at each grid point and expanded to a vector acting on the u_g coordinates. For each grid point (i) the BGPDT data is read, and using the CSTM data, a 3x3 basic to global transformation matrix $[T_i]$ is formed. The acceleration at the point i in basic coordinates is:

$$\{a_g^i\} = [T_i] \{g_b\}, \quad (35)$$

where $\{g_b\}$ is the gravity vector saved in Equation 25. The vector $\{a_g^i\}$ is placed in the total acceleration vector in positions SIL_i , SIL_i+1 , and SIL_i+2 . No values are calculated for scalar points or rotation coordinates (Subroutine GRAVL1).

When all $\{a_g\}$ vectors have been calculated for the whole structure, they are pre-multiplied by the structural mass matrix to produce a load vector:

$$\{P_g\} = [M_{gg}] \{a_g\}, \quad (36)$$

(subroutine SSG2B).

The gravity vectors are appended to the other load vectors, and scalar points are zeroes in case interaction occurred. Gravity loads on scalar points are not supported.

4.41.10.3 Direct-Applied Thermal Loads

Direct loads are applied to the heat transfer model by means of the QHBDY data cards. These cards contain the following data:

<u>Symbol</u>	<u>Description</u>
FLAG,	Identified type of load
Q_0 ,	Flux density
A_f ,	Area factor
G_1, G_2, G_3, G_4	Internal grid point numbers

The word "FLAG" indicates the type of load, "POINT," "LINE," "REV," "AREA3," or "AREA4," and the number of grid points defined by G_1, G_2 , etc. The loads are formed into a vector $\{P\}$ with a length equal to the number of points. The values of $\{P\}$ are:

$$\{P\} = \bar{A}Q_0\{V\}.$$

The values for \bar{A} and $\{V\}$ are given in the following table:

FLAG	Number of grid-points at which load vector is applied	\bar{A}	$\{V\}$
1	1	$-A_f$	$\{1\}$
2	2	$-\frac{A_f(\text{length})}{2}$	$\begin{Bmatrix} 1 \\ 1 \end{Bmatrix}$
3	2	$-\frac{\pi(\text{length})}{3}$	$\begin{Bmatrix} 2x_1 + x_2 \\ x_1 + 2x_2 \end{Bmatrix}$
4	3	$-\frac{(\text{area})}{3}$	$\begin{Bmatrix} 1 \\ 1 \\ 1 \end{Bmatrix}$
5	4	make into overlapping triangles, as in FLAG = 4 (divide loads by two)	

MODULE FUNCTIONAL DESCRIPTIONS

where the values x_i, y_i, z_i are the BGPDT values for point i and:

$$(\text{length}) = [(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2]^{1/2}$$

$$(\text{area}) = \frac{1}{2} |r_{12} \times r_{13}|$$

$$\text{where } r_{1j} = \begin{Bmatrix} x_j - x_1 \\ y_j - y_1 \\ z_j - z_1 \end{Bmatrix} \quad j = 2, 3$$

The load is to be inserted into all three translational degrees of freedom at the gridpoints. The scalar indices may be found in the SIL table.

4.41.11 Processing of Heat Transfer Loads

In a heat transfer problem, the applied load vector, $\{P_g\}$, is a vector of heat flow values input to each grid point. These loads may be input by QHBDY, QBDY1, QBDY2, QVECT, QVØL, or SLØAD data cards. All other load types are ignored. The process for each card is as follows:

QHBDY - Heat flux into area defined by grid points. This card is not processed by SSGSLT. The load applied to each point is from subroutine QHBDY as:

$$P_i = A_i Q_o,$$

where the values A_i are the area associated with each point if the points define a surface of TYPE = PØINT, LINE, REV, AREA3, and AREA4. These areas are defined in Section 4.87.18.

QBDY1 - Uniform heat flux into an element. SSGSLT processes the referenced HBDY elements to calculate the factors A_i for all connected points. The loads produced by subroutine QLØAD1 are:

$$P_i = A_i Q_i$$

QVECT - Vector heat flux into an element. SSGSLT processes the referenced HBDY elements to calculate the following data:

A_i - Area factors

α - Absorbitivity

MODULE FUNCTIONAL DESCRIPTIONS

$\{n_1\}$ - Normal vector No. 1

$\{n_2\}$ - Second normal vector for elliptic cylinder

If the element is an elliptic cylinder,

$$A_i = \frac{\ell}{2} n_1 = R_2 n_1$$

$$n_2 = R_1 n_2$$

Subroutine QLØADL generates the heat flow into each grid point by the following equations:

If $\{n_2\} = \{0\}$,

$$C = \{n_1\} \cdot \{e\},$$

$$P_i = -Q_0 A_i C, C < 0,$$

$$P_i = 0, C \geq 0.$$

If $\{n_2\} \neq \{0\}$ (elliptic cylinder),

$$P_i = A_i Q_0 \sqrt{(\{e\} \cdot \{u_1\})^2 + (\{e\} \cdot \{n_2\})^2}.$$

If QLØADL is called from a transient problem, the values of $\{C\}$ are checked. If any integers are encountered, the data is written on the IQVECT file.

QVØL - Volume heat input generation. SSGSLT processes the structural elements to calculate the volume of each element and divide it among the connected grid points. (The calculation is similar to that of the mass matrix.) The loads produced by subroutine QVØL are

$$P_i = V_i Q_0,$$

where V_i is the volume associated with point i for each element.

4.41.11 Subroutines

4.41.11.1 Subroutine Name: SSG1A.

1. Entry Point: SSG1A.
2. Purpose: To build a list of external loads, thermal loads and enforced deformation loads selected by the user in CASECC.
3. Calling Sequence: CALL SSG1A (N1,ILIST,NEDT,NTEMP,NCENT,CASECC,IHARM)

N1 - Number of external loads present - integer - output.

ILIST - List of load ID's with selected load ID's set negative - integer - output.

NEDT - Number of enforced deformation loads - integer - output.

NTEMP - Number of thermal loads - integer - output.

NCENT - Not used (set to zero) - integer - output.

CASECC - GINØ number of Case Control Data Block - integer - input.

IHARM - Boundary conditions for axisymmetric shell problem.

1 = sine, 2 = cosine - integer - output.

COMMON//XX,LØADNN

LØADNN - Number of records in CASECC to skip before beginning to build loads - integer - input.

COMMON/LØADX/

LØADX - See description of /LØADX/ common block below (section 4.41.11.8).

4.41.11.2 Subroutine Name: EXTERN.

1. Entry Point: EXTERN
2. Purpose: To compute user-selected external loads.

3. Calling Sequence: CALL EXTERN (NLIST,NGRAV,GVECT,ILIST,PG,N1,IHARM)

NLIST - Number of load id's in ILIST array - integer - input.

NGRAV - Number of gravity loads selected - integer - output.

GVECT - Array of gravity vectors, 3 numbers per vector - real - output.

ILIST - List of all load id's - integer - input.

PG - Matrix control block for file on which external loads will be written - integer - input/output.

N1 - Number of external load id's - integer - input.

IHARM - Boundary condition for axisymmetric shell problem
1 = sine, 2 = cosine.

COMMON//LUSET

LUSET - Length of PG - integer - input.

COMMON/LOADX/ - See /LOADX/ description in section 4.41.11.8.

4.41.11.3 Subroutine Name: TEMPL.

1. Entry Point: TEMPL

2. Purpose: To compute thermal loads for each element.

3. Calling Sequence: CALL TEMPL (NTEMP,ILIST (N1+1),PG(1))

NTEMP - Number of thermal loads - integer - input.

ILIST(N1+1) - Beginning of thermal load list - integer - input.

PG - Matrix control block for load file - integer - input/output.

COMMON/LOADX/ - See /LOADX/ description in section 4.41.11.8.

COMMON//LUSET

LUSET - Length of PG - integer - input.

MODULE FUNCTIONAL DESCRIPTIONS

4.41.11.4 Subroutine Name: EDTL.

1. Entry Point: EDTL
2. Purpose: To compute enforced deformation loads for each element.
3. Calling Sequence: CALL EDTL (NEDT,ILIST(N1+1),PG(1))

NEDT - Number of enforced deformation loads - integer - input.

The remainder of the variables has the same meaning as in TEMPL (section 4.41.11.3).

4.41.11.5 Subroutine Name: GRAVL1.

1. Entry Point: GRAVL1
2. Purpose: To build acceleration vectors for gravity loads.
3. Calling Sequence: CALL GRAVL1 (NGRAV,GVECT,SCR1,IHARM)

NGRAV - Number of gravity loads selected - integer - input.

GVECT - Array of gravity vectors, three words per gravity vector - real - input.

SCR1 - GINØ file number on which to build the acceleration vectors - integer - input.

IHARM - Boundary condition for axisymmetric shell problem.
1 = sine, 2 = cosine - integer - output.

4.41.11.6 Subroutine Name: GRAVL2.

1. Entry Point: GRAVL2
2. Purpose: To add gravity loads onto previously generated load vectors and check scalar points.
3. Calling Sequence: CALL GRAVL2 (NGRAV,PGG,PG(1))

NGRAV - Number of gravity vectors - integer - input.

PGG - GINØ file number of gravity loads - integer - input.

PG - Matrix control block for all other (non-gravity) loads - integer - input/output.

COMMON/LØADX/

LØADX - See /LØADX/ common block (section 4.41.11.8).

FUNCTIONAL MODULE SSG1 (STATIC SOLUTION GENERATOR - PHASE 1)

COMMON//LUSET

LUSET - Length of PG vector - integer - input.

4.41.11.7 Subroutine Name: COMBIN.

1. Entry Point: COMBIN
2. Purpose: To combine subloads into subcase loads.
3. Calling Sequence: CALL COMBIN (PG,ILIST,N1)

PG - Matrix control block for PG vector - integer - input/output.

ILIST - List of all load ID's - integer - input.

N1 - Number of entries in ILIST - integer - input.

COMMON/LADX/LC,SLT,BGPD,OLD,CSTM,SIL,ISIL,EST,MPT,GPT,EDT,IMPT,IGPTT,IEC,LADF,MGG,
NBLD,DIT,ICM

SLT,BGPD,CSTM,SIL,EST,MPT,GPTT,EDT,MGG,DIT - GINØ file numbers for respective data blocks - integer - input.

LC - Length of open core - integer - input.

OLD - Current grid point position of the BGPD - integer - input.

ISIL - Current SIL position - integer.

LADF - GINØ file number of load file - integer - input.

NBLD - Build-nobuild flag for direct load routines - integer.

IMPT - }
IGPTT - } Unused at present.
IEC - }

ICM - zero if CSTM is open

COMMON LUSET//

LUSET - Length of PG vector - integer - input.

COMMON/LADS/NLADS,IARY(1080)

NLADS - Number of loads to build.

IARY - For each load: Number of subloads

Subload ID	} repeated for each subload	} repeated for each load
Scale Factor		
Subload ID	}	
Scale Factor		

MODULE FUNCTIONAL DESCRIPTIONS

4.41.11.8 Subroutine Name: Direct.

1. Entry Point: DIRECT
2. Purpose: To apply loads due to FORCE and MOMENT cards.
3. Calling Sequence: CALL DIRECT

COMMON/LOADX/

LOADX - See description of /LOADX/ above (section 4.41.11.7).

4.41.11.9 Subroutine Name TPØNT.

1. Entry Point: TPØNT

2. Purpose: To apply loads due to FØRCE1 and MØMENT1 cards.

3. Calling Sequence: CALL TPØNT

COMMON/LØADX/

LØADX - See description of /LØADX/ above (section 4.41.11.7)

4.41.11.10 Subroutine Name: FPØNT.

1. Entry Point: FPØNT

2. Purpose: To apply loads due to FØRCE2 and MØMENT2 cards.

3. Calling Sequence: CALL FPØNT

COMMON/LØADX/

LØADX - See description of /LØADX/ above (section 4.41.11.7).

4.41.11.11 Subroutine Name: SLØAD.

1. Entry Point: SLØAD

2. Purpose: To apply loads due to SLØAD cards.

3. Calling Sequence: CALL SLØAD

COMMON/LØADX/

LØADX - See description of /LØADX/ above (section 4.41.11.7).

4.41.11.12 Subroutine Name: PLØAD.

1. Entry Point: PLØAD

2. Purpose: To apply loads due to PLØAD cards.

3. Calling Sequence: CALL PLØAD

COMMON/LØADX/

LØADX - See description of /LØADX/ above (section 4.41.11.7).

MODULE FUNCTIONAL DESCRIPTIONS

4.41.11.13 Subroutine Name: RFØRCE.

1. Entry Point: RFØRCE
2. Purpose: To apply loads due to RFØRCE cards.
3. Calling Sequence: CALL RFØRCE (LCØRE)

LCØRE - Current buffer top - integer - input.

CØMMØN/LØADX/

LØADX - See description of /LØADX/ above (section 4.41.11.7).

4.41.11.14 Subroutine Name: PRESAX.

1. Entry Point: PRESAX
2. Purpose: To apply loads due to axisymmetric pressure loads.
3. Calling Sequence: CALL PRESAX (IHARM)

IHARM - Axisymmetric boundary condition - integer - input.

CØMMØN/LØADX/

LØADX - See description of /LØADX/ above (section 4.41.11.7).

4.41.11.15 Subroutine Name: GRAV.

1. Entry Point: GRAV
2. Purpose: To extract gravity vector and convert to basic coordinates.
3. Calling Sequence: CALL GRAV (NGRAV,GVECT,NEX,ILIST,NLØØP)

NGRAV - Number of gravity loads - integer - output.

GVECT - Array of gravity vectors - real - output.

NEX - Number of external loads - integer - input.

ILIST - List of external load ID's - integer - input/output.

NLØØP - Current pointer into ILIST.

CØMMØN/LØADX/

LØADX - See description of /LØADX/ above (section 4.41.11.7).

FUNCTIONAL MODULE SSG1 (STATIC SOLUTION GENERATOR - PHASE 1)

4.41.11.16 Subroutine Name: PERMUT.

1. Entry Point: PERMUT
2. Purpose: To reorder a list of grid point ID's to allow the most efficient extraction of these grid points from the BGPDT.
3. Calling Sequence: CALL PERMUT (PØNT,IØRD,NP,ØLD)

PØNT - List of points - integer - input.
IØRD - Pointers to PØNT, i.e., IØRD (1) contains subscript of PØNT which should be extracted first from the BGPDT - integer - output.

NP - Number of points - integer - output.

ØLD - Current position of BGPDT - integer - input.

4.41.11.17 Subroutine Name: FNDPNT.

1. Entry Point: FNDPNT
2. Purpose: To extract BGPDT data from the BGPDT.
3. Calling Sequence: CALL FNDPNT (BGPDD,PØNT)

BGPDD - Four-word BGPDT entry - output.
PØNT - Grid point id of desired BGPDT entry - integer - input.

4.41.11.18 Subroutine Name: CRØSS.

1. Entry Point: CRØSS
2. Purpose: To compute the cross product of two vectors.
3. Calling Sequence: CALL CRØSS (V1,V2,V3)

V1 - Three-word vector - real - input.
V2 - Three-word vector - real - input.
V3 = V1 x V2 - real - output.

4.41.11.19 Subroutine Name: NØRM.

1. Entry Point: NØRM

MODULE FUNCTIONAL DESCRIPTIONS

2. Purpose: To normalize a vector.

3. Calling Sequence: CALL NØRM (V1, XLV)

V1 - Three-word vector - real - input/output.

XLV - Norm of V1.

V1 on output = V1/XL unless XL = 0.0, then V1 = V1.

4.41.11.20 Subroutine Name: FNDSIL.

1. Entry Point: FNDSIL (in subroutine FNDPNT)

2. Purpose: To find the SIL value of a particular grid point id.

3. Calling Sequence: CALL FNDSIL (GPID)

GPID - Grid point id on input, SIL value on output - integer - input/output.

COMMON/LØADX/

LØADX - See description of /LØADX/ above (section 4.41.11.7).

4.41.11.21 Subroutine Name: BASGLB.

1. Entry Point: BASGLB

2. Purpose: To convert a vector from the basic to the global coordinate system.

3. Calling Sequence: CALL BASGLB (VIN, VØUT, GRDPNT, CSYS)

VIN - Three-word input vector - real - input.

VØUT - Three-word output vector - real - output.

VIN may equal VØUT.

GRDPNT - Location of grid point at which vector is to be applied (not used unless coordinate system type is spherical or cylindrical) - real - input.

CSYS - Coordinate system id - integer - input.

COMMON/LØADX/

LØADX - See description of /LØADX/ above (section 4.41.11.7).

4.41.11.22 Subroutine Name: GLBBAS.

1. Entry Point: GLBBAS
2. Purpose: To convert a vector from global to basic coordinates.
3. Calling Sequence: CALL GLBBAS (VIN,VOUT,GRDPNT,CSYS)

Where the variables have the same names as in BASGLB.

4.41.11.23 Subroutine Name: MPYL.

1. Entry Point: MPYL
2. Purpose: To multiply two in-core matrices together: $[A] \cdot [B] = [C]$.
3. Calling Sequence: CALL MPYL (A,B,NCOLA,NROWA,NCOLB,C)

A (NCOLA,NROWA) matrix	} These matrices are stored row-wise.
B (NCOLB,NCOLA) matrix	
C (NCOLB,NROWA) matrix	

Note: A or B \neq C

4.41.11.24 Subroutine Name: MPYLT.

1. Entry Point: MPYLT
2. Purpose: To multiply two in-core matrices together $[A]^T \cdot [B] = [C]$.
3. Calling Sequence: CALL MPYLT (A,B,NROWB,NCOLA,NCOLB,C)

A (NCOLA,NROWB) matrix	} These matrices are stored row-wise.
B (NCOLB,NROWB) matrix	
C (NCOLB,NCOLA) matrix	

4.41.11.25 Subroutine Name: FDCSTM.

1. Entry Point: FDCSTM
2. Purpose: To extract the orientation matrix from the CSTM.
3. Calling Sequence: CALL FDCSTM (ICSTM)

ICSTM - Coordinate system id desired.

MODULE FUNCTIONAL DESCRIPTIONS

COMMON/TRANX/XX(5),T0(3,3)

T0 - 3x3 orientation matrix.

COMMON/L0ADX/

L0ADX - See description of /L0ADX/ above (section 4.41.11.7).

4.41.11.26 Subroutine Name: FEDT.

1. Entry Point: FEDT (in subroutine FNDPNT)
2. Purpose: To extract one enforced deformation from the EDT given one element id.
3. Calling Sequence: CALL FEDT (EID,DELTA,IDEFM)

EID - Element id - integer - input.

DELTA - Deformation of element EID - real - output.

IDEFM - Set id of current deformation set.

4.41.11.27 Subroutine Name: GRAVL3.

1. Entry Point: GRAVL3
2. Purpose: To compute an acceleration vector for an axisymmetric shell problem.
3. Calling Sequence: CALL GRAVL3 (NGRAV,GVECT,AG,IHARM)

NGRAV - Number of gravity vectors - integer - input.

GVECT - Gravity vector array - real - input.

AG - GIN0 file number of acceleration vector - integer - input.

IHARM - Boundary condition flag

1 = sine, 2 = cosine

COMMON//LUSET

LUSET - Length of PG vector.

COMMON/SYSTEM/IX(26),MN

MN - Packed word giving number of rings/number of harmonics.

4.41.11.28 Subroutine Name: TTRIRG

1. Entry Point: TTRIRG
2. Purpose: To calculate an element thermal load vector for a triangular cross-section ring in the SSG1 module.
3. Calling Sequence: CALL TTRIRG (TI,PG)

TI - Array of four temperatures at the four points of the ring - real - input.

PG - Load vector array - real - input.

COMMON/TRIMEX/

TRIMEX - This contains the EST entry for the element.

4.41.11.29 Subroutine Name: TTRAPR.

1. Entry Point: TTRAPR
2. Purpose: To calculate an element thermal load vector for a trapezoidal cross-section ring in the SSG1 module.
3. Calling Sequence: CALL TTRAPR (TI, PG)

COMMON/TRIMEX/

The arguments are the same as in TTRIRG.

4.41.11.30 Subroutine Name: TTORDR.

1. Entry Point: TTORDR
2. Purpose: To calculate an element thermal load vector for a toroidal thin shell ring in the SSG1 module.
3. Calling Sequence: CALL TTORDR (TI,PG)

COMMON/TRIMEX/

The arguments are the same as those in TTRIRG (section 4.41.11.28).

MODULE FUNCTIONAL DESCRIPTIONS

4.41.11.31 Subroutine Name: FCURL.

1. Entry Point: FCURL
2. Purpose: To form the element thermal load matrices in field coordinates for the toroidal thin shell ring in subroutine TTØRDR.
3. Calling Sequence: CALL FCURL (FMEØ,FME1,FFEØ,FFE1,YI,S,LAM1)

FMEØ, FME1 The resultant thermal load matrices.
FFEØ, FFE1

YI - Array of integral values.

S, LAM1 - Terms used in the evaluation of the thermal load matrices.

4.41.11.32 Subroutine Name: CØNE.

1. Entry Point: CØNE
2. Purpose: To calculate an element thermal load vector for an axisymmetric shell in the SSG1 module.
3. Calling Sequence: CALL CØNE (TI,PG)

CØMMØN/TRIMEX/

The arguments are the same as in TTRIG (section 4.41.11.28).

4.41.11.33 Subroutine Name: QDMEM.

1. Entry Point: QDMEM
2. Purpose: To calculate an element thermal load vector for quadrilateral elements in the SSG1 module.
3. Calling Sequence: CALL QDMEM (TI,PG)

CØMMØN/TRIMEX/

The arguments are the same as in TTRIG (section 4.41.11.28).

4.41.11.34 Subroutine Name: TRIMEM.

1. Entry Point: TRIMEM

FUNCTIONAL MODULE SSG1 (STATIC SOLUTION GENERATOR - PHASE 1)

2. Purpose: To calculate an element thermal load vector for triangular elements in the SSG1 module.

3. Calling Sequence: CALL TRIMEM (TYPE,TBAR,PG)

COMMON/TRIMEX/

TYPE - Flag indicating normal entry (0) or subelement call from a QDMEM (1) - integer - input.

TBAR - Average temperature over the element - real - input.

PG and /TRIMEX/ are as in TTRIRG (section 4.41.11.28).

4.41.11.35 Subroutine Name: BAR.

1. Entry Point: BAR

2. Purpose: To calculate an element thermal load vector or deformation load vector for the bar element in the SSG1 module.

3. Calling Sequence: CALL BAR (PG, IDEFM, ITEMP, IDEFT)

COMMON/TRIMEX/

IDEFM - 0 if element deformation load vector is not to be computed - integer - input.

ITEMP - 0 if element thermal load vector is not to be computed - integer - input.

IDEFT - Set ID of the element deformation set. This is used only if IDEFM \neq 0 - integer - input.

PG,/TRIMEX/ are as in Section 4.41.11.28).

4.41.11.36 Subroutine Name: FEDTST

1. Entry Point: FEDTST (in FNDPNT)

2. Purpose: To put in core the element id's and associated values for a given deformation set from the EDT.

3. Calling sequence: CALL FEDTST (IDEF)

IDEF - Set id of current deformation

COMMON /FPT/DUM(3),NR0W1,LC0RE

NR0W - Number of words of open core used.

MODULE FUNCTIONAL DESCRIPTIONS

LCØRE - End of available open core.

CØMMØN /SSG1BX/CØRE

CØRE - First cell of open core available.

4.41.11.37 Subroutine Name: FEDTED

1. Entry Point: FEDTED (in FNDPNT)
2. Purpose: To determine if all elements in a selected deformation set were used.
3. Calling Sequence: CALL FEDTED (IDEF)
IDEF - Set id of current deformation set.

4.41.11.38 Subroutine Name: HBDY

1. Entry Point: HBDY.
2. Purpose: To calculate heat transfer flux loads due to convective film coefficients and temperatures of the surrounding fluid (heat transfer analysis only).
3. Calling sequence: CALL HBDY.

4.41.11.39 Subroutine Name: QHBDY

1. Entry Point: QHBDY.
2. Purpose: To generate heat flux loads in a heat transfer problem.
3. Calling sequence: CALL QHBDY.

4.41.11.40 Subroutine Name: QDPLT

1. Entry Point: QDPLT
2. Purpose: To generate the element thermal load data for quadrilateral plate elements.
3. Calling Sequence: CALL QDPLT (TI)
TI - Array of grid point temperatures
CØMMØN /TRIMEX/

4.41.11.41 Subroutine Name: RØD

1. Entry Point: RØD

FUNCTIONAL MODULE SSG1 (STATIC SOLUTION GENERATOR - PHASE 1)

2. Purpose: To generate the element thermal load data and enforced deformation load data for the RØD, CØNRØD and TUBE elements.

3. Calling Sequence: CALL RØD

CØMMØN /TRIMEX/

4.41.11.42 Subroutine Name: SØLID

1. Entry Point: SØLID

2. Purpose: To generate the element thermal load data for all solid elements except TETRA.

3. Calling Sequence: CALL SØLID (T,P,I)

T = Temperature vector

P = Load vector

I = 1, element is WEDGE
2, element is HEXA1
3, element is HEXA2

4. Method: Calls are made to TETRA.

4.41.11.43 Subroutine Name: TETRA

1. Entry Point: TETRA

2. Purpose: Computes element thermal load data for tetrahedra either directly (for the TETRA element) or indirectly via SØLID (for the other solid elements).

3. Calling Sequence: CALL TETRA (T,P,K)

T = Temperature vector

P = Load vector

K = Option Flag

= 0, divide by 6.0

≠ 0, divide by 12.0

4.41.11.44 Subroutine Name: TRBSC

1. Entry Point: TRBSC

2. Purpose: To compute the thermal load data for the basic bending triangle element TRBSC.

MODULE FUNCTIONAL DESCRIPTIONS

3. Calling Sequence: CALL TRBSC (I,T)

I = 0 (basic bending triangle)
1 (sub-computations for SQDPL1)
2 (sub-computations for STRPL1)

T = Temperature vector

4.41.11.45 Subroutine Name: TRIQD

1. Entry Point: TRIQD
2. Purpose: To compute the thermal load data for the triangular and quadrilateral elements.
3. Calling Sequence: CALL TRIQD (N,T)

N = 1 implies TRIA1
2 implies TRIA2
3 implies QUAD1
4 implies QUAD2

T = Temperature vector

4.41.11.46 Subroutine Name: TRPLT

1. Entry Point: TRPLT
2. Purpose: To compute the thermal load data for the triangular bending elements.
3. Calling Sequence: CALL TRPLT (T)

T = Temperature vector

4.41.11.47 Subroutine Name: SSGHKI

1. Entry Point: SSGHKI
2. Purpose: Computes element thermal load and enforced deformation load data for use by TRBSC, TRPLT and QDPLT.
3. Calling Sequence: CALL SSGHKI (TR,TI,FN)

TR - Real Temperature Vector
TI - Integer Temperature Vector
FN - Multiplication fraction

FUNCTIONAL MODULE SSG1 (STATIC SOLUTION GENERATOR - PHASE 1)

4.41.11.48 Subroutine Name: SSGETD

1. Entry Point: SSGETD
2. Purpose: Computes element temperature from a pre-positioned record.
3. Calling Sequence: CALL SSGETD (E,T,G)
 - E - Element identification number for which temperature is desired.
 - T - Area into which temperature data is returned.
 - T(1) - Average temperature
 - T(2) - Temperature at G.P. 1
 - T(3) - Temperature at G.P. 2
 - ⋮
 - T_{n+1} - Temperature at n+1
 - G - =0, element temperature format data is desired
≠0, number of grid points.

4.41.11.49 Subroutine Name: SSGSLT

1. Entry Point: SSGSLT
2. Purpose: To convert applied loads referencing elements to applied load factors and connected grid points.
3. Calling Sequence: CALL SSGSLT (SLT,NEWSLT,EST), where:
 - SLT - Input, file number of Static Loads Table
 - NEWSLT - Output, file number of converted SLT.
 - EST - Input, file number of Element Summary Table.
4. Method: The SLT is read one load set at a time. Normal static loads are copied onto the NEWSLT file. If element-dependent loads are encountered, they are stored in core. The EST data block is read and each element is checked to find a load applied to it. Utility subroutine HBDY is used to calculate the coefficients for the HBDY elements. When all elements have been processed, the converted data is written on the load set record of NEWSLT. The process repeats for each load set.

4.41.11.50 Subroutine Name: QVØL

1. Entry Point: QVØL

MODULE FUNCTIONAL DESCRIPTIONS

2. Purpose: To calculate the applied heat flow loads due to a uniform internal heat generation in an element.

3. Calling Sequence: CALL QVØL

4.41.11.51 Subroutine Name: QLØADL

1. Entry Point: QLØADL

2. Purpose: To compute heat transfer applied loads on the HBDY element data due to QBDY1, QBDY2, and QVECT load cards.

3. Calling Sequence: CALL QLØADL (TYPE), where:

TYPE = 1 - QBDY1 data

TYPE = 2 - QBDY2 data

TYPE = 3 - QVECT data

/QVECT/ITRAN,IQVECT

ITRAN - Input, indicates transient problem.

IQVECT - File number of output IQVECT data.

4.41.11.52 Subroutine Name: QDMM1

1. Entry Point: QDMM1

2. Purpose: Computes thermal load for QDMEM1 element.

3. Calling Sequence: CALL QDMM1 (TI,CØRE(1))

TI - Element temperature above the reference state

CØRE(1) - Thermal load vector.

4.41.11.53 Subroutine Name: PLØAD3

1. Entry Point: PLØAD3

2. Purpose: To apply loads due to PLØAD3 cards

3. Calling Sequence: CALL PLØAD3

CØMMØN /LØADX/

4.41.11.54 Subroutine Name: IHEX

1. Entry Point: IHEX
2. Purpose: To calculate an element thermal load vector for the IHEX1, IHEX2, and IHEX3 elements in the SSG1 module.
3. Calling Sequence: CALL IHEX (TEMPS,PG,TYPE)
COMMON/TRIMEX/
TEMPS - Array of grid point temperatures - real-input.
PG - Load vector array - real-input
TYPE = $\begin{Bmatrix} 1 - \text{IHEX1} \\ 2 - \text{IHEX2} \\ 3 - \text{IHEX3} \end{Bmatrix}$ - integer-input
TRIMEX - EST entry for the element - mixed-input

4.41.11.55 Subroutine Name: TQUADS

1. Entry Point: TQUADS
2. Purpose: To calculate an element thermal load vector for a quadrilateral thin shell element
3. Calling Sequence: CALL TQUADS (TI,PG)
TI - Array of four temperatures at the four corner points of the quadrilateral-
real-input
PG - Load vector array-real-input
COMMON/TRIMEX/ contains the EST for the element

4.41.11.56 Subroutine Name: TTRIAS

1. Entry Point: TTRIAS
2. Purpose: To calculate an element thermal load vector for a triangular thin shell element.
3. Calling Sequence: CALL TTRIAS (TI,PG)
TI - Array of three temperatures at the three corner points of the triangle
real-input
PG - Load vector array-real-input
COMMON/TRIMEX/ contains the EST for the element

MODULE FUNCTIONAL DESCRIPTIONS

4.41.11.57 Subroutine Name: TPZTEM

1. Entry Point: TPZTEM
2. Purpose: To calculate an element load vector for an axisymmetric trapezoidal ring element in the SSG1 module
3. Calling Sequence: TPZTEM (TI,PG)
COMMON /TRIMEX/
The arguments are the same as in TTRIRG

4.41.11.58 Subroutine Name: TRTTEM

1. Entry Point: TRTTEM
2. Purpose: To calculate an element load vector for an axisymmetric triangular ring element in the SSG1 module
3. Calling Sequence: TRTTEM (TI,PG)
COMMON/TRIMEX/
The arguments are the same as in TTRIRG

4.41.11.59 Subroutine Name: ELTKL

1. Entry Point: ELTKL
2. Purpose: To evaluate integrals for the axisymmetric triangular and trapezoidal ring elements in subroutines TRZTEM and TRTTEM.
3. Calling Sequence: ELTKL (A,RZ,K)
A - Array of fifteen locations containing the results
R,Z - Array of four locations each containing the R and Z coordinates of all points used to describe the area of integration
K = 0 for triangular ring
1 for trapezoidal ring

FUNCTIONAL MODULE SSG1 (STATIC SOLUTION GENERATOR - PHASE 1)

4.41.11.60 Subroutine Name: BASGLB

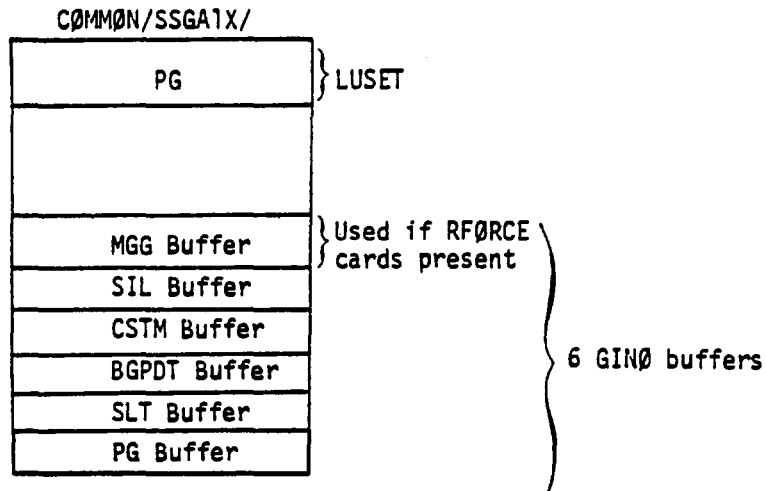
1. Entry Point: GBTRAN
2. Purpose: Finds a Global to Basic transformation matrix stored by rows.
3. Calling Sequence: CALL GBTRAN (IC,P,T)
IC - Coordinate system identification number.
P - Location of grid point at which vector is to be applied.
T - Transformation matrix stored by rows.

4.41.12 Design Requirements

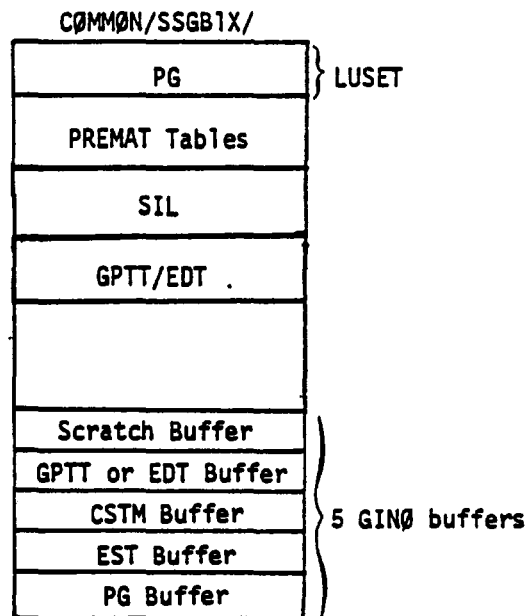
Three scratch files are needed.

Open core is defined as follows:

1. /SSGA1X/ during EXTERN phase



2. /SSGB1X/ during EDTL, TEMPL phase



100 LOAD ID's maximum.

4.41.13 Diagnostic Messages

If the core storage requirements as depicted in the above diagrams are not met, SSG will issue fatal error messages 3008 and 2143.

FUNCTIONAL MODULE SSG2 (STATIC SOLUTION GENERATOR - PHASE 2)

4.42 FUNCTIONAL MODULE SSG2 (STATIC SOLUTION GENERATOR - PHASE 2)

4.42.1 Entry Point: SSG2

4.42.2 Purpose

To reduce the applied load vectors and enforced displacements into equivalent load vectors applied to the independent displacement coordinate sets.

4.42.3 DMAP Calling Sequence

SSG2 {H USET}, GM, YS, KFS, GØ, DM, PG/ {QR}, {PØ}, {PS}, {PL}
 {USET}, {HQR}, {HPØ}, {HPS}, {HPL} \$
 {HPF}

4.42.4 Input Data Blocks

HUSET - Temperature set definitions table for heat problems.

USET - Displacement set definitions table.

GM - Multipoint constraint transformation matrix - m set.

YS - Constrained displacements - s set.

KFS - Partition of stiffness matrix after single-point constraints have been removed.

G_0 - Structural matrix partitioning transformation matrix.

DM - Rigid body transformation matrix.

PG - Static load vector matrix giving static loads - q set.

Notes: 1. USET must be present.

2. GM must be present if s set is not null.

3. YS must be present if s set is not null.

4. KFS must be present if s set is not null.

5. G_0 must be present if a set is not null.

6. DM must be present if r set is not null.

7. PG must be present.

4.42.5 Output Data Blocks

HQR } - Determinate support forces matrix - r set.
OR }

HP0 } - Partition of the load vector matrix giving loads due to static force - o set.
P0 }

MODULE FUNCTIONAL DESCRIPTIONS

HPS - Partition of load vector matrix giving loads in s set.
PS

HPF
HPL - Partition of load vector matrix giving static loads on l set.
PL

- Notes:
1. QR must be present if r set is non-null.
 2. PØ must be present if o set is non-null.
 3. PS must be present if s set is non-null.
 4. PL must be present if l set is non-null.
 5. If the problem has no sets, SSG2 will return.

4.42.6 Parameters

None

4.42.7 Method

The fifth word of the USET trailer control block is analyzed to determine the presence of m's, s's, o's, or r's.

Each of the following steps is omitted if the appropriate set is null. The following steps indicate the operations on one load set vector. The actual algorithm uses all vectors in each step by performing matrix operations.

1. If m's are present, the PG vectors are partitioned using USET (UG,UN,UM) and subroutines CALCV and SSG2A:

$$\{P_g\} \Rightarrow \begin{Bmatrix} \bar{P}_n \\ \bar{P}_m \end{Bmatrix} . \quad (1)$$

The loads on the u_n set are calculated as:

$$\{P_n\} = \{\bar{P}_n\} + [G_m]^T \{P_m\} , \quad (2)$$

by calling subroutine SSG2B.

2. If s's are present, the $\{P_n\}$ load vectors are partitioned using USET (UN,UF,US) and subroutines CALCV, and SSG2A:

$$\{P_n\} \Rightarrow \begin{Bmatrix} \bar{P}_f \\ -\bar{P}_s \end{Bmatrix} \quad (3)$$

The $\{P_s\}$ vectors are output in data block PS. The loads on the u_f set are calculated as:

$$\{P_f\} = \{\bar{P}_f\} - [K_{fs}] \{Y_s\} . \quad (4)$$

The $\{Y_s\}$ vector normally is zero. If more than one load vector is being reduced, $\{Y_s\}$ is expanded to have N identical columns such that the above matrix equation is dimensionally consistent. This is accomplished in subroutine SSG2B1 (see below).

3. If o's are present, the $\{P_f\}$ vectors are partitioned using USET (UF,UA,UØ) and subroutines CALCV and SSG2A:

$$\{P_f\} \Rightarrow \left\{ \begin{array}{c} \bar{P}_a \\ \bar{P}_o \end{array} \right\} . \quad (5)$$

$\{P_o\}$ vectors are written on data block PØ.

The equivalent loads on the u_a set are calculated as:

$$\{P_a\} = \{\bar{P}_a\} + [G_o]^T \{P_o\} . \quad (6)$$

Subroutine SSG2B is called to perform this operation.

4. If r's are present, the $\{P_a\}$ vectors are partitioned using USET (UA,UL,UR) and subroutines CALCV and SSG2A:

$$\{P_a\} \Rightarrow \left\{ \begin{array}{c} P_l \\ P_r \end{array} \right\} . \quad (7)$$

$\{P_l\}$ vectors are written on data block PL.

The reaction vectors on the support points are:

$$\{q_r\} = - \{P_r\} - [D_m]^T \{P_f\} \quad (8)$$

where $[D_m]$ corresponds to the data block DM. $\{q_r\}$ vectors are written on data block QR.

4.42.8 Subroutines

SSG2 uses matrix subroutines CALCV and SSG2A for matrix partitioning operations and subroutine SSG2B to drive subroutine MPYAD. See section 3 for details.

MODULE FUNCTIONAL DESCRIPTIONS

4.42.9 Design Requirements

/SSG2X/ is open core for CALCV. Four scratch files are used.

SSG2B1 is an entry point in SSG2B.

FUNCTIONAL MODULE SSG3 (STATIC SOLUTION GENERATOR - PHASE 3)

4.43 FUNCTIONAL MODULE SSG3 (STATIC SOLUTION GENERATOR - PHASE 3)

4.43.1 Entry Point: SSG3

4.43.2 Purpose:

To perform the actual static solutions. A displacement solution is produced for each applied load and tested for possible matrix decomposition errors.

4.43.3 DMAP Calling Sequence

SSG3 { LLL } { KLL } { PL } { IL00 } { K00 } { P0 } { ULV } { U00V } { RULV } { RU0V } /V,N,0MIT/
 { LBLL } { KAA } { PLI } { HL00 } { HK00 } { P0I } { UBLV } { HU00V } { RUBLV } { HRU0V }
 { HLLL } { HKLL } { HPL } { HULV } { HRULV }

V,Y,IRES/V,N,NSKIP/V,N,EPSI \$

4.43.4 Input Data Blocks

LLL }
 LBLL } - Lower triangular factor of KLL - 2 set.
 HLLL }

KLL }
 KBLL } - Partition of stiffness matrix - 2 set.
 HKLL }

KAA - Partition of stiffness matrix - a set.

PL }
 PLI } - Partition of the load vector matrix giving static loads on 2 set.
 PBL }
 HPL }

L00 }
 HL00 } - Lower triangular factor of K00 - 0 set.

K00 }
 HK00 } - Partition of stiffness matrix - 0 set.

P0 }
 P0I } - Partition of the load vector matrix giving loads due to static forces - 0 set.

- Notes:
1. ULL,LLL and PL must be present.
 2. KLL can be purged if RULV is purged.
 3. U00, L00 P0 can be purged if 0MIT < 0.
 4. K00 can be purged if 0MIT < 0 or RU0V is purged.

MODULE FUNCTIONAL DESCRIPTIONS

4.43.5 Output Data Blocks

ULV
UBLV - Partition of the displacement vector matrix giving displacements - l set.
HULV

UØØV - Partition of the displacement vector matrix giving displacements - o set.

RULV
RUBLV - Residual vector matrix for the l set.
HRULV

RUØV
HRUØV - Residual vector matrix for the o set.

HUØØV - Partition of the temperature vector matrix giving temperatures - o set.

HULV - Partition of the temperature vector matrix giving temperatures - l set.

- Notes:
1. ULV must be present.
 2. UØØV can be purged if ØMIT < 0.
 3. RULV and RUØV can be purged.
 4. $[RULV] = [KLL] [ULV] - [PL]$.
 $[RUØV] = [KØØ] [UØØV] - [PØ]$.

4.43.6 Parameters

ØMIT - Input-integer-no default. ØMIT controls operations on o-set matrices.

IRES - Not used-integer-no default. IRES is a user-controlled parameter that he may set to +1 on a Bulk Data PARAM card so that residual vectors may be printed. It is included here to define an initial value for IRES.

NSKIP - Input-integer-default = 0 - Identifies load vector numbers for diagnostic printout.

EPSI - Output-real-default = 0.0 - Value of total residual error, \sum_e , of last vector.

4.43.7 Method

4.43.7.1 Solution Algorithm

For normal statics problems PL and PØ vectors are used; for inertia relief problems the PLI and PØI vectors are generated in the SSG4 module and used instead of PL and PØ. The equations to be solved are:

$$[K_{ll}] \{u_l\} = \{P_l\}, \quad (1)$$

FUNCTIONAL MODULE SSG4 (STATIC SOLUTION GENERATOR - PHASE 4)

and if o's are present ($\emptyset MIT \geq 0$)

$$[K_{oo}] [u_o^0] = \{P_o\}, \quad (2)$$

where

$$[K_{\ell\ell}] = [L_{\ell\ell}] [U_{\ell\ell}], \quad (3)$$

$$[K_{oo}] = [L_{oo}] [U_{oo}]. \quad (4)$$

$[L_{\ell\ell}]$ and $[L_{oo}]$ are lower traingular matrices. $[U_{\ell\ell}]$ and $[U_{oo}]$ are upper triangular matrices. Note that $[U_{\ell\ell}]$ and $[U_{oo}]$ are obtained from the $[L_{\ell\ell}]$ and $[L_{oo}]$ files.

The equations are solved in the following manner:

$$[L] [U] \{u\} = \{P\}, \quad (5)$$

or

$$[L] \{y\} = \{P\}. \quad (6)$$

In turn the solution for the displacement vector is:

$$[U] \{u\} = \{y\}. \quad (7)$$

4.43.7.2 Error Check Algorithm

If RULV is not purged, an error check is made. Since it is possible for the stiffness matrices to be nearly singular or ill-conditioned, the module calculates the following error analysis terms for both the $\{u_\ell\}$ and $\{u_o\}$ solutions:

$$\{\delta P_\ell\} = \{P_\ell\} - [K_{\ell\ell}] \{u_\ell\}, \quad (8)$$

$$\epsilon_e = \frac{\{u_\ell\}^T \{\delta P_\ell\}}{\{P_\ell\}^T \{u_\ell\}}. \quad (9)$$

$\{\delta P_\ell\}$ forms data block RULV or RUØV. ϵ_e is printed for each solution.

MODULE FUNCTIONAL DESCRIPTIONS

4.43.8 Subroutines

SSG3A - is the only auxiliary subroutine in module SSG3 (see section 3.5.18 for description).

4.43.9 Design Requirements

Two scratch files are needed.

4.44 FUNCTIONAL MODULE SSG4 (STATIC SOLUTION GENERATOR - PHASE 4).

4.44.1 Entry Point: SSG4

4.44.2 Purpose

The purpose of this module is to calculate mass loads in a Static Analysis with Inertia Relief problem. The rigid body accelerations are functions of the reactions on the fictitious supports. The inertia loads on the structure are proportional to these accelerations.

4.44.3 DMAP Calling Sequence

SSG4 PL,QR,P0,MR,MLR,DM,MLL,M00,M0A,G0,USET/PLI,P0I/V,N,0MIT \$

4.44.4 Input Data Blocks

- PL - Partition of the load vector matrix giving static loads - 2 set.
- QR - Determinate support forces matrix - r set.
- P0 - Partition of the load vector matrix giving loads due to static force - o set.
- MR - Rigid body mass matrix - r set.
- MLR - Partition of mass matrix.
- DM - Rigid body transformation matrix.
- MLL - Partition of mass matrix - 2 set.
- M00 - Partition of mass matrix - o set.
- M0A - Partition of mass matrix.
- G0 - Structural matrix partitioning transformation matrix.
- USET - Displacement set definitions table.

Note: All matrices must be present if their appropriate set is non-null.

4.44.5 Output Data Blocks

- PLI - Partition of load vector for inertia relief matrix giving loads due to static and inertial forces on 2 set.

P0I - Partition of load vector for inertia relief matrix giving loads due to inertial and static forces on o set.

Note: Both matrices must be present if their appropriate set is non-null.

4.44.6 Parameters

ØMIT - Input-integer-no default. If ØMIT > 0, the o set is non-null.

4.44.7 Method

1. The accelerations of the u_r degrees of freedom are:

$$\{a_r\} = - [m_r]^{-1} \{q_r\}. \quad (1)$$

$[m_r]$ corresponds to data block MR. Subroutines FACTØR and SSG3A are used to solve for $\{a_r\}$.

2. The total load vectors on the structure are:

$$\{P_\ell^i\} = \{P_\ell\} + [M_{\ell\ell}] [D] + [M_{\ell r}] \{a_r\}. \quad (2)$$

$[D]$ corresponds to data block DM. Subroutine SSG2B is used to drive MPYAD.

3. If ØMIT \geq 0:

$$\{P_o^i\} = \{P_o\} + [M_{oo}] [G_o] + [M_{oa}] \begin{bmatrix} D \\ -I \end{bmatrix} \{a_r\}. \quad (3)$$

The product $\begin{bmatrix} D \\ -I \end{bmatrix} \{a_r\}$ is formed by merging columns of $[D] \{a_r\}$ with $\{a_r\}$ using USET (UA,UL,UR).

Subroutine SDR1B is used to drive MERGE, and SSG2B for the matrix products.

4.44.8 Subroutines

SSG2B - See subroutine description, section 3.5.13

SSG3A - See subroutine description, section 3.5.18

SDR1B - See subroutine description, section 3.5.8

4.44.9 Design Requirements

Five scratch files are necessary.

FUNCTIONAL MODULE SDR1 (STRESS DATA RECOVERY - PHASE 1)

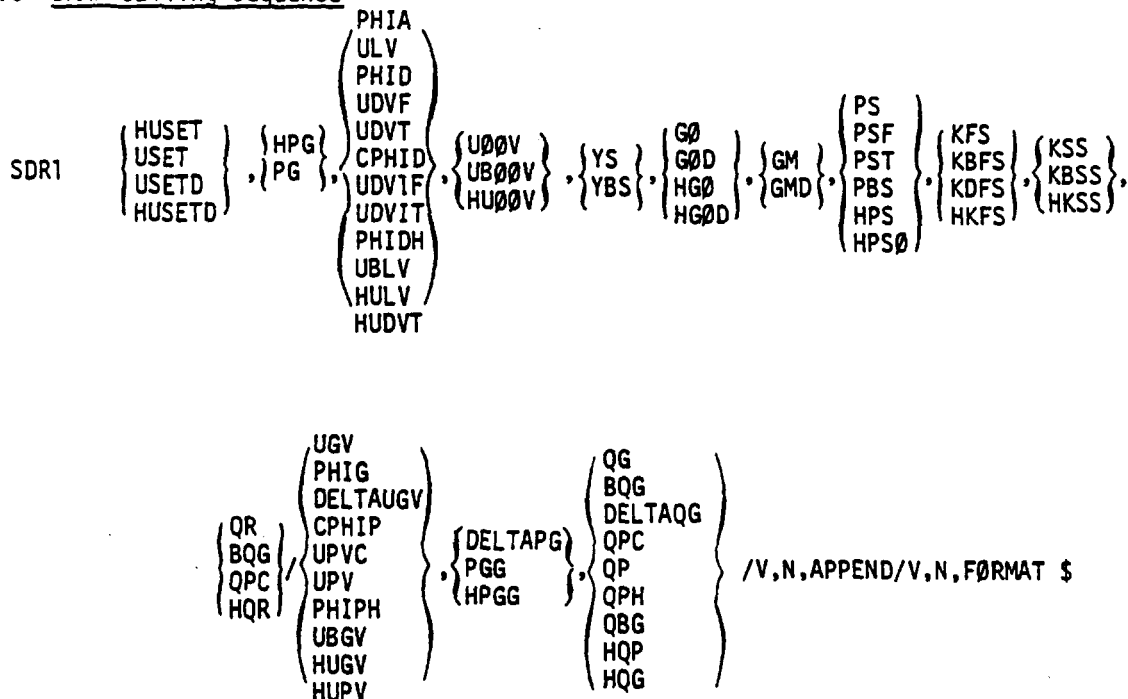
4.45 FUNCTIONAL MODULE SDR1 (STRESS DATA RECOVERY - PHASE 1)

4.45.1 Entry Point: SDR1

4.45.2 Purpose

The SDR1 module utilizes solution vectors to produce displacements, eigenvectors, velocities, accelerations, applied loads and reaction loads. The vectors input to SDR1 are in the form of packed matrices with each column a solution vector for a different subcase; eigenvalue, frequency/load, or transient output time. The row position of each term in a vector corresponds to a degree of freedom in a unique displacement set. The relative position of the term must be converted to a relative position in the vector which includes all displacement components in the system. The dependent components of the displacement vector are recovered and merged to produce a complete vector describing all degrees of freedom in the structural or dynamics model. In the Static Analysis or Static Analysis with Inertia Relief Rigid Formats, SDR1 collects solutions for each boundary condition onto a single file, convenient for the solution of symmetry problems.

4.45.3 DMAP Calling Sequence



MODULE FUNCTIONAL DESCRIPTIONS

4.45.4 Input Data Blocks

- USET } - Displacement set definition table.
- USETD }
- PG - Static load vector matrix giving static loads - g set.
- PHIA } - Partition of the displacement vector matrix giving displacements in the l set
- ULV } (d set for transient response).
- PHID }
- UDVF }
- UDVT }
- CPHID }
- UDVIF }
- UDVIT }
- PHIDH }
- UBLF }
- U00V } - Partition of the displacement vector matrix giving displacements in the o set.
- UB00V }
- YS } - Constrained displacements - s set.
- YBS }
- G0 } - Structural matrix partitioning transformation matrix.
- G0D }
- HG0 }
- HG0D }
- GM } - Rigid body transformation matrix.
- GMD }
- HPS } - Partition of load vector matrix giving loads in - s set.
- HPS0 }
- PS }
- PSF }
- PST }
- PBS }
- HUSET } - Temperature set definition table.
- HUSETD }
- HULV } - Partition of the temperature vector matrix giving temperatures in the l set
- HUDUT }
- (d set for transient response)
- KFS } - Partition of stiffness matrix after single-point constraints have been removed.
- KBFS }
- KDFS }
- KSS } - Partition of stiffness matrix after single-point constraints have been removed -
- KBSS }
- s set.
- HQR } - Determinant support forces matrix - r set.
- QR }
- BQG }
- QPC }
- HU00V - Partition of the temperature vector matrix giving temperatures in the o set.
- HKFS - Partition of conductivity matrix after single-point constraints have been removed.
- HKSS - Partition of conductivity matrix after single-point constraints have been removed -
- s set.

FUNCTIONAL MODULE SDR1 (STRESS DATA RECOVERY - PHASE 1)

- Notes:
1. The first input block must always be present.
 2. The second input block may or may not be present.
 3. The third input block must always be present.
 4. The fourth input block must be present unless the o set is null or FØRMAT = DYNAMICS (see below).
 5. The fifth input block may or may not be present.
 6. The sixth input block must be present unless the o set is null.
 7. The seventh input block must be present unless the m set is null.
 8. The eighth input block may or may not be present.
 9. The ninth input block must be present unless the s set or the third output block is not present.
 10. The tenth input block must be present unless the fifth input block is absent or the s set is null or the third output block is not present.
 11. The eleventh input block may or may not be present.

4.45.5 Output Data Blocks

UGV PHIG DELTAUGV CPHIP UPVC UPV PHIPH UBGV	}	- Displacement vector matrix giving displacements in the g set.
DELTAPG PGG HGG	}	- Static load vector appended to include all boundary conditions - g set.
QG BQG DELTAQG QPC QP QPH QBG HQG HQP	}	- Single-point constraint forces and determinate support forces matrix - g set.
HUPV HUGV	}	- Temperature vector matrix giving temperatures in the g set.

4.45.6 Parameters

- APPEND - Input-integer-no default. See note 4 above.
- FØRMAT - Input-BCD-no default. Format indicates the problem type.
- STATICS - Statics type problem.
- REIGEN - Real Eigenvalue problem.
- DYNAMICS - Dynamic problem.

4.45.7 Method

The following steps are performed by the SDR1 module in the most general case. Most problems, however, do not use all of the constraint options. In these cases certain steps are skipped. The task SDR1 performs also varies from Rigid Format to Rigid Format.

1. If PG is present, it is copied or appended onto PGG. (Subroutine SDR1A).
2. If r's are present and the problem type is not DYNAMICS or REIGEN:

$$\left\{ \frac{u_r}{o} \right\} \Rightarrow \left\{ u_a \right\} . \quad (1)$$

This is accomplished by subroutine SDR1B, the driver for MERGE. See section 3.5.8 for details.

If s's are also present:

$$\left\{ \frac{q_r}{o} \right\} \Rightarrow \left\{ q_f \right\} . \quad (2)$$

Otherwise,

$$\left\{ \frac{q_r}{o} \right\} \Rightarrow \left\{ q_g \right\} . \quad (3)$$

which is then copied or appended onto QG. (Subroutine SDR1B and SDR1A).

3. If o's are present, the dependent degrees of freedom of the "omitted" coordinates $\{u_o\}$ are computed. For statics problems:

$$\{u_o\} = [G_o] \{u_a\} + \{u_o^o\} . \quad (4)$$

For Dynamic problems:

$$\{u_o\} = [G_o^d] [u_d] . \quad (5)$$

This is accomplished by calling SSG2B (see section 3.5.13). The independent degrees of freedom $\{u_a\}$ or $\{u_d\}$ are merged with the dependent coordinates $\{u_o\}$, using subroutine SDR1B:

$$\left\{ \begin{matrix} u_a \\ u_o \end{matrix} \right\} \Rightarrow \{u_f\} . \quad (6)$$

4. If s's are present, the coordinates fixed by single-point constraints, $\{u_s\}$, may have constrained displacements, $\{Y_s\}$, in a statics problem. These are merged with the $\{u_f\}$ vector using subroutine SDR1B:

$$\left\{ \begin{matrix} u_f \\ -Y_s \end{matrix} \right\} \Rightarrow \{u_n\} \quad (7)$$

The forces of single-point constraint $\{q_s\}$ are calculated using SSG2B from the equation:

$$\{q_s\} = -\{P_s\} + [K_{fs}]^T \{u_f\} + [K_{ss}] \{Y_s\} . \quad (8)$$

Subroutine SDR1B is used to merge $\{q_s\}$ with $\{q_f\}$ to form $\{q_g\}$. $\{q_f\}$ is $\{q_r\}$ expanded to the f-set:

$$\left\{ \begin{matrix} q_f \\ q_s \end{matrix} \right\} \Rightarrow \{q_g\} . \quad (9)$$

$\{q_g\}$ is copied or appended onto data block QG (Subroutine SDR1A). If YS is not present,

$$\left\{ \begin{matrix} u_f \\ u_o \end{matrix} \right\} \Rightarrow \{u_n\} . \quad (10)$$

MODULE FUNCTIONAL DESCRIPTIONS

The forces of single-point constraint are computed as in Equation 7 except the $\{Y_s\}$ term is omitted.

If e-points (extra points) are present in a dynamics problem, $[K_{fs}]$ must be converted to $[K_{fs}^d]$ by subroutine SDR1C. If the problem is a transient problem, $\{u_f\}$ must be reduced to displacement vectors only, since $\{u_f\}$ contains triples - displacement, velocity and acceleration in transient problems. This reduction is accomplished in SDR1D.

5. If m's are present, the dependent coordinates, $\{u_m\}$, of the multipoint constraint equations are calculated using:

$$\{u_m\} = [G_m] \{u_n\} . \quad (11)$$

In dynamics problems $[G_m] = [G_m^d]$. The two vectors $\{u_m\}$ and $\{u_n\}$ are merged

$$\left\{ \begin{matrix} u_n \\ u_m \end{matrix} \right\} \Rightarrow \{u_g\} . \quad (12)$$

6. $\{u_g\}$ (or $\{u_p\}$ for dynamics problems) is copied or appended onto UGV.

4.45.8 Subroutines

SSG2B and SDR1B are used as utility routines. See section 3.5.13 and 3.5.8 for details.

4.45.8.1 Subroutine Name: SDR1A

1. Entry Point: SDR1A
2. Purpose: To copy or append vectors.
3. Calling Sequence: CALL SDR1A (IN,IOUT)

IN - GINØ file number of input file - integer - input.

IØUT - GINØ file number of output file - integer - input.

COMMON//IAPEND- Append flag. If IAPEND \neq 1, IØUT will be positioned after the last record and IN copied onto IØUT at this point. Otherwise IN is copied onto IØUT.

4.45.8.2 Subroutine Name: SDR1C

1. Entry Point: SDR1C
2. Purpose: To expand a file (KFS) from the f set to the fe set for dynamics.
3. Calling Sequence: CALL SDR1C (IPVECT,KFS,KDFS)
 IPVECT - GINØ file number of the partitioning vector previously generated - integer - input.
 KFS - GINØ file number of $[K_{fs}]$ - integer - input.
 KDFS - GINØ file number of $[K_{fs}^d]$ - integer - input.
4. Design Requirement: SDR1C depends on a previous call to SDR1B to initialize set parameters.

4.45.8.3 Subroutine Name: SDR1D

1. Entry Point: SDR1D
2. Purpose: To strip velocity and acceleration vectors from $\{u_f\}$.
3. Calling Sequence: CALL SDR1D (PS,IUF,IUF1,ITRAN)
 PS - GINØ file number of PS - integer - input.
 IUF - GINØ file number of $\{u_f\}$ - integer - input.
 IUF1 - GINØ file number of $\{u_f\}$ stripped - integer - input.
 ITRAN - Flag which shows whether problem is a transient analysis or not - integer - input. Its values are:
 1 \Rightarrow not transient
 0 \Rightarrow transient.

4.45.9 Design Requirements

Six scratch files are required.

FUNCTIONAL MODULE SDR2 (STRESS DATA RECOVERY - PHASE 2)

4.46 FUNCTIONAL MODULE SDR2 (STRESS DATA RECOVERY - PHASE 2)

4.46.1 Entry Point: SDR2

4.46.2 Purpose

The SDR2 module processes the output requests for forces of single-point constraint, loads, point displacements, point velocities, point accelerations, element stresses or element strains/curvatures, and element forces, formatting the output data blocks with these final output results for: a) direct outputting by the Output File Processor (ØFP) module, b) input to the SØRT2 processor (SDR3) module and then the XY-output modules (XYTRAN and XYPLØT) or c) input to the Stress/Strain Coordinate System Transformation (CURV) module for further processing.

4.46.3 DMAP Calling Sequence

SDR2 { CASECC
 CASEXX } ,CSTM,MPT,DIT, { HEQEXIN
 EQEXIN
 EQDYN } , { HSIL
 SIL
 SILD } ,GPTT,EDT,

BGPDT, { PGG
 PGV1
 TØL
 LAMA
 CLAMA } , { HQG
 QG
 QBG
 BQG
 QG1
 QPC
 QP } , { HUGV
 UGV
 UGV1
 UPV
 UPVC
 UBGV
 PHIG
 CPHIP } , { HEST
 EST
 ESTL } ,XYCDB, { HPGG
 PGG
 PPF } /

{ ØPPCA
ØPG1
ØPPC1
ØPP1
HØPG } , { IQP1
ØQG1
ØQBG1
ØBQG1
ØQPC1
ØQP1 } , { IPHIP1
ØUGV1
ØUPV1
ØUPVC1
ØUBGV1
ØPHIG
ØCPHIP
HØUGV1 } , { IES1
ØES1
ØESC1
ØESB1
ØBES1
HØES1
ØES1A } , { HØEF1
IEF1
ØEF1
ØEFC1
ØEFB1
ØBEF1 } , { PUGV
PUGV1
PPHIG
PUBGV1
PCHIP
PUPVC1
HPUGV1 } /

C,N,APP / V,N,NØSØRT2 / C,Y,IFLAG \$

MODULE FUNCTIONAL DESCRIPTIONS

4.46.4 Input Data Blocks

CASECC	- Case Control Data Table.
CASEXX	- Case Control Data Table for Dynamics problems.
CSTM	- Coordinate System Transformation Matrices.
MPT	- Material Property Table.
DIT	- Direct Input Tables.
HEQEXIN EQEXIN }	- Equivalence between external grid or scalar numbers and internal numbers.
EQDYN	- Equivalence between external points and scalar index values.
HSIL SIL }	- Scalar Index List.
SILD	- Scalar Index List for Dynamics.
GPTT	- Grid Point Temperature Table.
EDT	- Element Deformation Table.
BGPDT	- Basic Grid Point Definition Table.
HPGG PGG }	- Static load vector appended to include all boundary conditions.
PGV1	- Matrix of successive sums of incremental load vectors.
TØL	- Table of output times.
PPF	- Dynamic loads for frequency response.
LAMA	- Real Eigenvalue Table.
CLAMA	- Complex Eigenvalue Table.
HQG QG }	- Single-point constraint forces and determinant support forces matrix.
QBG	- Single-point forces of constraint matrix for Differential Stiffness - g set.
BQG	- Single-point forces of constraint matrix for a Buckling Analysis problem - g set.
QG1	- Matrix of successive sums of incremental vectors of single-point constraint forces.
QPC	- Complex single-point forces of constraint - p set.
QP	- Transient single-point forces of constraint - p set.
UGV	- Displacement vector matrix giving displacements in the g set.
UGV1	- Matrix of successive sums of incremental displacement vectors.
UPV	- Transient solution vectors - p set.
HUGV	- Temperature vector matrix giving temperatures in the g set.

FUNCTIONAL MODULE SDR2 (STRESS DATA RECOVERY - PHASE 2)

UPVC - Frequency response solution vectors - p set.
UBGV - Displacement vector matrix for differential stiffness giving displacements in the g set.
PHIG - Eigenvector matrix giving eigenvecotrs.
CPHIP - Complex eigenvectors in the p set.
HEST } - Element Summary Table.
EST }
ESTL - Element Summary Table for Linear Elements.
XYCDB - XY Case Control Data Block.

Notes:

1. If the first input data block is purged, it is a fatal error. This data block is called "Case Control" in this Module Functional Description.
2. The CSTM may be purged if no coordinate systems are referenced, or if stresses and/or forces are not requested.
3. The MPT may be purged if no stress or force requests are present.
4. The DIT may be purged if no stress or force requests are present, or if no temperature dependent materials are referenced.
5. The second record of EQEXIN or EQDYN must exist if a request exists for any of: loads, forces of single-point constraint, displacements, velocities, accelerations, or plots.
6. SIL or SILD may be purged if no stress or force requests exist, or there are no extra-points, no plots and no thermal loads. (The second record is used by SDR2).
7. The GPTT may be purged if no thermal loading exists, or there are no requests for stresses for forces.
8. The EDT may be purged if there are no element requests for forces or stresses, or if there are no enforced element deformations in the problem.
9. The BGPOT may be purged if the problem is in basic coordinates and no element requests for stresses or forces exist. No plots will result however.
10. LAMA or CLAMA may not be purged if an eigenvalue or frequency response problem.

MODULE FUNCTIONAL DESCRIPTIONS

11. If input data block 11 (QG or QBG, etc.) is purged, forces of single-point constraint requests are ignored.
12. If input data block 12 (UGV or UGV1, etc.) is purged, SDR2 will process only loads and forces of single-point constraint requests.
13. If the EST or ESTL is purged, element stresses and force requests are ignored.
14. The XYCDB may be purged.

4.46.5 Output Data Blocks

H0PG1	-	}	Output load vector requests.
0PPCA	-		
0PG1	-		
0PPC1	-		
0PP1	-		
IQP1	-	}	Output forces of single-point constraint requests.
0QG1	-		
0QBG1	-		
0BQG1	-		
0QPC1	-		
0QO1	-		
H0QG1	-		
0UGV1	-	}	Output displacement vector requests.
0UPVC1	-		
0UBGV1	-		
IPHIP1	-	}	Output eigenvector requests.
0PHIG	-		
0CPHIP	-		
IES1	-	}	Output element stress requests.
0ES1	-		
0ESC1	-		
0ESB1	-		
0BES1	-		
H0ES1	-		
0ES1A	-		Output element strain/curvature requests.
IEF1	-	}	Output element force requests.
0EF1	-		
0EFC1	-		
0EFB1	-		
0BEF1	-		
PUGV	-		Translation components of the displacement vector rotated to basic coordinates.
PUGV1	-		
PPHIG	-		
PUBGV1	-		
PCPHIP	-		
PUPVC1	-		
H0UGV1	-	}	Output temperature vector requests.
HPUGV1	-	}	Translation components of the temperature vector rotated to basic coordinates.

Notes: Output data blocks purged will result in output requests to those data blocks not being processed.

FUNCTIONAL MODULE SDR2 (STRESS DATA RECOVERY - PHASE 2)

4.46.6 Parameters

APP - Input-BCD-no default. Specifies the approach code to be placed in the output data blocks as per the following table.

<u>APP Parameter</u>	<u>Solution Type</u>	<u>Approach Code</u>
STATICS	Statics solution	1
REIGEN	Real eigenvalue solution	2
DSO	Statics phase of differential stiffness solution	3
DS1	Final phase of differential stiffness solution	4
FREQ	Frequency response solution	5
TRANSNT	Transient response solution	6
BKLO	Statics phase of buckling solution	7
BKL1	Final phase of buckling solution	8
CEIGEN	Complex eigenvalue solution	9
PLA	Piecewise linear analysis solution	10

NØSØRT2 - Output-integer-default=1. Set to 1 if there are SØRT2 requests or requirements and set to -1 otherwise.

IFLAG - Input-integer-default=-1. Processing flag. If IFLAG<0, element stresses are computed. If IFLAG≥0, element strains/curvatures are computed instead. (Strains/curvatures are computed only for TRIA1, TRIA2, QUAD1 and QUAD2 elements).

4.46.7 Method

The SDR2 functional module is constructed in a modular form consisting of five stages. A small executive control program (subroutine SDR2) containing the main entry point for the module serially calls the five stages for execution.

1. Stage I, performed by subroutine SDR2AA, prepares, if necessary, a modified Case Control data block, internal to SDR2, to insure that any XY-output requests present and not included in the \emptyset FP output requests of Case Control are included in the output of SDR2 for later processing by functional modules XYTRAN and XYPL \emptyset T.
2. Stage II, performed by subroutine SDR2A, analyzes the overall output requests within the subcases of Case Control and sets flags for use by stages III, IV, and V.
3. Stage III, performed by subroutine SDR2B, is executed only if stage II has determined that some element force or stress output requests are present within Case Control. If executed, element stress (or strain/curvature) matrices are computed once and stored along with certain other element properties appearing in the EST for each element appearing in a master set of element requests. (The master set is a set which is the union of all elements for which output requests are present in any subcase of Case Control.) These stored data are used in stage V repeatedly as necessary to satisfy the various output request combinations and multiple displacement vectors that may be present.
4. Stage IV, performed by subroutine SDR2C, is executed only if in stage II it has been determined that some output requests are present for any of: forces of single-point constraint, loads, displacement, velocities, accelerations, or deformed structure plots. If they are present, these requests (except for structure plots) are fully processed within this stage.
5. Stage V, performed by subroutine SDR2D, is executed only if stage III was executed. Stage V performs final element stress (or strain/curvature) and force computations. For each subcase of Case Control containing element output requests, the appropriate displacement vector is applied to the stress (or strain/curvature) matrices, computed in Stage III, of the elements requested for output to arrive at the final stress (or strain/curvature) and/or force outputs.

FUNCTIONAL MODULE SDR2 (STRESS DATA RECOVERY - PHASE 2)

For several of the five stages, the main subroutine listed utilizes additional subroutines to accomplish its particular task. The methods employed within each stage are further described in the subroutine descriptions in the next section.

The user may request in the Case Control Deck (by NCHECK = n) a stress and force precision check. For each element listed in a stress and force output request the number of significant digits in the result will be calculated. This check is based on the element stiffness matrix without corrections for element loads (thermal or misfit). For each stress or force component, i , the element routines will calculate:

$$r_i = \frac{|\sum_j S_{ij} u_j|}{\sum_j (|S_{ij}| \cdot |u_j|)} \quad (1)$$

and

$$h_i = -\log_{10}(r_i) \quad (2)$$

where u_j are the displacement components and S_{ij} are the stress matrix components. The number of significant digits, N_d , in the printed answer is therefore;

$$N_{di} = \log_{10}(2^p) - h_i \quad (3)$$

where p is approximately the number of bits in the mantissa and $\log_{10}(2^p)$ is 7.22 on IBM, 8.13 on UNIVAC, and 14.45 on CDC. If any one of the stress components has less precision than the value supplied by NCHECK = n, the results are printed by the element routine in a format unique to the type. A special case is r_i equal to zero for which full precision is assumed.

MODULE FUNCTIONAL DESCRIPTIONS

THIS PAGE HAS BEEN LEFT BLANK INTENTIONALLY.

FUNCTIONAL MODULE SDR2 (STRESS DATA RECOVERY - PHASE 2)

4.46.8 Subroutines

The utility routines PRETRS and PREMAT are called within the SDR2 module for initialization purposes so that the structural element subroutines can call the entry point TRANSS of PRETRS and MAT of PREMAT to fetch Coordinate System Transformation Matrices (CSTM) data and material properties (MPT and DIT) data respectively. GMMATS is used by element routines as a general matrix multiply routine and INVERS is used for inversion of small in-core (order usually ≤ 12) matrices. It should be noted that all matrices referenced in the structural element subroutines are stored by rows and are single precision. See the subroutine descriptions for these routines in section 3.

The axisymmetric shell element routines STRIR1, STRAP1, STØRD1, STRIR2, STRAP2, STØRD2 utilize the following functions and subroutines whose double precision versions are described in the Module Functional Description for SMA1 (section 4.27.8).

SDR2 (Single precision)

AI
BINT
F89
FF100
F6211
RØMBER
AMATRX

SMA1 (Double precision)

DKI
DKINT
DK89
DK100
DK211
RØMBDK
DMATRX

MODULE FUNCTIONAL DESCRIPTIONS

4.46.8.1 Subroutine Name: SDR2AA

1. Entry Point: SDR2AA
2. Purpose: To perform stage I as defined above, under "Method".
3. Calling Sequence: CALL SDR2AA
4. Method: SDR2AA attempts to open the XYCDB data block. If it is purged, a return is given to SDR2. Otherwise, the header record and first data record of XYCDB are skipped and data applying to all subcases are read from the second data record. If no such data exist a dummy master is created. Otherwise, the master data are reduced to a list of unique pairs. If only master data exist, flags are set appropriately.

For each record in the Case Control data block the following processing occurs.

- a. The record is read into core. If no XYCDB subcase corresponds to the Case Control subcase, pointers are set to the master data. Otherwise, the master data and appropriate XYCDB subcase data are merged and reduced to unique pairs.
- b. For each request for solution set output in XYCDB, the corresponding request in Case Control is examined. If no request is present in Case Control, the XYCDB request is reduced to a set in Case Control format, and a request for the set is turned on in Case Control. If the Case Control set is "ALL", no further action is taken. If the Case Control request is a set, the set is "merged" with the XYCDB set, and the request is altered to reflect the new set (unless all points in the XYCDB set were already in the Case Control set). A flag is set if any new requests are formed.
- c. When all requests for the current Case Control record have been analyzed, the record (as modified) is written on a scratch file.
- d. When all Case Control records have been read, the GINØ file name for the Case Control data block is switched to the scratch file (unless no modifications were made to Case Control).

4.46.8.2 Subroutine Name: SDR2A

1. Entry Point: SDR2A

2. Purpose: To perform stage II as defined above, under "Method".
3. Calling Sequence: CALL SDR2A
4. Method: SDR2A analyzes Case Control to determine the overall output requests. Flags are set to zero for all request possibilities, and, as each record of Case Control is analyzed, flags are set to 1 for those requests present. Simultaneously, a master set of all structural elements requested for output is created. The master set is then a union of all element requests present with duplicates removed.

4.46.8.3 Subroutine Name: SDR2B

1. Entry Point: SDR2B
2. Purpose: To perform stage III as defined above, under "Method".
3. Calling Sequence: CALL SDR2B
4. Method: SDR2B performs the "phase I" stress (or strain/curvature) and force recovery computations.

The CSTM is first read into core, if present, and then the material property data are read into core via the routine PREMAT. At this point, the EST is opened and processed with one pass.

SDR2A determined a master set list of all elements requested by the user to be output. This master set list, residing in core, is now used as the EST is processed.

For each record of the EST a particular element type is represented. Thus the first word of an EST record (giving the element type) is read, and the element dependent variables are set. The element summary for each element of this type is read, and, if the element is included in the master set, the phase I stress recovery routine is called to compute the element stress (or strain/curvature) matrices which are functions of element geometry and material properties only. The results of this computation are output to a scratch data block for use by SDR2D (stage V). When the end-of-record is encountered on the EST for this element type, the next element type record is processed, until all records of the EST have been passed.

MODULE FUNCTIONAL DESCRIPTIONS

4.46.8.4 Subroutine Name: SRØD1

1. Entry Point: SRØD1
2. Purpose: To generate element stress matrices for the RØD element.
3. Calling Sequence: CALL SRØD1

4.46.8.5 Subroutine Name: STUBE1

1. Entry Point: STUBE1
2. Purpose: To generate element stress matrices for the TUBE element.
3. Calling Sequence: CALL STUBE1

4.46.8.6 Subroutine Name: SPANL1

1. Entry Point: SPANL1
2. Purpose: To generate element stress matrices for the SHEAR and TWIST elements.
3. Calling Sequence: CALL SPANL1 (IARG)

IARG = $\begin{cases} 4 & \text{implies SHEAR panel element stress matrices will be generated.} \\ 5 & \text{implies TWIST panel element stress matrices will be generated.} \end{cases}$

4.46.8.7 Subroutine Name: STRBS1

1. Entry Point: STRBS1
2. Purpose: To generate element stress (or strain/curvature) matrices for the TRBSC element and perform sub-computations for the SQPD1 and STRPL1 routines.

3. Calling Sequence: CALL STRBS1 (IARG)

IARG $\begin{cases} 0 & = \text{TRBSC element} \\ 1 & = \text{Sub-computations for SQDPL1.} \\ 2 & = \text{Sub-computations for STRPL1.} \end{cases}$

4.46.8.8 Subroutine Name: STRPL1

1. Entry Point: STRPL1
2. Purpose: To generate element stress matrices for the TRPLT element.

3. Calling Sequence: CALL STRPL1

4.46.8.9 Subroutine Name: SQDPL1

1. Entry Point: SQDPL1

2. Purpose: To generate element stress matrices for the QDPLT element.

3. Calling Sequence: CALL SQDPL1

4.46.8.10 Subroutine Name: STRME1

1. Entry Point: STRME1

2. Purpose: To generate element stress (or strain/curvature) matrices for the TRMEM element and perform sub-computations for the SQDME1 routine.

3. Calling Sequence: CALL STRME1 (IARG)

IARG $\begin{cases} 0 & = \text{TRMEM} \\ 1 & = \text{Sub-computations for SQDME1 subroutine.} \end{cases}$

4.46.8.11 Subroutine Name: SQDME1

1. Entry Point: SQDME1

2. Purpose: To generate element stress matrices for the QDMEM element.

3. Calling Sequence: SQDME1

4.46.8.12 Subroutine Name: SELAS1

1. Entry Point: SELAS1

2. Purpose: To generate element stress matrices for the elements listed under the Calling Sequence.

3. Calling Sequence: CALL SELAS1 (IARG)

IARG $\begin{cases} 1 & = \text{ELAS1} \\ 2 & = \text{ELAS2} \\ 3 & = \text{ELAS3} \\ 4 & = \text{ELAS4} \end{cases}$

MODULE FUNCTIONAL DESCRIPTIONS

4.46.8.13 Subroutine Name: STRQD1

1. Entry Point: STRQD1
2. Purpose: To generate element stress matrices for the elements listed under the Calling Sequence.
3. Calling Sequence: CALL STRQD1 (IARG)

{	1	=	TRIA1
	2	=	TRIA2
	3	=	QUAD1
	4	=	QUAD2

4.46.8.14 Subroutine Name: SBAR1

1. Entry Point: SBAR1
2. Purpose: To generate element stress matrices for the BAR element.
3. Calling Sequence: CALL SBAR1

4.46.8.15 Subroutine Name: SCONE1

1. Entry Point: SCONE1
2. Purpose: To generate element stress matrices for the CONE element.
3. Calling Sequence: CALL SCONE1

4.46.8.16 Subroutine Name: STRIR1

1. Entry Point: STRIR1
2. Purpose: To generate element stress matrices for the TRIRG element.
3. Calling Sequence: CALL STRIR1

4.46.8.17 Subroutine Name: STRAP1

1. Entry Point: STRAP1
2. Purpose: To generate element stress matrices for the TRAPRG element.
3. Calling Sequence: CALL STRAP1

FUNCTIONAL MODULE SDR2 (STRESS DATA RECOVERY - PHASE 2)

4.46.8.18 Subroutine Name: STØRD1

1. Entry Point: STØRD1
2. Purpose: To generate stress matrices for the TØRDRG element.
3. Calling Sequence: CALL STØRD1

4.46.8.19 Subroutine Name: SQDM11

1. Entry Point: SQDM11
2. Purpose: To generate stress matrices for the QDMEM1 element.
3. Calling Sequence: CALL SQDM11

4.46.8.20 Subroutine Name: SDR2C

1. Entry Point: SDR2C
2. Purpose: To perform stage IV as defined above, under "method".
3. Calling Sequence: CALL SDR2C
4. Method: SDR2C operations are dependent on the Rigid Format being executed. In all cases the second record of EQEXIN or EQDYN is first read into core. An over-all loop of 3 passes is then executed to process the following: pass 1: displacements, velocities, accelerations; pass 2: single-point constraint forces; and pass 3: loads.

For each pass the Case Control data block is opened for input, and the following operations are performed depending on Rigid Format:

- a. For eigenvalue problems, a list of eigenvalues and mode numbers is read into core from LAMA or CLAMA.
- b. For Differential Stiffness or Buckling phase 1 problems, the first record of Case Control, which is used in phase 0 of Buckling or Differential Stiffness, is skipped.
- c. For frequency or transient response problems, a list of frequencies or times is read into core from TØL.

At this point a record in Case Control is read, and it is determined if a symmetry sequence of length LSYM is to be input. If it is, the previous LSYM vectors of the UGV data block are unpacked and a linear combination is formed in core. Otherwise, UGV is

MODULE FUNCTIONAL DESCRIPTIONS

opened, if not yet opened, and the next vector present is unpacked into core.

Data items are now assembled for the identification record, and this identification record is output to the output data block. Output line entries for the point-ID's requested are then written on the output data block forming a data record. At this time, if the user requested magnitude/phase for complex outputs, the magnitude/phase computations are performed on the real/imaginary pairs.

When all requests have been processed for this vector, the next Case Control record is read. If no more Case Control records exist and there are more vectors present, those vectors are processed using the last Case Control record's specifications.

When all vectors have been processed for the current loop pass, the next pass may be made for forces of single-point constraint or loads.

If deformed structure plots are requested, an output plot data block is formed during the first loop pass, described above, containing translation components of the displacement vector rotated to basic coordinates.

4.46.8.21 Subroutine Name: SDR2D

1. Entry Point: SDR2D
2. Purpose: To perform stage V as defined above, under "Method".
3. Calling Sequence: CALL SDR2D
4. Method: SDR2D performs the phase 2 stress (or strain/curvature) and force recovery computations. In this phase, actual stresses (or strains/curvatures) and forces are computed for the user-requested elements. These stresses (or strains/curvatures) and forces are a function of the stress (or strain/curvature) matrices computed in Stage III and the displacements at the grid points of the elements.

The operations of SDR2D are dependent upon the Rigid Format being executed. In all cases, the Case Control data block is opened first. For eigenvalue problems, a list of eigenvalues and mode numbers is read into core from LAMA and CLAMA. For Differential Stiffness or Buckling phase 1 problems, the first record of Case Control, which is used in phase 0 of Buckling or Differential Stiffness, is skipped. For frequency or transient response problems, a list of frequencies or times is read into core from PPF or PPT.

FUNCTIONAL MODULE SDR2 (STRESS DATA RECOVERY - PHASE 2)

Core and GINØ buffers are then allocated as required for a) the Case Control data block, b) the Element Deformation Table, c) the Grid Point Temperature Table, d) the element stress (or strain/curvature) matrices, and e) the EQEXIN data block. If there is insufficient space in core for the element stress (or strain/curvature) matrices, they are maintained on the scratch data block generated in stage III.

The displacement data block (UGV) is now opened, and the displacement vectors present are processed serially with Case Control as in Stage IV. Each element requested for output has its respective phase 2 element stress and force recovery routine called. The element routine outputs form the entries for the output data record of this element type. In the case of a complex displacement vector, the element routine is called first with a pointer to the real displacement vector and then with a pointer to the imaginary displacement vector. The results of these two calls are merged to form the complex output stresses and forces.

Stress and force precision checks are calculated, if requested, by SDRCHK. Only the following elements provide this capacity.

RØD,TUBE,CØNRØD,BAR

TRMEM,TRIA1,TRIA2,TRPLT,TRBSC

SHEAR,TWIST,QUAD1,QUAD2,QDPLT,QDMEM,QDMEM1,QDMEM2

HEXA1,HEXA2,TETRA,WEDGE,IHEX1,IHEX2,IHEX3

4.46.8.22 Subroutine Name: SDR2E

1. Entry Point: SDR2E
 2. Purpose: To pass through the element stress (or strain/curvature) matrices once, executing the final element stress (or strain/curvature) and force computations for the requests in the current subcase of Case Control.
 3. Calling Sequence: SDR2E (\$n)
- n = FØRTRAN statement number defining the return taken in the event of an error in SDR2E.

4.46.8.23 Subroutine Name: SRØD2

1. Entry Point: SRØD2
2. Purpose: To perform final stress and force computations for the RØD element.
3. Calling Sequence: CALL SRØD2

MODULE FUNCTIONAL DESCRIPTIONS

4.46.8.24 Subroutine Name: SPANL2

1. Entry Point: SPANL2
2. Purpose: To perform final stress and force computations for the SHEAR and TWIST elements.
3. Calling Sequence: CALL SPANL2 (IARG)
IARG 4 = SHEAR element.
 5 = TWIST element.

FUNCTIONAL MODULE SDR2 (STRESS DATA RECOVERY - PHASE 2)

THIS PAGE HAS BEEN LEFT BLANK INTENTIONALLY.

MODULE FUNCTIONAL DESCRIPTIONS

4.46.8.25 Subroutine Name: SELAS2

1. Entry Point: SELAS2
2. Purpose: To perform final stress and force computations for the ELAS1, ELAS2, ELAS3, and ELAS4 elements.
3. Calling Sequence: CALL SELAS2

4.46.8.26 Subroutine Name: SBSPL2

1. Entry Point: SBSPL2
2. Purpose: To perform final stress and force computations for the TRBSC, TRPLT, and QDPLT elements.
3. Calling Sequence: CALL SBSPL2 (IARG)

IARG {
0 = TRBSC element.
3 = TRPLT element.
4 = QDPLT element.

4.46.8.27 Subroutine Name: STQME2

1. Entry Point: STQME2
2. Purpose: To perform final stress computations for the TRMEM and QDMEM elements.
3. Calling Sequence: CALL STQME2 (IARG)

IARG {
1 = TRMEM element.
2 = QDMEM element.

4.46.8.28 Subroutine Name: STRQD2

1. Entry Point: STRQD2
2. Purpose: To perform final stress (or strain/curvature) and force computations for the TRIA1, TRIA2, QUAD1, and QUAD2 elements.

FUNCTIONAL MODULE SDR2 (STRESS DATA RECOVERY - PHASE 2)

3. Calling Sequence: CALL STRQD2 (IARG)

IARG { 3 = TRIA1 or TRIA2 element.
4 = QUAD1 or QUAD2 element.

4.46.8.29 Subroutine Name: SCONE2

1. Entry Point: SCONE2

2. Purpose: To perform final harmonic stress and force computations for the CONE element.

3. Calling Sequence: CALL SCONE2

4.46.8.30 Subroutine Name: SCONE3

1. Entry Point: SCONE3

2. Purpose: To compute the final stresses and forces for one of the 14 possible points in a CONE element.

3. Calling Sequence: CALL SCONE3 (LARG)

LARG = Logical argument set .FALSE. initially and then set .TRUE. by SCONE3 after the last point defined for a particular element has had its final stresses and forces computed.

4.46.8.31 Subroutine Name: SBAR2

1. Entry Point SBAR2

2. Purpose: To perform final stress and force computations for the BAR element.

3. Calling Sequence: CALL SBAR2

4.46.8.32 Subroutine Name: STRIR2

1. Entry Point: STRIR2

2. Purpose: To perform final stress and force computations for the TRIRG element.

3. Calling Sequence: CALL STRIR2 (TGRID)

TGRID = 3 word real array giving grid point temperatures at the 3 connection grid points.

MODULE FUNCTIONAL DESCRIPTIONS

4.46.8.33 Subroutine Name: STRAP2

1. Entry Point: STRAP2
2. Purpose: To perform final stress and force computations for the TRAPRG element.
3. Calling Sequence: CALL STRAP2 (TGRID)

TGRID = 4 word real array giving grid point temperatures at the 4 connection grid points.

4.46.8.34 Subroutine Name: STØRD2

1. Entry Point: STØRD2
2. Purpose: To perform final stress and force computations for the TØRDRG element.
3. Calling Sequence: CALL STØRD2 (TGRID)

TGRID = Two-word real array giving grid point temperatures at the two connection grid points.

4.46.8.35 Subroutine Name: MAGPHA

1. Entry Point: MAGPHA
 2. Purpose: To compute the magnitude and phase of a complex number, c.
 3. Calling Sequence: CALL MAGPHA (A,B)
- A - Real part of c on input, magnitude of c on return.
B - Imaginary part of c on input, phase of c on return.

4.46.8.36 Subroutine Name: SAXIF1

1. Entry Point: SAXIF1
2. Purpose: To generate element pressure-velocity matrices for the AXIF elements.
3. Calling Sequence: CALL SAXIF1 (IARG)

$$IARG = \begin{cases} 0 & = \text{AXIF2} \\ 1 & = \text{AXIF3} \\ 2 & = \text{AXIF4} \end{cases}$$

4.46.8.37 Subroutine Name: SAXIF2

1. Entry Point: SAXIF2
2. Purpose: To generate element velocities or accelerations for the AXIF elements.

FUNCTIONAL MODULE SDR2 (STRESS DATA RECOVERY - PHASE 2)

3. Calling Sequence: CALL SAXIF2 (IØPT, IPART, BRANCH, EIGEN)

IØPT = $\begin{cases} 0 = \text{AXIF2} \\ 1 = \text{AXIF3} \\ 2 = \text{AXIF4} \end{cases}$

IPART = $\begin{cases} 1 = \text{Real Vector} \\ 2 = \text{Imaginary Vector} \end{cases}$

BRANCH = SDR2 Process code word

EIGEN = 3 words for complex or real eigenvalue or real frequency.

4.46.8.38 Subroutine Name: SSLØT1

1. Entry Point: SSLØT1

2. Purpose: To generate element pressure-velocity matrices for the SLØT elements.

3. Calling Sequence: CALL SSLØT1 (IØPT)

IØPT = $\begin{cases} 0 = \text{SLØT3} \\ 1 = \text{SLØT4} \end{cases}$

4.46.8.39 Subroutine Name: SSLØT2

1. Entry Point: SSLØT2

2. Purpose: To compute element velocities or accelerations for the SLØT elements.

3. Calling Sequence: CALL SSLØT2 (IØPT, IPART, BRANCH, EIGEN)

IØPT = $\begin{cases} 0 = \text{SLØT3} \\ 1 = \text{SLØT4} \end{cases}$

IPART = $\begin{cases} 1 = \text{Real Vector} \\ 2 = \text{Imaginary Vector} \end{cases}$

BRANCH = SDR2 Process code word

EIGEN = 3 words for eigenvalue or frequency.

4.46.8.40 Subroutine Name: SSØLD1

1. Entry Point: SSØLD1

2. Purpose: To generate stress matrices for the solid elements.

MODULE FUNCTIONAL DESCRIPTIONS

3. Calling Sequence: CALL SSØLD1 (I)

<u>I</u>	<u>Element Type</u>
1	TETRA
2	WEDGE
3	HEXA1
4	HEXA2

4.46.8.41 Subroutine Name: SSØLD2

1. Entry Point: SSØLD2
2. Purpose: To perform final stress and force computations for the solid elements.
3. Calling Sequence: CALL SSØLD2 (I,T)

<u>I</u>	<u>Element Type</u>
1	TETRA
2	WEDGE
3	HEXA1
4	HEXA2

T - Temperature Vector

4.46.8.42 Subroutine Name: SDRETD

1. Entry Point: SDRETD
2. Purpose: Reads element temperature from a pre-positioned record.
3. Calling Sequence: CALL SDRETD (ID,T,G)

ID - Element identification number for which the data is desired.

T - Area into which data will be stored

G - = 0, element temperature format data is desired.

≠ 0, number of grid points.

4.46.8.43 Subroutine Name: SDHTF1

1. Entry Point: SDHTF1
2. Purpose: To extract heat flux data for the following elements - RØD, CØNRØD, TUBE, BAR, TRMEM, QDMEM, TRIA1, TRIA2, QUAD1, QUAD2, TRIARG, TRAPRG, TETRA, WEDGE, HEXA1, HEXA2, and HBDY.

3. Calling Sequence: CALL SDHTF1(TYPE,REJECT), where:

TYPE - Input, integer, element type number.

REJECT - Output, logical, true if element is not part of above set.

4.46.8.44 Subroutine Name: SDHTFF

1. Entry Point: SDHTFF
2. Purpose: To calculate the Phase 1 matrices which describe heat flow in terms of the temperatures of the connected grid points.
3. Calling Sequence: CALL SDHTFF
/SDR2X5/EST(100),DATA(45) - Input element data and output matrices.
/SDR2X6/ - Contains element parameters rearranged by SDHTF1 in a uniform format.

4.46.8.45 Subroutine Name: SDHTF2

1. Entry Point: SDHTF2
2. Purpose: To calculate the output element "forces" for the heat transfer problem.
3. Calling Sequence: CALL SDHTF2

4.46.8.46 Subroutine Name: SQDM12

1. Entry Point: SQDM12
2. Purpose: To perform final stress computations for the QDMEM1 element.
3. Calling Sequence: CALL SQDM12

4.46.8.47 Subroutine Name: SIHEX1

1. Entry Point: SIHEX1
2. Purpose: To generate element stress matrices for the IHEX1, IHEX2, and IHEX3 elements, SIHEX1 must be called once for each stress point in an element.
3. Calling Sequence: CALL SIHEX1 (TYPE,STRSPT)

TYPE = $\begin{Bmatrix} 1 - \text{IHEX1} \\ 2 - \text{IHEX2} \\ 3 - \text{IHEX3} \end{Bmatrix}$ - integer-input

STRSPT = stress point number - integer-input

4.46.8.48 Subroutine Name: IHEXSS

1. Entry Point: IHEXSS
2. Purpose: To generate isoparametric shape function data. (Single precision version of subroutine IHEXSD, Section 4.27.8.42).
3. Calling Sequence: CALL IHEXSS (ITYPE,SHP,DSHP,JINV,DETJ,EID,XI,ETA,ZETA,CORD)
See Section 4.27.8.42 for description of arguments.

4.46.8.49 Subroutine Name: SIHEX2

1. Entry Point: SIHEX2
2. Purpose: To perform final stress computations for the IHEX1, IHEX2, and IHEX elements. SIHEX2 must be called once for each stress point in an element.
3. Calling Sequence: CALL SIHEX2 (TYPE,GPT)

TYPE = $\begin{Bmatrix} 1 - \text{IHEX1} \\ 2 - \text{IHEX2} \\ 3 - \text{IHEX3} \end{Bmatrix}$ - integer-input

GPT - Array of grid point temperatures - real-input

4.46.8.50 Subroutine Name: STPAX1

1. Entry Point: STPAX1
2. Purpose: To generate element stress matrices for TRAPAX element.
3. Calling Sequence: CALL STPAX1

4.46.8.51 Subroutine Name: STRAX1

1. Entry Point: STRAX1
2. Purpose: To generate element stress matrices for the TRIAAX element
3. Calling Sequence: CALL STRAX1

4.46.8.52 Subroutine Name: STPAX2

1. Entry Point: STPAX2
2. Purpose: To perform final stress and force computations for the TRAPAX element

3. Calling Sequence: CALL STPAX2 (\$,TI)

\$ 1 = sin
 2 = cos

TI = four word real array giving grid point temperatures at the four connection grid points

4.46.8.53 Subroutine Name: STPAX3

1. Entry Point: STPAX3
2. Purpose: To compute the final stresses and forces for one of the fourteen possible points in a TRAPAX element
3. Calling Sequence: CALL STPAX (LARG)

LARG = Logical argument set .FALSE. initially and then set .TRUE. by STPAX3 after the last point defined for a particular element has had its final stresses and forces computed

4.46.8.54 Subroutine Name: STRAX2

1. Entry Point: STRAX2
2. Purpose: To perform final stress and force computations for the TRIAX element
3. Calling Sequence: CALL STRAX2 (\$,TI)

\$ 1 = sin
 2 = cos

TI = Three word real array giving grid point temperatures at the three connection grid points

4.46.8.55 Subroutine Name: STRAX3

1. Entry Point: STRAX3
2. Purpose: To compute the final stresses and forces for one of the fourteen possible points in a TRIAX element
3. Calling Sequence: CALL STRAX3 (LARG)

LARG = Logical argument set .FALSE. initially and then set .TRUE. by STRAX3 after the last point defined for a particular element has had its final stresses and forces computed

MODULE FUNCTIONAL DESCRIPTIONS

4.46.8.56 Subroutine Name: SDRCHK

1. Entry Point: SDRCHK
2. Purpose: To calculate the stress/force precision check data, $ND_i = \log_{10}(2^P) + \log_{10}(r_i)$
3. Calling Sequence: CALL SDRCHK (F,C,LVEC,K,PREC,H)
 - F - Array of stresses and forces
 - C - Corresponding array of stresses and forces calculated using absolute values. On output, the number of significant digits.
 - LVEC - Length of arrays F and C.
 - K - Current count of terms exceeding limit PREC.
 - PREC - Limit on minimum precision before incrementing K.
 - H - Value of $\log_{10}(2^P)$.

4.46.8.57 Subroutine Name: SD2RHD

1. Entry Point: SD2RHD
2. Purpose: To provide heading for stress/force precision check output.
3. Calling Sequence: CALL SD2RHD (IS,SETUP)
 - IS - Seven-word array for heading. On input, word 1 = subcase, word 2 = load or mode number, word 6 and 7 = the frequency, eigenvalue, or time. On output (SETUP \neq 0), words 3 through 5 are BCD identifying values.
 - SETUP - If nonzero, supply words 3 through 5 of IS and print the heading, otherwise just print the heading.

4.46.8.58 Subroutine Name: SMMATS

1. Entry Point: SMMATS
2. Purpose: General multiply and add using real single-precision matrices. Also returns values based on using the absolute values of the input matrices terms.
3. Calling Sequence: CALL SMMATS (A,IR0WA,IC0LA,MTA,B,IR0WB,IC0LB,MTB,C,AC)
 - AC - Resultant matrix when absolute values of A, B, and C are used.All other arguments are the same as subroutine GMMATS and GMMATD.

4.46.9 Design Requirements

1. Since the five stages of the SDR2 module must be able to operate under all Rigid Formats, a branch design has been used within each stage whereby operations that are variant under different Rigid Formats are grouped into substructures within that stage. At these locations, a branch is always made to the substructure appropriate to the Rigid Format being executed.
2. The following common blocks appear only within the subroutines of the SDR2 module.

a. COMMON/SDR2X1/

This common block contains seventeen flag words which are set on the basis of requests in the Case Control data block by SDR2A.

b. COMMON/SDR2X2/

This common block contains twenty-nine points defining the locations of the various requests and set definitions within the Case Control data block.

c. COMMON/SDR2X3/

This common block contains data constants unique to the structural elements.

d. COMMON/SDR2X4/

This common block contains local variables and flags set by the subroutines of the SDR2 module for communications between these subroutines.

e. COMMON/SDR2X5/

This common block is used by SDR2B to send EST data to the phase 1 element routines and to receive outputs from the phase 1 element routines.

f. COMMON/SDR2X6/

This common block consists of a three hundred word scratch area for use by the element routines while performing phase 1 computations.

g. COMMON/SDR2X7/

This common block is used by SDR2D for sending and receiving data to the element routines performing phase 2 computations.

h. COMMON/SDR2X8/

This common block consists of a three hundred word scratch area in core for use by the element routines while performing phase 2 computations.

i. COMMON/SDR2X9/

This common block consists of 7 words for sending stress and force precision check data to the element routines.

4.46.10 Diagnostic Messages

SDR2 being one of the last modules to execute in a problem solution, makes every attempt at execution in any event. Should an error be detected by SDR2, a message is queued for output and,

MODULE FUNCTIONAL DESCRIPTIONS

if possible, SDR2 continues to execute portions of the solution not affected by the error. The following NASTRAN messages may be output by SDR2: 2075, 2076, 2077, 2078, 2079, 2080, 2143, 3001, 3002, 3003, 3008, and 4024.

FUNCTIONAL MODULE DPD (DYNAMICS POOL DISTRIBUTOR)

4.47 FUNCTIONAL MODULE DPD (DYNAMICS POOL DISTRIBUTOR)

4.47.1 Entry Point: DPD

4.47.2 Purpose

DPD is the principal data processing module for dynamics problems. New tables are assembled to account for any extra points in the model and the additional displacement sets used in dynamics. Bulk data cards which control the solution of a dynamics problem are processed and assembled into various data blocks for convenience and efficiency in solution of the dynamics problem.

4.47.3 DMAP Calling Sequence

```
DPD      DYNAMICS,GPL,{HSIL},{HUSER}/GPLD,{HSILD},{HUSERD},{TFP00L},{HDLT},{PSDL,FRL},{HNLFT},{HTRL},
        EED,{HEQDYN}/V,N,LUSET/V,N,LUSETD/V,N,N0TFL/V,N,N0DLT/V,N,N0PSDL/V,N,N0FRL/V,N,N0NLFT/
        V,N,N0TRL/V,N,N0EED/C,N,0/V,N,N0UE  $
```

4.47.4 Input Data Blocks

DYNAMICS - Collection of bulk data cards for dynamics problem.

GPL - Grid Point List.

HSIL }
SIL } - Scalar Index List.

USER - Displacement set definitions table.

HUSER - Temperature set definitions table.

Note: DYNAMICS may be purged. Other input data blocks may not be purged.

4.47.5 Output Data Blocks

GPLD - Grid Point List Dynamics.

HSILD }
SILD } - Scalar Index List Dynamics.

USERD - Displacement set definition table dynamics.

HUSERD - Temperature set definition table dynamics.

TFP00L - Transfer Function Pool.

MODULE FUNCTIONAL DESCRIPTIONS

HDLT }
DLT } - Dynamic Loads Table.

PSDL - Power Spectral Density List.

FRL - Frequency Response List.

HNLFT }
NLFT } - Non-Linear Forcing Table.

HTRL }
TRL } - Transient Response List.

EED - Eigenvalue Extraction Data.

HEQDYN }
EQDYN } - Equivalence between external and internal numbers - dynamics.

Note: GPLD, SILD, USETD and EQDYN may not be purged. All other data blocks may be purged.

4.47.6 Parameters

LUSET - Input-integer-no default. Degrees of freedom in the g - displacement set.

LUSETD - Output-integer-no default. Degrees of freedom in the p - displacement set.

NØTFL - Output-integer-no default. Number of transfer function sets in the bulk data, -1 if no sets are defined.

NØDLT - Output-integer-no default. +1 if dynamics load data are present in the bulk data (i.e., DLT is created), -1 otherwise.

NØPSDL - Output-integer-no default. +1 if the PSDL is created, -1 otherwise.

NØFRL - Output-integer-no default. +1 if the FRL is created, -1 otherwise.

NØNLFT - Output-integer-no default. +1 if the NLFT is created, -1 otherwise.

NØTRL - Output-integer-no default. +1 if the TRL is created, -1 otherwise.

NØEED - Output-integer-no default. +1 if the EED is created, -1 otherwise.

NØUE - Output-integer-no default. Number of extra points in the model, -1 if there are no extra points.

4.47.7 Method

4.47.7.1 General

Subroutine DPD is the main control program for the module. It initializes each of the DMAP parameters, allocates buffers in open core (/DPDCØR/), and calls each of the principal routines of the module as follows.

FUNCTIONAL MODULE DPD (DYNAMICS POOL DISTRIBUTOR)

1. DPD1 to assemble GPLD, USETD, SILD and EQDYN.
2. DPD2 to assemble DLT.
3. DPD3 to assemble FRL and PSDL.
4. DPD4 to assemble NLFT and TRL.
5. DPD5 to assemble EED and TFL.

4.47.7.2 Assembly of GPLD, USETD, SILD and EQDYN

The second logical record of GPL, which contains pairs of external point identification and sequence numbers, is read into core. Three words are used for each entry in the GPL. In the third word of each entry the internal index is stored. The list of extra points is read from the EPØINT record in DYNAMICS. For each extra point, a three-word entry is added to the list now in core. The first word contains the extra point identification, the second contains the initial sequence number equal to 1000 times the point ID, and the third word is zero. The SEQEP record in DYNAMICS is read. For each referenced point, the sequence number is replaced by the new sequence number. The list in core is now sorted on sequence number by subroutine SØRT. The sequence number in each entry is replaced by an internal index according to the position of the entry following the sort. The GPLD is now written. It consists of one logical record of one word entries, each entry containing the external point identification. The internal index is implied by the position of the entry in the record.

The SIL is read into core following the table of three-word entries currently in core. Bit masks are initialized for the various displacement sets in statics and dynamics. Files containing the USETD and SILD data blocks are opened to write. The file containing the USET data block is opened to read. Each of the three-word entries in core is processed as follows.

1. If the entry corresponds to a grid point, six words are read from USET, bits for displacement sets in dynamics are turned on according to the statics sets to which the point belongs, six words are written on USETD, one word is written on SILD, the second word of the three-word entry is replaced with the scalar index value in the p-set, the third word is replaced with the scalar index value in the g-set, and the scalar index counter for the p-set is incremented by six.

MODULE FUNCTIONAL DESCRIPTIONS

2. If the entry corresponds to a scalar point, one word is read from USET, and the process proceeds as above except that one word is written on USETD and the scalar index counter is incremented by one.

3. If the entry corresponds to an extra point, one word is written on USETD, the extra point counter is incremented, and the process proceeds as with a scalar point.

When all entries have been processed, USETD is complete and the first logical record of SILD is complete. The second logical record of SILD is now written. It comprises two-word entries, each pair containing a scalar index value in the g-set and the corresponding scalar index value in the p-set.

The third word of each of the three word entries in core is now replaced with a code word which is ten times the scalar index value in the p-set plus the type of point (1 = grid, 2 = scalar, 3 = extra). The table is sorted on external point identification. EQDYN is written as two logical records. The first record contains pairs, each consisting of an external point ID and a scalar index value in the p-set. The second record contains pairs, each consisting of an external point ID and a code word.

4.47.7.3 Assembly of the DLT

The DAREA, DELAY and DPHASE tables are read from DYNAMICS, one table at a time. Grid point and component codes are converted to a scalar index value in the p-set by subroutine DPDA. When all entries of a table have been read, the table is sorted on scalar index value and written as a logical record on a scratch file (three scratch files are used, one for each of the three types of tables). The table identification is saved in core.

The RL0AD1, RL0AD2, TL0AD1 and TL0AD2 cards are read from DYNAMICS and stored in core. Eleven words are used for each entry. In the first word of each entry, a code for the card type is stored. When all cards have been read and stored in core, the data are sorted on load set identification.

DL0AD cards are read from DYNAMICS and stored in core, and the data within each DL0AD card are sorted on referenced set identification. The file containing the DLT data block is opened to write. The header record is written. It contains the data block name, a list of set

identifications defined on DLØAD cards and a list of set identifications defined on RLØAD1, RLØAD2, TLØAD1 and TLØAD2 cards. The DLØAD data are then written as the first logical record of the DLT. The remainder of the DLT comprises one logical record per load set. For each entry in the 11-word per entry table in core the following processing occurs.

1. The scratch file containing the tables referenced in the entry is positioned to read the referenced table.
2. Entries are read from each of the referenced tables. A four-word entry is written on the DLT consisting of the scalar index value, A, τ , and Θ .
3. When all entries from the tables have been read, the DLT record is closed, and the process repeats for the next load set.

4.47.7.4 Assembly of the FRL and the PSDL

FREQ1, FREQ2 and FREQ cards are read from DYNAMICS and stored in core. Frequencies on FREQ cards are converted to radians. When all the data have been read, a list containing three words per entry is accumulated in core. The first word contains the frequency set identification, the second word contains a pointer to the first word where data belonging to the set are stored, and the third word defines the type (0 = FREQ1, -1 = FREQ2, N = FREQ, where N is the number of words in the frequency set). The list is sorted on set identification. The file containing the FRL data block is opened to write. The header record is written. It contains the data block name and a list of all frequency set identifications. The remainder of the FRL is comprised of one logical record per frequency set.

For each entry in the three-word per entry list in core the following processing occurs.

1. If the entry corresponds to a FREQ1 set, then $N+1$ frequencies are written as a logical record on the FRL by the following equation:

$$f_i = f_0 + (i-1) \Delta f, \quad i = 1, 2, \dots, N + 1. \quad (1)$$

2. If the entry corresponds to a FREQ2 set, then $N+1$ frequencies are written as one logical record on the FRL by the following equation:

MODULE FUNCTIONAL DESCRIPTIONS

$$f_i = f_0 10^{(i-1)\delta} \quad i = 1, 2, \dots, N + 1 \quad (2)$$

where

$$\delta = \frac{1}{N} \log_{10} \left(\frac{f_e}{f_0} \right). \quad (3)$$

3. If the entry corresponds to a FREQ set, the frequencies in the set are sorted, and any duplicate frequencies are discarded. The sorted list is written as one logical record on the FRL.

The RANDPS cards are read into core (if no RANDPS data are present, the PSDL is not assembled). The RANDT1 and RANDT2 cards are read into core, and a list similar to that in the frequency processing is formed. This list is sorted on set identification number. The file containing the PSDL is opened to write, and the set identifications are written in the header record. The RANDPS data are written as the first logical record of the PSDL. The remainder of the PSDL contains one logical record per set. For RANDT2 sets, the data are sorted on time lag, and duplicates are discarded prior to writing the record. For RANDT1 sets, $N + 1$ time lags, τ_i , are written where

$$\tau_i = \tau_0 + (i-1)\Delta\tau \quad i = 1, 2, \dots, N + 1 \quad (4)$$

4.47.7.5 Assembly of the NLFT and TRL

The NØLINi ($i = 1, 2, 3, 4$) cards are read into core. Each referenced grid point and component code is converted to a scalar index value in the u_p -set. The data are sorted on set identification number. USETD is read into core. The file containing the NLFT data block is opened to write, and the set identifications are written in the header record. The remainder of the NLFT contains one logical record per set. Scalar index values within each set are converted to scalar index values in the u_d and u_e sets. The data within each set are sorted on the scalar index value to which the forcing function is applied.

The TIC cards are read, referenced grid points and component codes are converted to scalar index values in the u_p -set, and the data are written on SCR1, one logical record per set. A list of the TIC set identifications is accumulated in core. USETD is read into core. The

file containing the TRL data block is opened to write. The set identifications are written in the header record. The last word of the header contains the degrees of freedom in the u_d -set. Data are read from SCR1. Scalar index values are converted to scalar index values in the u_d -set. Each TIC set is written as one logical record on the TRL. When all the TIC data have been processed, the TSTEP data are copied from DYNAMICS to the TRL, one logical record per TSTEP set.

4.47.7.6 Assembly of the EED and TFL

Processing of EIGB, EIGC, EIGP and EIGR cards is minimal. For each card type present, a corresponding logical record is written on EED. For each of the cards which specify PØINT, the referenced grid point and component code is converted to a scalar index value (u_a set for EIGB and EIGR cards, u_d set for EIGC cards).

Transfer function data are read from the TF record on DYNAMICS one set at a time. For each transfer function set, the point and component codes are converted to scalar index values in the u_p set, which in turn form row and column numbers of the transfer function matrices. The data are written on the TFL, one transfer function set per logical record. The set identification number is the first word of each logical record. Four word entries follow. The first word is 65536*column number plus row number; the next three words are the terms of the matrices.

4.47.8 Subroutines

Auxiliary subroutines DPD1, DPD2, DPD3, DP4 are described above.

4.47.8.1 Subroutine Name: DPDAA

1. Entry Point: DPDAA
2. Purpose: To convert a grid point and component code to a scalar index value in the u_p set.
3. Calling Sequence: CALL DPDAA
4. Method: A flag called INEQ is maintained in /DPDCØM/. If the flag is zero, EQDYN is read into core and INEQ is set to one. The grid point and component to be converted is stored in BUF(L) and BUF(L+1) where BUF and L are in /DPDCØM/. A binary search is performed in EQDYN. If the point is found, the corresponding scalar index value is stored in BUF(L). Otherwise, an error message is queued, and an internal

MODULE FUNCTIONAL DESCRIPTIONS

NØGØ flag is turned on.

4.47.9 Design Requirements

4.47.9.1 Allocation of Core Storage

In general, core storage requirements in DPD are the EQDYN table plus one set of data being processed plus two or three GINØ buffers. In DPD3 where EQDYN is not required, it is assumed that all data required to assemble the FRL or PSDL can be held in core at one time.

4.47.9.2 Environment

The Block Data program DPDCBD initializes /DPDCØM/ with GINØ file names, LØCATE codes for the various card types processed by DPD, and miscellaneous data. It must be resident in core when DPD is executed.

DPD is designed to operate in a single overlay segment. Communication in the module occurs through /DPDCØM/ and open core /DPDCØR/. If an alternate overlay is desired, DPDCDB, DPD and DPDAA could form a local primary segment and each of DPD1, DPD2, DPD3, DPD4 and DPD5 could form separate secondary segments. In this case, /DPDCØR/ must be inserted after the longest of the secondary segments. Four scratch files are used by DPD.

4.47.10 Diagnostic Messages

The following messages may be issued by DPD:

2064, 2066, 2068, 2069, 2071, 2107, 2135, 2136.

FUNCTIONAL MODULE READ (REAL EIGENVALUE ANALYSIS - DISPLACEMENT)

4.48 FUNCTIONAL MODULE READ (REAL EIGENVALUE ANALYSIS - DISPLACEMENT)

4.48.1 Entry Point: REIG

4.48.2 Purpose:

To solve the equation

$$[K] - \lambda [M] \{u\} = 0$$

for eigenvalues λ and their associated eigenvectors.

4.48.3 DMAP Calling Sequence

READ $\left\{ \begin{matrix} KKK \\ KAA \end{matrix} \right\}, \left\{ \begin{matrix} KDAAM \\ MAA \\ MKK \end{matrix} \right\}, MR, DM, EED, USET, CASECC, \left\{ \begin{matrix} LAMK \\ LAMA \end{matrix} \right\}, \left\{ \begin{matrix} PHIK \\ PHIA \end{matrix} \right\}, MI, \emptyset EIGS/V, N, F\emptyset RMAT/V, N, NEIGVS/$
V, N, NSKIP \$

4.48.4 Input Data Blocks

- $\left\{ \begin{matrix} KAA \\ KKK \end{matrix} \right\}$ - Partition of stiffness matrix - a set.
- KDAAM - Negative of partition of differential stiffness matrix $[K_{aa}^d]$ - a set.
- $\left\{ \begin{matrix} MAA \\ MKK \end{matrix} \right\}$ - Partition of mass matrix - a set.
- MR - Rigid body mass matrix - r set.
- DM - Rigid body transformation matrix.
- EED - Eigenvalue Extraction Data.
- USET - Displacement set definitions table.
- CASECC - Case Control Data Table.

Notes:

1. KAA must be present.
2. MR may or may not be present.
3. DM and USET must be present if MR is present.
4. EED must be present.
5. CASECC must be absent when substructure modal reduce is being performed. In all other cases, it must be present.
6. In Buckling Analysis $MAA = -KDAAM$.

4.48.5 Output Data Blocks

LAMA } - Real Eigenvalue Table.
 LAMK }
 PHIA } - Eigenvectors matrix giving the eigenvectors in the a set.
 PHIK }
 MI - Modal Mass Matrix.
 ØEIGS - Real Eigenvalue Summary Table.

Notes: LAMA and PHIA may also be input data blocks if the append mode is being used.

4.48.6 Parameters

FØRMAT - Input-BCD-no default. If FØRMAT ≠ MØDES, READ will solve a buckling problem (i.e., $[\lambda M - K] \{u\} = 0$) using EIGB data cards where M is the negative of the differential stiffness matrix.
 NEIGVS - Output-integer-no default. NEIGVS is the number of eigenvalues found. If none were found, NEIGVS = -1.
 NSKIP - Input-integer-default value of one. The method used by READ is taken from the NSKIP record of CASECC. This parameter is ignored when CASECC is purged, in which case the first available data on file EED is used.

4.48.7 Method

REIG is the main controlling program for the READ module. Its responsibility is to decide which method was asked for (Inverse Power, Determinant or Givens) and to pass control to the appropriate routine. Once eigenvalues have been extracted, REIG directs the sorting and normalizing of the vectors for final output. The flow of the module can be seen in the flow charts shown in Figure 1. If the output files LAMA and PHIA are present and non-null, READ enters the append mode (see READ7).

4.48.8 Subroutines

The subroutines used by READ are divided into five classes: 1) subroutines used by REIG, 2) subroutines used for the Inverse Power Method, 3) subroutines used for the Determinant Method, 4) subroutines used for the Givens Method, 5) subroutines used for the FEER Method, and 6) general subroutines. The descriptions for the general subroutines can be found in Section 3.

FUNCTIONAL MODULE READ (REAL EIGENVALUE ANALYSIS - DISPLACEMENT)

<u>REIG</u>	<u>INVERSE POWER</u>	<u>DETERMINANT</u>	<u>GIVENS</u>
READ1	INVPWR	DETM SUMM	VALVEC
READ2	INVP1	DETM1 SQRTM	SMLEIG
READ3	INVP2	DETM2 DETFBS	TRIDI
READ4	INVP3	DETM3	SICØX
READ5	NØRM1	DETM4	SINCAS
READ6	MTIMSU	DETM5	RØTAX
ØRTCK	XTRNSY	DETM6	RØTATE
INVERT	SUB	FDVECT	EMPCØR
INVTR	INVFS	EADD	FILCØR
READ7		DETDET	QRITER
		ARRM	WILVEC

GENERAL

DECØMP
 ADD
 PRELØC
 SSG2B
 SDR1B
 SDCØMP
 FACTØR
 TRANP1
 TRNSP
 MERGE
 MPYAD

4.48.8.1 Subroutine Name: READ1

1. Entry Point: READ1

2. Purpose: To compute the eigenvectors for the rigid body modes.

3. Calling Sequence: CALL READ1 (DM,MR,SCR1,SCR2,SCR3,PHIAT,USET,NR,LAMAT,SCR4)

DM,MR,USET are the GINØ file numbers of their respective data blocks - integer - input.

SCR1,...,SCR4 are the GINØ file numbers of 4 scratch files - integer - input.

PHIAT - GINØ file number of a temporary storage file for the eigenvectors - integer - input.

LAMAT - GINØ file number of a temporary storage file for the eigenvalues - integer - input.

NR - Number of rigid body modes - integer - output.

COMMON /READ1A/Z(1)

Z(1) - Array of open core for READ1

4. Method: Let r be the number of rigid body modes and let

$$I_1 = \begin{Bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{Bmatrix}, \quad (2)$$

$$I_2 = \begin{Bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{Bmatrix}, \quad (3)$$

etc.

Set $\{v_i\} = \{I_i\}$ and perform the following three steps for $i = 1, 2, \dots, r$.

a. Normalize $\{v_i\}$ by the following equations:

$$S_i = \{v_i\}^T [m_r] \{v_i\}, \quad (4)$$

$$\{\phi_i\} = \frac{1}{\sqrt{S_i}} \{v_i\}. \quad (5)$$

S_i must be greater than zero for a consistent rigid body system.

FUNCTIONAL MODULE READ (REAL EIGENVALUE ANALYSIS - DISPLACEMENT)

b. Calculate using $j = 1, 2, \dots, i$:

$$\alpha_j = \{\phi_j\}^T [m_r] \{I_{i+1}\}. \quad (6)$$

c. The next vector is then:

$$\{v_{i+1}\} = \{I_{i+1}\} - \sum_{j=1}^i \alpha_j \{\phi_j\}. \quad (7)$$

(Return to step b).

This procedure is a modification of the Schmidt orthogonalization procedure using the $\{I\}$ vectors as a starting point. Since the $[m_r]$ matrix is non-singular, the $\{I\}$ vectors are independent with respect to the matrix. Each new vector $\{I_{i+1}\}$ is made orthogonal with respect to the previous vectors by subtracting its scalar matrix products (α) with the other vectors. The matrix of resulting vectors $[\phi_{ro}]$ should form a diagonal, unit matrix $[m_{ro}]$ with the equation:

$$[m_{ro}] = [\phi_{ro}]^T [m_r] [\phi_{ro}]. \quad (8)$$

The remaining displacements of the rigid body eigenvectors are formed from the equation

$$[\phi_{lo}] = [D] [\phi_{ro}], \quad (9)$$

where $[D]$ corresponds to the data block DM.

Each column of $[\phi_{ro}]$ is merged with $[\phi_{lo}]$ using USET (UA,UL,UR).

4.48.8.2 Subroutine Name: READ2

1. Entry Points: READ2, READ5

2. Purpose: To compute the modal mass matrix $[MI]$, to normalize the extracted eigenvalues, and to prepare the output files LAMA and ØEIGS.

3. Calling Sequence: CALL READ2 (MAA,PHIA,SCR1,NØRM,IA,USET,MI,LAMA,ØEIGS,SCR2,EPSI,SCR3)
CALL READ5 (IPØUT)

MODULE FUNCTIONAL DESCRIPTIONS

MAA,PHIA,USET,MI,LAMA,ØEIGS are the GINØ file numbers of their respective data blocks - integer - input.

SCR1,...,SCR3 are GINØ file number of 3 scratch files - integer - input.

NØRM - Normalization method requested.

MAX	- Implies maximum component.	} Input - BCD
PØINT	- Implies specified component.	
MASS	- Implies unit modal mass matrix.	

IA - If NØRM = PØINT, IA is the component number which is set to 1.0 - integer - input.

EPSI - If EPSI \neq 0.0, the off-diagonal terms of the modal mass matrix [MI] are checked for the number which exceed EPSI.

COMMON /READ2A/Z(1)

Z(1) - Array of open core for READ2.

4.48.8.3 Subroutine Name: READ3

1. Entry Point: READ3

2. Purpose: To sort the eigenvalues in ascending order and to output the eigenvalues and eigenvectors in order. Also, to pack the eigenvectors in standard matrix format.

3. Calling Sequence: CALL READ3 (NØVECT,NCØL,SCR1,SCR2,PHI,LAMBDA)

NØVECT - Number of eigenvectors extracted - integer - input.

NCØL - Length of the vectors - integer - input.

SCR1 - GINØ file containing the unsorted eigenvalues - integer - input.

SCR2 - GINØ file containing the unsorted eigenvectors - integer - input.

PHI - GINØ file for the output sorted vectors - integers - input.

LAMBDA - GINØ file for the output sorted eigenvalues - integer - input.

COMMON /READ2A/Z(1)

Z(1) - Area of open core available to READ3.

4.48.8.4 Subroutine Name: READ4

1. Entry Point: READ4

2. Purpose: To test for close and equal roots found by either the Determinant Method or the Givens Method and make sure the corresponding vectors are orthogonal.

3. Calling Sequence: CALL READ4 (LAMAT,MPHIA,SCR1,EPSI,MAA)

LAMAT - GINØ file number of temporary storage file for the eigenvalues found - integer - input.

MPHIA - Matrix control block for PHIA - integer - input.

SCR1 - GINØ file number of a scratch file - integer - input.

EPSI - Close root test criteria - real - input.

MAA - GINØ file number of MAA - integer - input.

COMMON /READ2A/Z(1) - See READ2A

Z(1) - Array of open core for READ4.

4.48.8.5 Subroutine Name: ØRTCK

1. Entry Point: ØRTCK

2. Purpose: ØRTCK will generate the generalized mass matrix for the close roots and make the epsilon test to determine if the vectors should be orthogonalized.

3. Calling Sequence: CALL ØRTCK (X,MAA,BUFFER(1),NUM,NDIM,GM,ACCUM,EPSI)

X - Unorthogonalized eigenvectors - real - input/output.

MAA and EPSI are as described in READ4.

BUFFER - GINØ buffer.

NUM - Number of close roots - integer - input.

NDIM - Order of the problem - integer - input.

GM - Generalized mass for the close roots - real array - output.

ACCUM - Running sum of $[M_{aa}] [\phi_a]$ - real - input/output.

4.48.8.6 Subroutine Name: INVPWR

1. Entry Point: INVPWR

2. Purpose: INVPWR is the main driver for the Inverse Power Method of eigenvalue extraction.

3. Calling Sequence: CALL INVPWR

COMMON /INVPWX/K(7),M(7),LAM(7),PHI(7),SCRFIL(8),EIGSUM,LMIN,LMAX,NØEST,NDPLUS,NDMNUS,
EPS,NØRTHØ

K,M - Matrix control blocks for the input stiffness and mass matrices, [K] and [M].

LAM,PHI - Matrix control blocks for the output eigenvalue and eigenvector files.

SCRFIL(8) - GINØ file numbers for eight scratch files.

EIGSUM - GINØ file number for the eigenvalue summary file - integer.

LMIN-LMAX - Desired range for eigenvalues - real.

NØEST - Number of estimated eigenvalues in the range - integer.

NDPLUS - Number of desired positive eigenvalues - integer.

NDMNUS - Number of desired negative eigenvalues - integer.

EPS - Convergence criteria - real.

NØRTHØ - Number of roots extracted - integer.

COMMON /INVPX/Z(1)

Z(1) - Area of open core available to INVPWR.

The general methods and flowcharts used for Inverse Power Method with Shifts can be found in Section 9.3, and theoretical implications can be found in Section 10.4 of the Theoretical Manual.

4.48.8.7 Subroutine Name: INVP1

1. Entry Point: INVP1

2. Purpose: To set up a call to ADD to form

$$[A] = [K] - \lambda_0 [M] \quad (10)$$

3. Calling Sequence: CALL INVP1

COMMON /INVPWX/K(7),M(7),A

COMMON /INVPXX/LAM0

COMMON /INVP1X/Z(1)

K,M - Matrix control blocks for the input matrices.

A - GIN0 file number for the output matrix.

LAM0 - Double precision scalar multiplier.

Z(1) - Area of open core available to ADD.

4.48.8.8 Subroutine Name: INVP2

1. Entry Point: INVP2

2. Purpose: To initialize and call DEC0MP for subroutine INVPWR.

3. Calling Sequence: CALL INVP2

COMMON /INVPWX/DUM(14),A(7),L(7),XX(2),U,SCR1,SCR2,SCR3,LL,UU

COMMON /INVPXX/DUMM(12),SWITCH

COMMON /INVP2X/Z(1)

A - GIN0 file number of the input matrix - integer - input.

L,U - GIN0 file number for the lower and upper triangular matrices output from DEC0MP - integer - input.

SCR1, }
SCR2, } - Three scratch files used by DEC0MP - integer - input.
SCR3

LL,UU - GIN0 file numbers for alternate storage of L and U - integer - input.

SWITCH - 0 - Store output matrices in L and U. 1 - Store output matrices in LL and UU - integer - input.

Z(1) - Area of open core available for DEC0MP.

4.48.8.9 Subroutine Name: INVP3

1. Entry Point: INVP3

2. Purpose: To solve for an eigenvalue and eigenvector using the Inverse Power Method.

3. Calling Sequence: CALL INVP3

COMMON /INVPWX/K(7),M(7),LAM(7),PHI(7),SCRFIL(8),EIGSUM,LMIN,LMAX,N0EST,NOPLUS,NDMNUS,EPS

COMMON /INVP3X/Z(1)

MODULE FUNCTIONAL DESCRIPTIONS

See INVPWR for a description of /INVPWX/(section 4.48.8.6).

Z(1) - Open core for INVP3.

4. Method: The logic flow of INVP3 and the mathematical equations are supplied in Section 9.3. Theoretical considerations are presented in the Theoretical Manual.

INVP3 was designed with two aspects in mind: first to assure that all roots within a given region are found, and second to avoid any possibility of looping uncontrollably. To accomplish these ends, considerable testing was inserted around the mathematical equations.

5. Design Requirements: INVP3 needs sufficient storage to hold seven double precision a set vectors and four GIN0 buffers in core.

4.48.8.10 Subroutine Name: N0RM1

1. Entry Point: N0RM1
2. Purpose: To normalize a vector {x} such that its maximum component is one.
3. Calling Sequence: CALL N0RM1 (X,DIV)

COMMON /INVPWX/XX,N

N - Length of the double precision vector {x}.

X - Double precision vector {x}.

DIV - Double precision value of the divisor used to normalize {x}- output.

4.48.8.11 Subroutine Name: MTIMSU

1. Entry Point: MTIMSU
2. Purpose: To pre-multiply a vector by a matrix i.e.,:

$$\{x\} = [M] \{y\}. \quad (11)$$

3. Calling Sequence: CALL MTIMSU (Y,X,BUF) (BUF is not used)

COMMON /INVPWX/DUM(7),M(7)

COMMON /INVPXX/DUM(13),NZER0

- M - Matrix control block for matrix [M].
- X,Y - Double precision left and right hand side vectors.
- BUF - GINØ buffer.
- NZERØ - Number of zero columns of matrix [M].

4.48.8.12 Subroutine Name: XTRNSY

1. Entry Point: XTRNSY
2. Purpose: To form the dot product of two vectors

$$a = \{x\}^T \{y\}. \quad (12)$$

3. Calling Sequence: CALL XTRNSY (X,Y,A)

COMMON /INVPWX/XX,N

- N - Length of vectors {x} and {y}.
- X,Y - Double precision vectors.
- A - Double precision value of the dot product.

4.48.8.13 Subroutine Name: SUB

1. Entry Point: SUB
2. Purpose: To evaluate the vector equation

$$\{z\} = a\{x\} - b\{y\}. \quad (13)$$

3. Calling Sequence: CALL SUB (X,Y,A,B)

COMMON /INVPWX/XX,N

- N - Length of the vectors {x} and {y}.
- X,Y - Double precision vectors in the above equation. Y, upon return from SUB, contains the difference vector {z}.
- A,B - Double precision scalar multipliers.

MODULE FUNCTIONAL DESCRIPTIONS

4.48.8.14 Subroutine Name: INVFB

1. Entry Point: INVFB

2. Purpose: INVFB will perform the forward-backward substitution necessary to solve one iteration of the Inverse Power Method given by

$$([K] - \lambda_0 [M])\{w_n\} = [M]\{u_{n-1}\}. \quad (14)$$

3. Calling Sequence: CALL INVFB (X,Y,BUF)

COMMON /INFBSX/L(7),U(7)

L,U - Matrix control blocks for the lower and upper triangular matrices generated by decomposition of $([K] - \lambda_0 [M])$.

X - Double precision right hand vector, $[M]\{u_{n-1}\}$.

Y - Double precision solution vector, $\{w_n\}$.

BUF - GINØ buffer.

4. Method: INVFB is a stripped down version of GFBS. Both vectors are stored in core.

4.48.8.15 Subroutine Name: DETM

1. Entry Point: DETM
2. Purpose: To supervise the operations of the Determinant Method.
3. Calling Sequence: CALL DETM

All determinant routines use common blocks /REGEAN/ and /DETMX/.

COMMON /REGEAN/IM(7),IK(7),IEV(7),SCR1,SCR2,SCR3,SCR4,SCR5,LCØRE,RMAX,RMIN,MZ,NEV,EPSI,
RMINR,NE,NIT,NEVM,SCR6,SCR7,NFØUND,LAMA,IBUCK,NSYM

IM(7) - Matrix control block for MAA.

IK(7) - Matrix control block for KAA.

IEV(7) - Matrix control block for the eigenvectors.

SCR1,...,
SCR7 - GINØ file number of 7 scratch files - integer - input.

SCR1 - Contains [KAA] - λ [MAA].

SCR2 - Contains [LLL].

SCR3 - Contains [ULL].

LCØRE - Amount of open core reserved for starting points, determinants, scale factors
and accepted eigenvalues - integer - input.

RMAX - Maximum eigenvalue of interest - real - input/output.

RMIN - Minimum eigenvalue of interest - real - input.

MZ - Number of zero eigenvalues - integer - input.

NEV - Number of estimated eigenvalues in the range of interest - integer - input.

EPSI - Convergence criteria - real - input/output. EPSI = 1.0×10^{-11} currently.

RMINR - Lower boundary to search region. (RMINR = $-0.01 \times RMIN$)

For buckling problems a pole of geometrically increasing order is placed at
RMINR each time the search procedure points below RMIN - real - input.

NE - Number of changes in the convergence criteria - integer - input. NE is currently
set at 4.

MODULE FUNCTIONAL DESCRIPTIONS

- NIT - Number of iterations allowed to converge to an eigenvalue. NIT is currently set to 20 - integer - input.
- NEVM - Number of eigenvalues desired - integer - input.
- NFØUND - Number of eigenvalues found - integer - output.
- LAMA - GINØ file number of eigenvalue storage file - integer - input.
- IBUCK - Buckling flag. If IBUCK = 3, the current problem is a Buckling Analysis problem - integer - input.
- NSYM - Symmetric determinant flag. If NSYM = 1, symmetric decomposition is used to compute the determinant of A - integer - input.
- CØMMØN/DETMX/P(4),DETX(4),PS1(4),DET1(4),N2EV,IPSAV,IPS,IDET,IPDETA,PREC,NSTART,NDCMP,IC,NSMØVE,ITERM,IS,ND,IADD,SML1,IPDETX(4),IPDET1(4),IFAIL,K,FACT1,IFFND,NFAIL,NPØLE,ISNG
- P - The 3 trial values of the eigenvalue - double precision - input/output.
- DETX - The determinants of the 3 P's above - double precision - input/output.
- PS1 - The 3 current starting points - double precision - input/output.
- DET1 - The determinants of the 3 starting points - double precision - input/output.
- N2EV - The number of starting points - integer - input.
- IPSAV - Pointer to the accepted eigenvalues - integer - input.
- IPS - Pointer to the starting points - integer - input.
- IDET - Pointer to the determinants of the starting points - integer - input.
- IPDETA - Pointer to the scale factors of the determinants - integer - input.
- PREC - Precision of the calculations - integer - input.
- NSTART - Number of passes through the starting points - integer - output.
- NDCMP - Total number of decompositions - integer - output.
- IC - Total number of convergence criteria changes - integer - output.

FUNCTIONAL MODULE READ (REAL EIGENVALUE ANALYSIS - DISPLACEMENT)

NSMOVE - Number of starting point moves - integer - output.

ITERM - Reason for termination - integer - output.

IS - Start set counter - integer - input/output.

ND - Number of new starting points to evaluate - integer - input.

IADD - Pointer to next starting point to evaluate - integer - input/output.

SML1 - Magnitude of smallest diagonal element of [ULL] - real - output.

IPDETX - Scale factors for the determinants in DETX - integer - input/output.

IPDET1 - Scale factors for the determinants in DET1 - integer - input/output.

IFAIL - If this set of starting points produced a failure to iterate to a root,
IFAIL = 1 - integer - input/output.

K - Current iteration counter - integer - input.

FACT1 - Constant = $\text{EPSI} \cdot \text{SQRT}(\text{RMAX})$ - real - input/output.

IFFND - If an eigenvalue is accepted on a given pass through the starting points,
IFFND = 1 - integer - input/output.

NFAIL - Number of failures to iterate to a root - integer - output.

NP0LE - Order of the current pole at RMINR - integer - input/output.

ISNG - Number of singular matrices detected during decomposition - integer - input/output.

COMMON/DETDX/DZ(1)

DZ(1) - Array of open core for DETM.

4. Method: The general method used for the Determinant Method is described in Section 9.1.

5. Design Requirements: DETM needs sufficient open core to hold two double precision a set vectors and one GIN0 buffer. Open core at /DETDX/ is used for storage of starting points, determinants, powers and accepted eigenvalues.

4.48.8.16 Subroutine Name: DETM1

1. Entry Point: DETM1
2. Purpose: To compute the locations of the starting points.
3. Calling Sequence: CALL DETM1 (\$n)

MODULE FUNCTIONAL DESCRIPTIONS

- n - The statement number to which DETM1 will return if the first three starting points yield a singular matrix.

4.48.8.17 Subroutine Name: DETM2

1. Entry Point: DETM2
2. Purpose: To evaluate the determinant of ND starting points.
3. Calling Sequence: CALL DETM2

4.48.8.18 Subroutine Name: DETM3

1. Entry Point: DETM3
2. Purpose: To iterate for an eigenvalue.
3. Calling Sequence: CALL DETM3 (\$n₁, \$n₂, \$n₃)

n₁ - Return for a new starting point.

n₂ - Return for a new pass through the starting points.

n₃ - Return for problem time expired.

4.48.8.19 Subroutine Name: DETM4

1. Entry Point: DETM4
2. Purpose: To move any starting points necessary.
3. Calling Sequence: CALL DETM4

4.48.8.20 Subroutine Name: DETM5

1. Entry Point: DETM5
2. Purpose: To write out the eigenvalue analysis summary for the determinant method.
3. Calling Sequence: CALL DETM5.

4.48.8.21 Subroutine Name: DETM6

1. Entry Point: DETM6

2. Purpose: To rescale any number by powers of 10.

3. Calling Sequence: CALL DETM6 (D,PPOWER)

The arguments are both input and output to the routine and are defined by the following equation.

$$D_{out} \times 10^{PPOWER_{out}} = D_{in} \times 10^{PPOWER_{in}}, \quad (15)$$

$$.1 \leq |D_{out}| \leq 10. \quad (16)$$

D is double precision and PPOWER is integer.

4.48.8.22 Subroutine Name: FDVECT

1. Entry Point: FDVECT

2. Purpose: To build the load vector to solve for an eigenvector.

3. Calling Sequence: CALL FDVECT (SML1,PK)

SML1 - Smallest diagonal term of [ULL] - real - input.

PK - Accepted eigenvalue - double precision - input.

4.48.8.23 Subroutine Name: EADD

1. Entry Point: EADD

2. Purpose: To compute $[A] = [KAA] - p[MAA]$.

3. Calling Sequence: CALL EADD (P,PREC)

P - Trial value for one eigenvalue - double precision - input.

PREC - "2" - integer - input.

4.48.8.24 Subroutine Name: DETDET

1. Entry Point: DETDET

2. Purpose: To compute the swept determinant of [A].

3. Calling Sequence: CALL DETDET (DETA,IPPOWER,P,SML1,DLDD,IPRDLDD)

MODULE FUNCTIONAL DESCRIPTIONS

- DETA - Swept determinant of [A] - double precision - output.
- IPØWR - Scale factor of DETA - integer - output.
- P - Value of p used in computing [A] - double precision - input.
- SML1 - Value of smallest diagonal term of [ULL] - real - output.
- ØLDD - Swept determinant of previous value of p - double precision - input.
- IPRØLD - Scale factor of ØLDD - integer - input.

4.48.8.25 Subroutine Name: ARRM

1. Entry Point: ARRM
 2. Purpose: To arrange three starting points in order by the magnitude of their scaled determinants.
 3. Calling Sequence: CALL ARRM (P,D,ND)
- P - Array of 3 starting points - double precision - input/output.
 - D - Array of 3 determinants at P - double precision - input/output.
 - ND - Array of 3 scale factors of D - integer - input/output.

4.48.8.26 Subroutine Name: SUMM

1. Entry Point: SUMM
2. Purpose: To add two scaled numbers together.
3. Calling Sequence: CALL SUMM (SUM,ISUM,T1,IT1,T2,IT2,N)

The arguments are defined by the following equation:

$$\text{SUM} \times 10^{\text{ISUM}} = \text{T1} \times 10^{\text{IT1}} \pm \text{T2} \times 10^{\text{IT2}} \quad (17)$$

If N = 1, the plus sign is used, otherwise the minus sign is used.

SUM, T1, T2 are double precision.

N, IT1, ISUM, IT2 are integers.

4.48.8.27 Subroutine Name: SQRTM

1. Entry Point: SQRTM
2. Purpose: To compute the square root of a scaled number.
3. Calling Sequence: CALL SQRTM (A,IA,B,IB)

The arguments are defined by the following equation:

$$A \times 10^{IA} = \sqrt{B \times 10^{IB}}$$

A and B are double precision.

IA and IB are integers.

4.48.8.28 Subroutine Name: DETFBS

1. Entry Point: DETFBS
 2. Purpose: To solve for the eigenvector.
 3. Calling Sequence: CALL DETFBS (F,X,BUFFER,FU,NRØW)
- F - Array containing the load vector {F} - double precision - input.
- X - Array containing the eigenvector - double precision - output.
- BUFFER - GINØ buffer.
- FU - Matrix control block for [ULL] - integer - input.
- NRØW - Order of the problem (length of F and X) - integer - input.

4.48.8.29 Subroutine Name: VALVEC

1. Entry Point: VALVEC
 2. Purpose: To extract the eigenvalues and eigenvectors of a symmetric matrix.
 3. Calling Sequence: CALL VALVEC
- COMMON /GIVN/X1,MØ,MD,MRI,M1,M2,M3,M4,X9(3),RSTRT,X18(82),N,LFREQ,ØRDER,Y4,HFREQ,LAMA,NV,X8(2),NFØUND,ØEIGS,PHIA,NVER,NEVER,MAX,ITERM
- COMMON /XXVLVC/Z(1)
- /GIVN/ is used by all Givens routines.

MØ GINØ file number of the input matrix - integer - input.
 MD, MR1, M1, M2, M3, M4 - GINØ file numbers of scratch files - integer - input.
 RSTRT - '0' indicates no restart is being made - integer - input.
 N - Order of the problem - integer - output.
 LFREQ, HFREQ - Frequency range for computation of eigenvectors - real - input.
 ØORDER - Eigenvalue sort order flag - integer - input.
 LAMA, ØEIGS, PHIA - GINØ file name of the associated data blocks - integer - input.
 NV - Number of eigenvectors to compute - integer - input.
 NFØUND - Number of rigid body modes previously found - integer - input.
 NVER - Number of fails to converge on eigenvectors - integer - output.
 NEVER - Number of fails to converge on eigenvalues - integer - output.
 MAX - Maximum number of QR iterations allowed - integer - input.
 ITER - Reason for termination - integer - input.
 Z - Open core for VALVEC.

X1, X9, X18, YY, X8 are dummy variables and are not currently used.

4.48.8.30 Subroutine Name: SMLEIG

1. Entry Point: SMLEIG
 2. Purpose: To compute the eigenvalues for a 1 by 1 and 2 by 2 matrix and the eigenvector for a 1 by 1 matrix.
 3. Calling Sequence: CALL SMLEIG (D,Ø,VAL)
- D - Array of diagonal values - double precision - output.
 Ø - Array of off-diagonal values - double precision - output.
 VAL - Array of eigenvalues - double precision - output.

4.48.8.31 Subroutine TRIDI

1. Entry Point: TRIDI

2. Purpose: To tridiagonalize a symmetric matrix.

3. Calling Sequence: CALL TRIDI (D,Ø,C,A,B,AA)

D - Diagonal terms of the tridiagonal matrix - double precision - output.

Ø - Off diagonal terms of the triadiagonal matrix - double precision - output.

C - Scratch array which contains another copy of the diagonal terms at the conclusion of TRIDI - double precision - output.

A - Remainder of core - double precision - scratch.

B - Scratch array which contains the square of the off-diagonal terms - double precision - output.

AA - Remainder of core - single precision - scratch.

4.48.8.32 Subroutine Name: SICØX

1. Entry Point: SICØX

2. Purpose: To initialize the arrays in SICAS. See section 4.48.8.33.

3. Calling Sequence: CALL SICØX (D,Ø,CØS)

D - Array of diagonal values - double precision - input/output.

Ø - Array of off-diagonal values - double precision - input.

CØS - Array of cosine rotation factors - double precision - input/output.

4.48.8.33 Subroutine Name: SICØX (D,Ø,CØS)

1. Entry Point: SINCAS

2. Purpose: To compute the rotation factors for a given row.

3. Calling Sequence: CALL SINCAS (RØW,FLAG)

RØW - The number of the current row to rotate - integer - input.

FLAG - If no rotations are required for this row, FLAG = 0. Otherwise, FLAG = 1 - integer - input.

4.48.8.34 Subroutine Name: RØTAX

1. Entry Point: RØTAX
 2. Purpose: To initialize the arrays in RØTATE. See section 4.48.8.35.
 3. Calling Sequence: CALL RØTAX (Ø,SIN,CØS)
- Ø - Array of off-diagonal values - double precision - input/output.
- SIN - Array of sine rotation factors - double precision - input.
- CØS - Array of cosine rotation factors - input.

4.48.8.35 Subroutine Name: RØTATE

1. Entry Point: RØTATE
 2. Purpose: To rotate as much of the matrix as fits into core.
 3. Calling Sequence: CALL RØTATE (A,RØW,RØW1,RØW2)
- A - Partition of the matrix held in core - double precision - input.
- RØW -- The row number of current rotation row - integer - input.
- RØW1 - The row number of the first row of the matrix partition in core - integer - input.
- RØW2 - The row number of the last row of the matrix partition in core - integer - input.

4.48.8.36 Subroutine Name: EMPCØR

1. Entry Point: EMPCØR
 2. Purpose: To empty core of a triangular matrix partition.
 3. Calling Sequence: CALL EMPCØR (MT1,MT2,PT,PC,FRSRØW,MIDRØW,LASRØW,NX,A,Z)
- MT1 - GINØ file name of the first output file - integer - input.
- MT2 - GINØ file name of the second output file - integer - input.
- PT - Precision (1 = single precision, 2 = double precision) of the input matrix - integer - input.
- PC - Precision (1 = single precision, 2 = double precision) of the output matrix - integer - input.
- FRSRØW - The row number of the first row in core - integer - input.
- MIDRØW - When the current row is greater than MIDRØW, write the remainder of the matrix

onto MT2 - integer - input.

LASRØW - The row number of the last row in core - integer - input.

NX - Order of the matrix - integer - input.

A - Storage containing the triangular matrix - double precision - input.

Z - GINØ buffer - input.

4.48.8.37 Integer Function Name: FILCØR

1. Entry Point: FILCØR

2. Purpose: To fill core with a triangular matrix.

3. Calling Sequence: RØW = FILCØR (MT1,MT2,PC,FRSRØW,MIDRØW,NX,A,NZA,Z)

MT1 - GINØ file name of the first part of the matrix (up to MIDRØW) - integer - input.

MT2 - GINØ file name of the rest of the matrix - integer - input.

PC - Precision (1 = single precision, 2 = double precision) of the matrix in core - integer - input.

FRSRØW - The row number of the first row of the matrix to be read - integer - input.

MIDRØW - Breakpoint of the matrix - integer - input.

NX - Order of the matrix - integer - input.

A - Core storage to hold the matrix - integer - input.

NZA - Number of single precision words at A - integer - input.

Z - GINØ buffer - input.

RØW - The row number of the last row read into A - integer - output.

4.48.8.38 Subroutine Name: QRITER

1. Entry Point: QRITER

2. Purpose: To obtain the eigenvalues of a tridiagonal matrix by the Ortega - Kaiser QR iteration technique.

3. Calling Sequence: CALL QRITER (VAL,Ø,LØC,QR)

FUNCTIONAL MODULE READ (REAL EIGENVALUE ANALYSIS - DISPLACEMENT)

- VAL - Diagonal terms of the tridiagonal matrix on input, reordered eigenvalues on return - double precision - input/output.
- Ø -- Squares of the off-diagonal terms of the tridiagonal matrix. These are iteratively reduced to zero - double precision - input/output.
- LØC - Array giving the order in which each eigenvalue is found - integer - output.
- QR -- QR ≠ 0 implies that the eigenvalues already exist at VAL and this is an ordering call only - integer - input.

4.48.8.39 Subroutine Name: WILVEC

1. Entry Point: WILVEC
 2. Purpose: To compute eigenvectors by the Wilkinson method.
 3. Calling Sequence: CALL WILVEC (D,Ø,VAL,VLØC,V,F,P,Q,R,VEC,NX,SVEC)
- D - Array of diagonal values - double precision - input.
- Ø - Array of off-diagonal values - double precision - input.
- VAL - Array of eigenvalues - double precision - input.
- VLØC - Array of original ordering of eigenvalues - integer - input.
- V,F,P,Q,R - Scratch arrays of length equal to the order of the problem - double precision - input/output.
- VEC - Array of core to store vectors - double precision - scratch.
- NX - Not used
- SVEC - Array of core to store vectors - single precision - scratch.

4.48.8.40 Subroutine Name: INVERT

1. Entry Point: INVERT
 2. Purpose: To drive INVTR (see section 4.48.8.41) to compute the inverse of an upper or lower triangular matrix.
 3. Calling Sequence: CALL INVERT (IA,IB,SCRI)
- COMMON /INVTRX/Z(1)

- GINØ file name of an upper or lower triangular matrix - integer - input.
- IB - GINØ file name of the inverse of the matrix on IA - integer - input.
- SCR1 - GINØ file name of a scratch file - integer - input.
- Z - Open core for INVERT.

4.48.8.41 Subroutine Name: INVTR

1. Entry Point: INVTR
2. Purpose: To invert a triangular matrix.
3. Calling Sequence: CALL INVTR (X,DX)

COMMON /INVTRX/FA(7),FB(7),SCRFIL,NX,PREC

- X - Open core for INVTR - single precision - scratch.
- DX - Open core for INVTR (same address as X) - double precision - scratch.
- FA - Matrix control block for the input matrix - integer - input.
- FB - Matrix control block for the output matrix - integer - input.
- SCRFIL - GINØ file name of a scratch file - integer - input.
- NX - Length in words of X - integer - input.
- PREC - Precision (1 = single precision, 2 = double precision) of the computation - integer - input.

4.48.8.42 Subroutine Name: READ6

1. Entry Point: READ6
2. Purpose: To merge the rigid body eigenvectors with vectors computed by GIVENS.
3. Calling Sequence: CALL READ6 (IRBM,IMGIV,NR,IPHA)

COMMON /READ6X/Z(1)

- IRBM - GINØ file name for eigenvectors computed by READ1 - integer - input.
- IMGIV - GINØ file name for eigenvectors computed by GIVENS - integer - input.
- NR - Number of eigenvectors computed by READ1 - integer - input.

MODULE FUNCTIONAL DESCRIPTIONS

IPHIA - GINØ file name for merged eigenvectors - integer - input.

Z - Open core for READ6.

4.48.8.43 Subroutine Name: READ7

1. Entry Point: READ7

2. Purpose: To extract eigenvalues and eigenvectors previously found and to place them on the internal scratch files. The vectors are renormalized to "MASS".

3. Calling Sequence: CALL READ7 (NR,ØLAMA,ØPHIA,NLAMA,NPHIA)

where:

NR is the number of old eigenvectors found on ØPHIA.

ØLAMA,ØPHIA,NLAMA,NPHIA are the GINØ file numbers of the old and new LAMA and PHIA files.

4. Design Requirements: Open core is at /READ1A/.

4.48.8.44 Subroutine Name: FCNTL

1. Entry Point: FCNTL

2. Purpose: FCNTL is the main driver for the real Tridiagonal Reduction (FEER) method.

3. Calling Sequence: CALL FCNTL

COMMON/FEERCX/IFKAA(7),IFMAA(7),IFLELM(7),IFLVEC(9),SR1FLE,SR2FLE,SR3FLE,SR4FLE,SR5FLE,
SR6FLE,SR7FLE,SR8FLE,DMPFLE,NØRD,XLMDBA,NEIG,MØRD,IBK,CRITF,NØRTHØ,IFLRVA,IFLRVC

IFKAA,IFMAA - Matrix control blocks for the input stiffness and mass matrices.

IFLELM,IFLVEC - Matrix control blocks for the output eigenvalue and eigenvector files.

SR1FLE

SR2FLE

:

SR8FLE

- GINØ file numbers for eight scratch files (integer).

DMPFLE

- GINØ file number for the eigenvalue summary file (integer).

NØRD

- Problem size (set internally using the dimension of the stiffness matrix).

XLMDBA

- User specified shift.

NEIG

- Desired number of eigenvalues specified by the user.

FUNCTIONAL MODULE READ (REAL EIGENVALUES ANALYSIS - DISPLACEMENT)

MØRD - Reduced order of the problem (set internally).

IBK - Buckling option indicator (set internally).

CRITF - The user specified (or default) desired theoretical accuracy of the eigenvalues expressed as a percentage.

NØRTHØ - Number of orthogonal vectors in present set (includes previously computed vectors).

IFLRVA,IFLRVC - GINØ file numbers for the previously computed eigenvalues and eigenvectors.

CØMMØN/FEERXX/LAMBDA,CNDFLG,ITER,TIMED,L16,IØPTF,EPX,NØCHNG,IND,LMBDA,IFSET,NZERØ,NØNUL,IDIAG,MRANK,ISTART

LAMBDA - Value of the shift actually used (double precision).

CNDFLG - Termination indicator.

ITER - Number of starting points used.

TIMED - Not currently activated.

L16 - DIAG 16 output indicator set internally.

IØPTF - Specified shift option indicator set internally.

EPX - Orthogonality convergence criteria.

NØCHNG - Theoretical error parameter.

IND - Not activated.

LMBDA - Not activated (double precision).

IFSET - Internally computed shift indicator.

NZERØ - Number of previously obtained eigenvectors.

NØNUL - Number of vector iterations.

IDIAG - Not activated.

MRANK - Maximum rank of the problem.

ISTART - FEER start time.

MODULE FUNCTIONAL DESCRIPTIONS

COMMON/FEER3X/Z(1)

Z(1) - Area of open core used by FCNTL.

COMMON/ØPINV/MCBLT(7),MCBSMA(7),MCBVEC(7),MCBRM(7)

MCBLT - Lower triangular matrix, [L], control block.

MCBSMA - Conditioned [M] matrix control block.

MCBVEC - Orthogonal vector file control block.

MCBRM - Trial vector, [V] or $([C]^{-1})^T[V]$ control block.

4. Design Requirements: The FEER method requires sufficient core for four GINØ buffers and five vectors. The method can be performed in single or double precision. Default values of key parameters in COMMON/FEERCX/ are provided in the block data subprogram FEERBD.

5. Subroutine Glossary for the FEER method.

FCNTL
FEER1
FEER2
FEER3
FEER4
FNXTV,FNXTVC
FQRW,FQRWV
FRBK,FRBK2
FRMAX
FRMLT,FRMLTD
FRMLTX,FRMLTA
FRSW,FRSW2

6. The user can obtain special detailed information relating to the generation of the reduced problem size, the elements of the reduced tridiagonal matrix, computed error bounds and other numerical tests by requesting DIAG 16 in the NASTRAN Executive Control Deck.

The meaning of this information is explained below in the order in which it appears in the DIAG 16 output.

ØORDER - The order of the unreduced problem (size of the $[K_{aa}]$ matrix)

MAX RANK - The maximum number of existing finite eigensolutions as initially detected by FEER

RED ØORDER - The order of the reduced eigenproblem which will be solved to obtain the number of accurate solutions requested by the user

ØRTH VCT - The number of previously computed accurate eigenvectors on the eigenvector file which were generated prior to restart or by the NASTRAN rigid body mode generator

FUNCTIONAL MODULE READ (REAL EIGENVALUES ANALYSIS - DISPLACEMENT)

USER SHIFT - Used only in frequency problems. The user specified shift after conversion from cycles to radians - squared

INTERNAL SHIFT- Used only in frequency problems. A small positive value automatically computed to remove singularities if the user has specified a zero shift. Otherwise, the negative of the user shift

SINGULARITY CHECK - PASS: the shifted stiffness matrix is non-singular
****: the number of internal shifts needed to remove stiffness matrix singularities

TRIDIAGONAL ELEMENTS ROW j, **, ***, **** - Lists the computed tridiagonal elements of the reduced eigenmatrix:

j - Matrix row
** - Diagonal element
*** - Off-diagonal element
**** - First estimate of off-diagonal element in the next row

ORTH ITER - The number of times a reorthogonalization of a trial vector has been performed

MAX PROJ - The maximum projection of the above trial vector on the previously computed accurate trial vectors (prior to the current reorthogonalization)

NORMAL FACT - The normalization factor for the reorthogonalized trial vector

OPEN CORE NOT USED *** FEER3 - open core not used by Subroutine FEER3, in single-precision words

FEER QRW ELEMENT *, ITER **, ***, RATIO ****, PROJ *****:

* - The internal eigenvalue number in the order of its extraction by FEER
** - The number of inverse power iterations performed to extract the associated eigenvector of the reduced system (this is not a physical eigenvector)
*** - If a multiple root has been detected, the number of times that the previous multiple-root, reduced-system eigenvectors have been projected out of the current multiple-root eigenvector before repeating the inverse power iterations
**** - The absolute ratio of maximum, reduced-system eigenvector elements for successive inverse power iterations
***** - The maximum projection of a current multiple-root eigenvector on previously computed eigenvectors for the same root.

PHYSICAL EIGENVALUE *, **, THEOR ERROR *** PERCENT, PASS OR FAIL:

* - The internal eigenvalue number in the order of its extraction by FEER
** - The associated physical eigenvalue (λ for buckling problems, ω^2 for frequency problems)

MODULE FUNCTIONAL DESCRIPTIONS

- *** - Theoretical upper bound on the relative eigenvalue error, in percent
- PASS - The computed error is less than or equal to the allowable specified on the EIGB or EIGR bulk data card (default is .001% where n is the order of the stiffness matrix)
- FAIL - The computed error is greater than the allowable and this mode is not accepted for further processing
- OPEN CORE NOT USED *** FEER4 - open core not used by Subroutine FEER4, in single precision words
- FEER COMPLETE *, **, ***, ****
- * - The remaining CPU time available following decomposition of the shifted stiffness matrix, in seconds (the total time is specified on the TIME card in the Executive Control Deck)
- ** - The remaining CPU time, in seconds after completing Subroutine FEER3
- *** - The remaining CPU time, in seconds after completing Subroutine FEER4
- **** - The total operation count for FEER after decomposition of the shifted stiffness matrix. One operation is considered to be a multiplication or division followed by an addition

4.48.8.45 Subroutine FEER1

1. Entry Point: FEER1
2. Purpose: To set up the call to SADD to form

$$[K] = [K] + \lambda[M].$$

3. Calling Sequence: CALL FEER1

COMMON/FEERCX/FILEK(7),FILEM(7),SCR1

FILEK,FILEM - Matrix control blocks for the input matrices.

SCR1 - GINØ file number for the output matrix.

COMMON/FEERX/LAMBDA

LAMBDA - Input shift parameter (double precision).

COMMON/FEER1X/Z(1)

Z(1) - Area of open core used by FEER1.

4. Design Requirements: FEER1 can perform single or double precision operations.

4.48.8.46 Subroutine FEER2

1. Entry Point: FEER2

2. Purpose: To set up the call to SDCOMP to perform matrix decomposition. Only the output lower triangular matrix is used in the FEER method.

3. Calling Sequence: CALL FEER2 (IRET)

IRET - Output singularity indicator (integer).

COMMON/FEERCX/IFKAA(7),IFMAA(7),IFLELM(7),IFLVEC(7),SR1FLE,SR2FLE,SR3FLE,SR4FLE,SR5FLE,
SR6FLE,SR7FLE,SR8FLE,DMPFLE,NORD,XLMBDA,NEIG,MORD,IBK,CRITF,NORTH0,IFLRVA,IFLRVC

IFKAA - Input control block parameters.

IFLELM - GIN0 file number of the input matrix.

IFLVEC - Matrix control block for the output lower triangular matrix.

SR3FLE
SR4FLE
SR5FLE
SR6FLE - Scratch files used by SDCOMP.

IBK - Buckling indicator used to determine if the Cholesky method is to be used in SDCOMP.

See FCNTL for the remaining definitions (Section 4.48.8.44).

COMMON/FEERXX/DUMM(12),IFSET

IFSET - Internal shift indicator used to determine if the Cholesky method is to be used.

COMMON/OPINV/MCBLT(7)

MCBLT(7) - Matrix control block for the output lower triangular matrix.

COMMON/FEER2X/Z(1)

Z(1) - Area of open core used by SDCOMP.

4. Design Requirements: FEER2 can perform single or double precision operations.

MODULE FUNCTIONAL DESCRIPTIONS

4.48.8.47 Subroutine FEER3

1. Entry Point: FEER3
2. Purpose: FEER3 obtains the reduced tridiagonal matrix for the FEER method of eigenvalue extraction.
3. Calling Sequence: CALL FEER3

COMMON/FEERCX/

COMMON/FEERXX/

COMMON/ØPINV/

See FCNTL for a description of /FEERCX/, /FEERXX/ and /ØPINV/ (Section 4.48.8.44).

COMMON/FEER3X/Z(1)

Z(1) - Area of open core used by FEER3.

4. Design Requirements: FEER3 can perform single or double precision operations.

4.48.8.48 Subroutine FEER4

1. Entry Point: FEER4
2. Purpose: FEER4 obtains from the reduced tridiagonal matrix, the physical eigenvalues and eigenvectors of the reduced problem.
3. Calling Sequence: CALL FEER4

COMMON/FEERCX/

COMMON/FEERXX/

COMMON/ØPINV/

See FCNTL for a description of /FEERCX/, /FEERXX/ and /ØPINV/ (Section 4.48.8.44).

COMMON/FEER4X/Z(1)

Z(1) - Area of open core used by FEER4.

4. Design Parameters: FEER4 can perform single or double precision operations.

4.48.8.49 Subroutines FNXTV, FNXTVC

1. Entry Points: FNXTV (single precision)
FNXTVC (double precision)

FUNCTIONAL MODULE READ (REAL EIGENVALUES ANALYSIS - DISPLACEMENT)

Purpose: To perform the tridiagonal reduction algorithm for FEER3.

3. Calling Sequence: CALL FNXTV (V1,V2,V3,V4,V5,ZB,IFN)

CALL FNXTVC (V1,V2,V3,V4,V5,ZB,IFN)

V1 - Space for the previous current trial vector. Initially null.
V2 - Space for the current trial vector. Initially a pseudo-random start vector.
V3,V4,V5 - Working space for three vectors.
ZB - Working space for a single GINØ buffer.
IFN - Number of trial vectors extracted. Initially zero.

COMMON/FEERCX/

COMMON/FEERXX/

COMMON/ØPINV/

See FCNTL for a description of /FEERCX/, /FEERXX/ and /ØPINV/ (Section 4.48.8.44).

4. Design Requirements: FNXTV and FNXTVC are intended to be used in conjunction with the FEER3 subprogram that performs the necessary initialization, restart and termination processes. FEER3 either invokes FNXTV or FNXTVC according to the precision of the required input files. All vectors are of dimension consistent with the initial problem size.

4.48.8.50 Subroutines FQRW, FQRWV

1. Entry Point: FQRW (single precision)

FQRWV (double precision)

2. Purpose: To solve the reduced-system eigenvalue problem and obtain the associated theoretical errors and eigenvectors.

3. Calling Sequence: CALL $\begin{smallmatrix} \text{FQRW} \\ \text{FQRWV} \end{smallmatrix}$ (M,E,ER,A,B,W,P,Q,XM,INT,ZB,SFLE,MCBC)

M - Size of the reduced problem
E - Working space for vector of reduced-problem eigenvalues.
ER - Intermediate theoretical error vector associated with the reduced system.
A - Vector of diagonal elements of the reduced tridiagonal matrix.
B - Vector of off-diagonal elements of the reduced tridiagonal matrix.
W,P,Q,XM - Working space for four vectors of dimension associated with the reduced problem.

MODULE FUNCTIONAL DESCRIPTIONS

- INT - Vector of pivot flags of dimension associated with the reduced problem (logical).
- ZB - Working space for a single GINØ buffer.
- SRFLE - Input GINØ file number of [Y] set (integer).
- MCBC(7) - Matrix control block for output [Y] vector set.

COMMON/FEERXX/LAMBDA

- LAMBDA - Shift parameter (double precision).

4. Design Requirements: FQRW and FQRWV are intended to be used in conjunction with the FEER4 subprogram that performs the necessary initialization and termination processes. FEER either invokes FQRW or FQRWV according to the precision of the required input files. All vectors are of dimension consistent with the reduced problem size plus one.

4.48.8.51 Subroutines FRBK, FRBK2, FRSW, FRSW2

1. Entry Points: FRBK (single precision)
FRSW
FRBK2 (double precision)
FRSW2

2. Purpose: To perform the operational inverse algorithms.

3. Calling Sequence: CALL ... (V1,V2,V3,VB).

- V1 - Input vector.
- V2 - Working space for one vector.
- V3 - Output vector.
- VB - Vector of matrix elements used in a matrix multiply.

The output vector is defined as follows for calls to FRBK or FRBK2:

$$\{V3\} = ([L]^{-1})^T [D] [L]^{-1} [M] \{V1\}$$

The output vector is defined as follows for calls to FRSW or FRSW2:

$$\{V3\} = [C]^{-1} [M] ([C]^{-1})^T \{V1\}$$

COMMON/ØPINV/MCBLT(7),MCBSMA(7)

- MCBLT - Matrix control block for ([L][D]) or [C].
- MCBSMA - Matrix control block for [M].

4. Design Requirements: FRBK and FRBK2 are intended to be used to perform the specialized operational inverse algorithms associated with the FEER method of eigenvalue extraction. The subroutines use the direct output of the SDCOMP subprogram.

4.48.8.52 Subroutine FRMAX

1. Entry Point: FRMAX
2. Purpose: To obtain parameters for FCNTL when the shift parameter is internally computed.
3. Calling Sequence: CALL FRMAX(IFK,IFM,N,IPR,RSN,RSM).

IFK - GINØ file number for the stiffness matrix, [K].
 IFM - GINØ file number for the mass or differential stiffness matrix, [M].
 N - Dimension of the matrices.
 IPR - Precision of the matrices.
 RSN - $\left| \frac{k_{ij}}{m_{ij}} \right|_{\min}$ (double precision).
 RSM - $\left| \frac{k_{ij}}{m_{ij}} \right|_{\max}$ (double precision).

4. Design Requirements: FRMAX can operate with either single or double precision input matrices. Output is always double precision.

4.48.8.53 Subroutines FRMLT, FRMLTD, FRMLTA, FRMLTX

1. Entry Points: FRMLT (single precision)
 FRMLTA (single precision)
 FRMLTD (double precision)
 FRMLTX (double precision)
2. Purpose: To perform standard matrix transpose multiply and non-standard matrix transpose multiply using the lower triangular matrix output of SDCOMP.

4.48.9 Design Requirements

Design requirements are peculiar to the method chosen. The appropriate subroutines should be consulted. Nine scratch files are used.

4.48.10 Diagnostic Messages

Messages output from READ are peculiar to the individual subroutine.

4.48.11 Mathematical Considerations for the Inverse Power Method

The algorithm for finding the eigenvalues and eigenvectors of

$$([K] - \lambda[M]) [\Phi] = 0, \quad (19)$$

is given as follows.

1. The iteration algorithm is given by the following equations.

$$([K] - \lambda_0 [M]) \{W_n\} = [M] \{u_{n-1}\}, \quad (20)$$

$$\{\bar{u}_n\} = \frac{1}{C_n} \{W_n\}, \quad (21)$$

$$\{u_n\} = \{\bar{u}_n\} - \sum_i (\{\phi_i\}^T [M] \{\bar{u}_n\}) \{\phi_i\}, \quad (22)$$

where C_n is the absolute value of the maximum component of $\{W_n\}$. The sum over i extends over all previously extracted eigenvectors.

2. Form

$$\{F_n\} = [M] \{u_n\}. \quad (23)$$

3. Compute

$$\alpha_n = (\{u_n\}^T \{F_n\})^{1/2}. \quad (24)$$

4. Compute

$$\{\delta u_n\} = \frac{\{u_n\}}{\alpha_n} - \frac{\{u_{n-1}\}}{\alpha_{n-1}}. \quad (25)$$

5. Compute

$$\{\delta F_n\} = [M] \{\delta u_n\}. \quad (26)$$

6. Compute the approximate eigenvalue

$$\lambda_1 = \frac{1}{C_n} \frac{\alpha_{n-1}}{\alpha_n}. \quad (27)$$

7. For the rapid convergence test, compute

$$\eta = (\{\delta u_n\}^T \{\delta F_n\})^{1/2}. \quad (28)$$

8. For normal convergence, compute

$$\frac{\lambda_2}{\lambda_1} = \frac{\{\delta u_n\}^T \{\delta F_{n-1}\}}{\eta^2}, \quad (29)$$

and

$$\delta_n = (\{\delta u_n\}^T \{\delta F_n\}) / (1 - \frac{\lambda_2}{\lambda_1})^2. \quad (30)$$

9. For the shift decision test, compute

$$h_{1,n} = \frac{\lambda_{1,n} - \lambda_{1,n-1}}{R_0}. \quad (31)$$

and

$$k = \frac{\log_{10} \left(\frac{\sqrt{\delta_n}}{A\epsilon} \right)}{\log_{10} (\lambda_2/\lambda_1)}. \quad (32)$$

10. For the λ_2 reliability test, compute

$$h_{2,n} = \frac{\lambda_{2,n} - \lambda_{2,n-1}}{R_0} \quad (33)$$

The above equations, along with the proper logic given in Figure 1 form the basis for the Inverse Power Method.

MODULE FUNCTIONAL DESCRIPTIONS

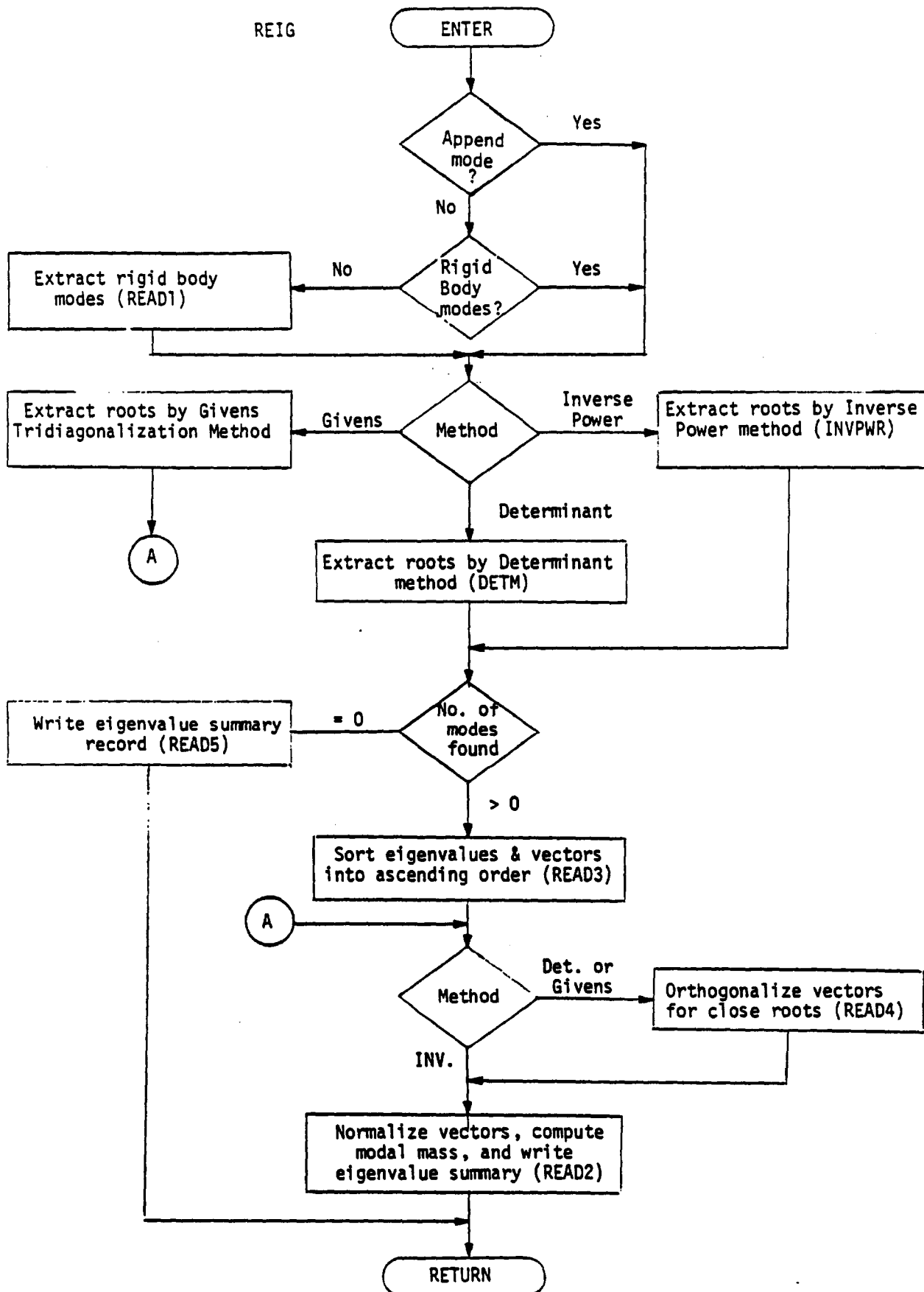


Figure 1.(a) Flowchart for module READ.

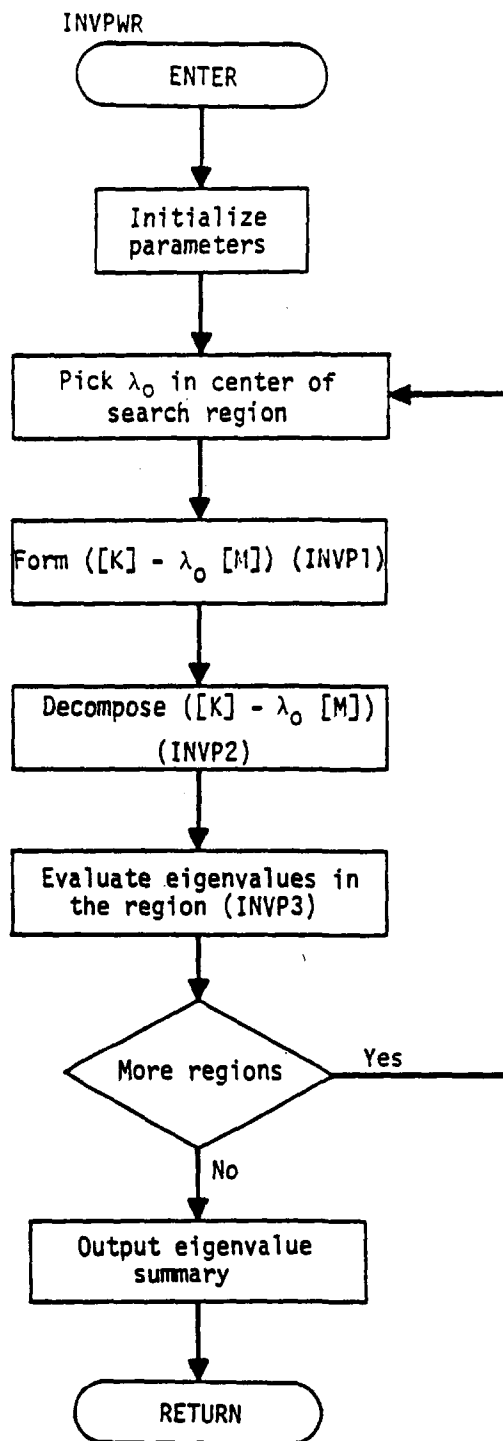


Figure 1.(b) Flowchart for module READ

MODULE FUNCTIONAL DESCRIPTIONS

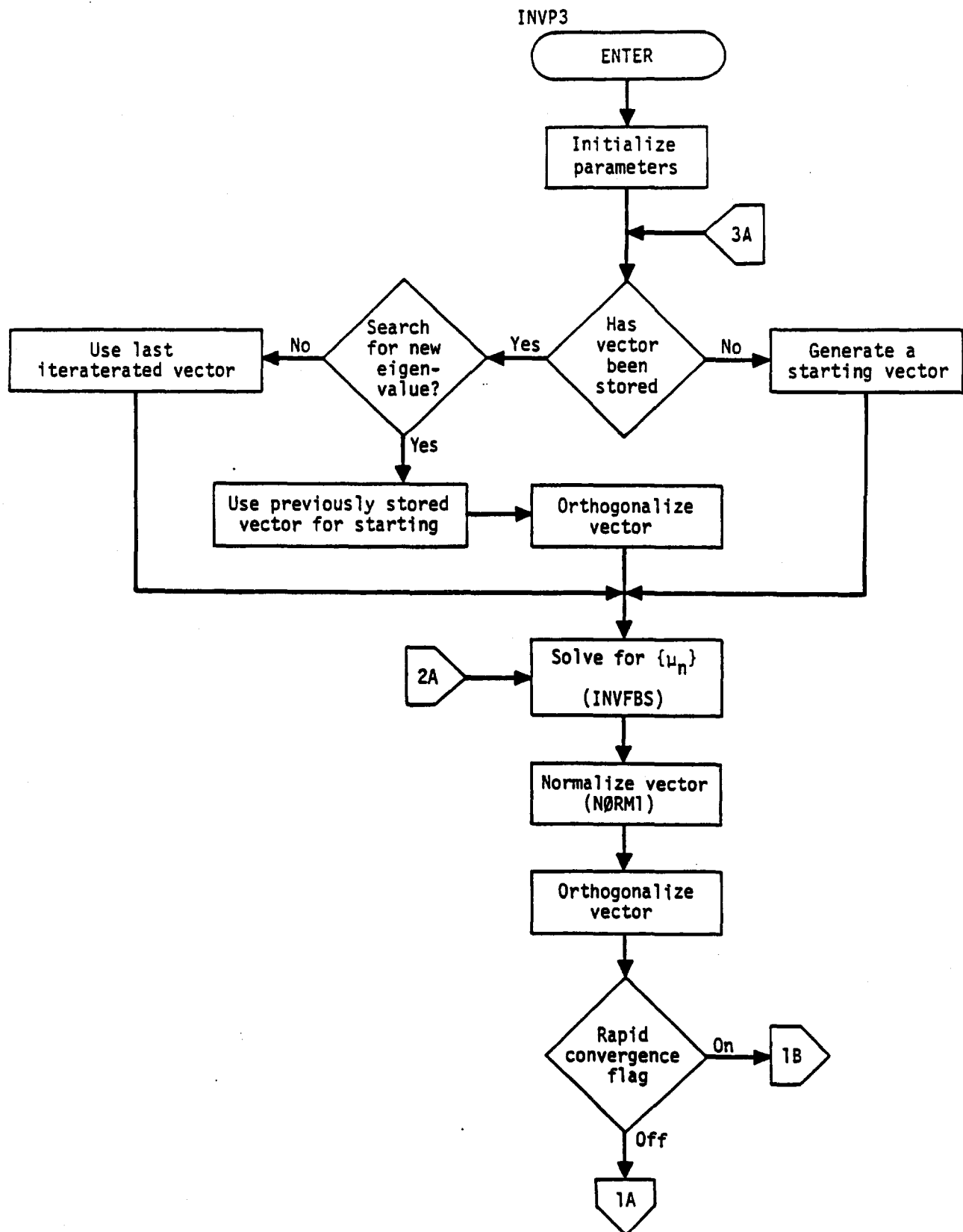


Figure 1.(c) Flowchart for module READ

FUNCTIONAL MODULE READ (REAL EIGENVALUE ANALYSIS - DISPLACEMENT)

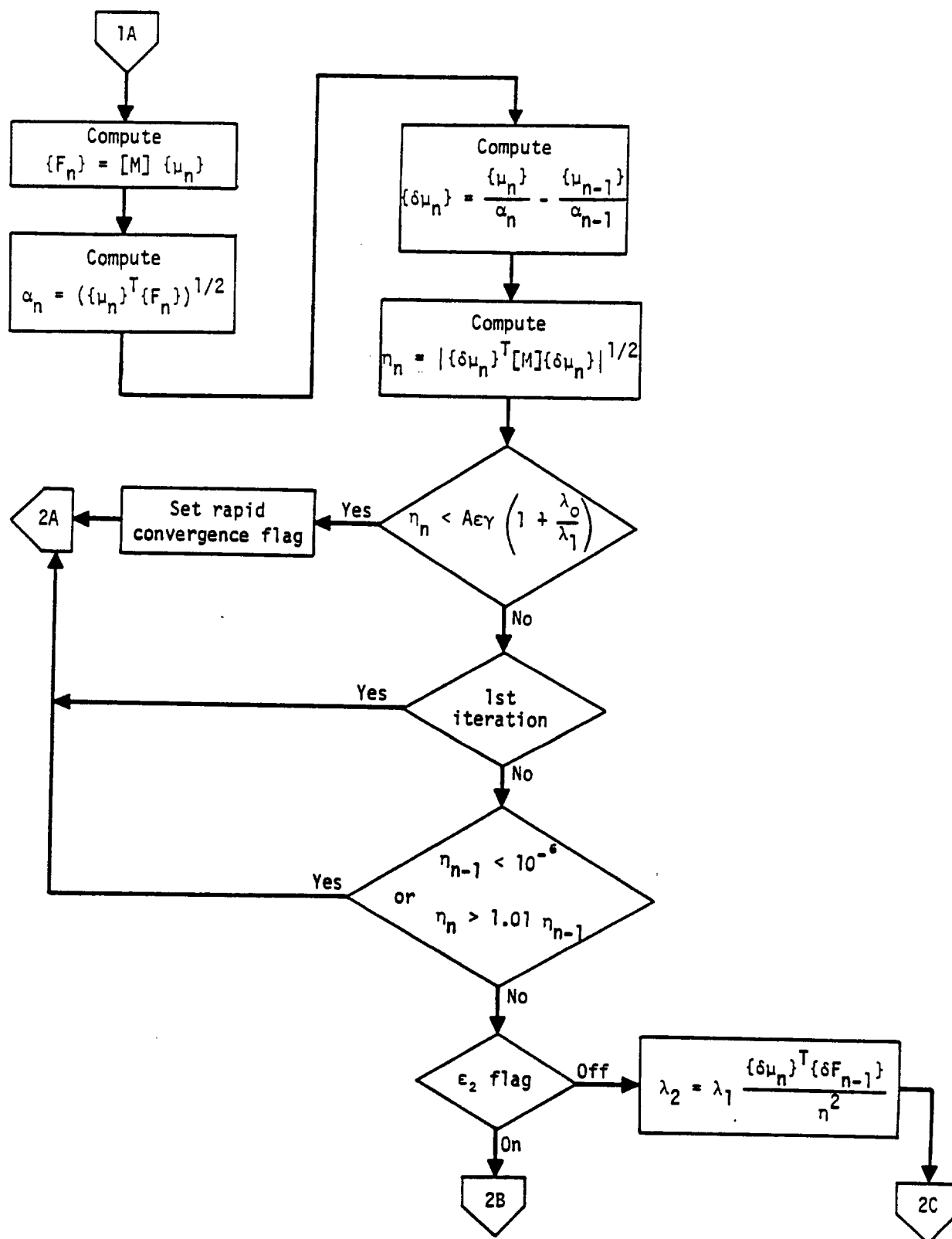


Figure 1.(d) Flowchart for module READ

MODULE FUNCTIONAL DESCRIPTIONS

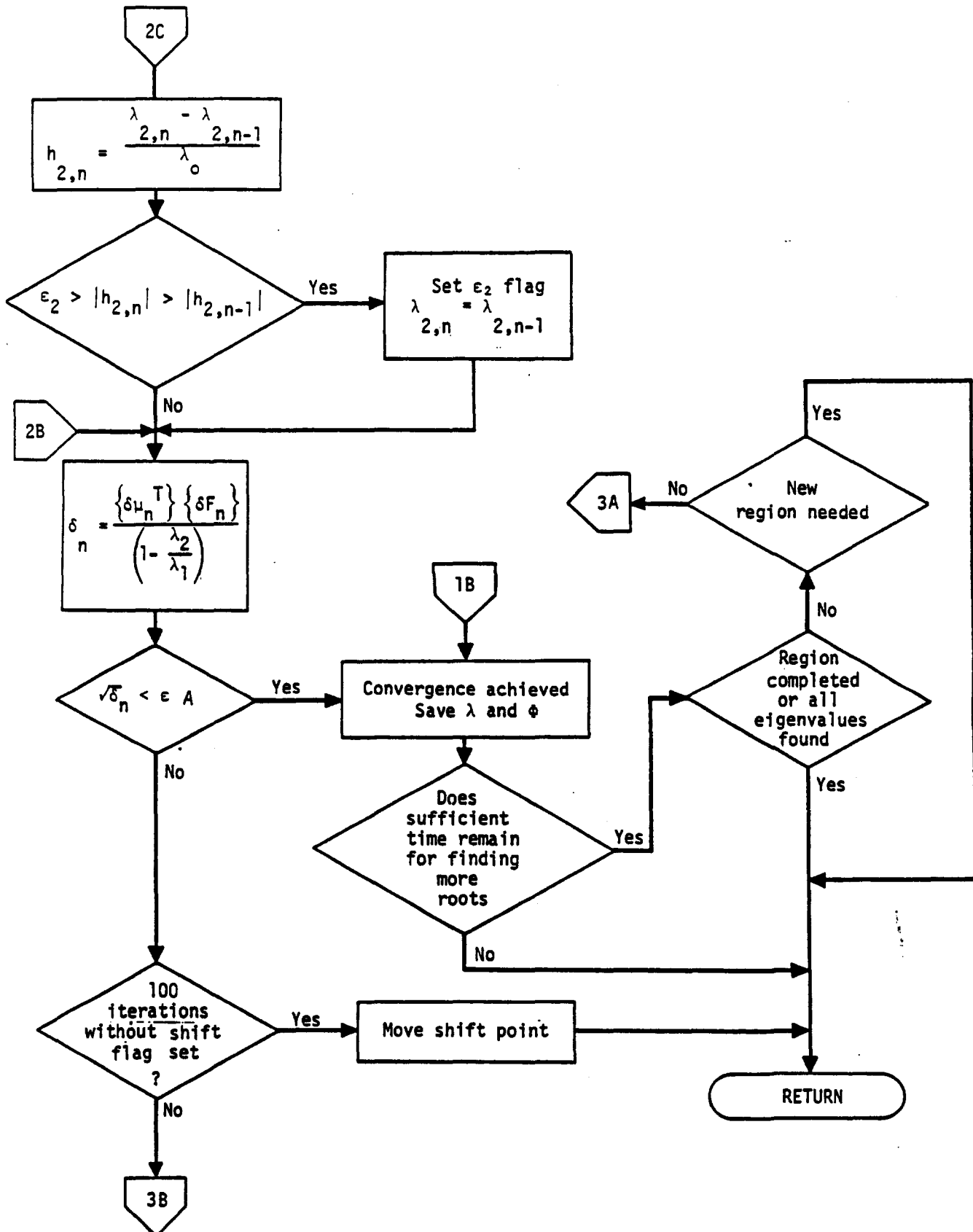


Figure 1.(e) Flowchart for module READ.

FUNCTIONAL MODULE READ (REAL EIGENVALUE ANALYSIS - DISPLACEMENT)

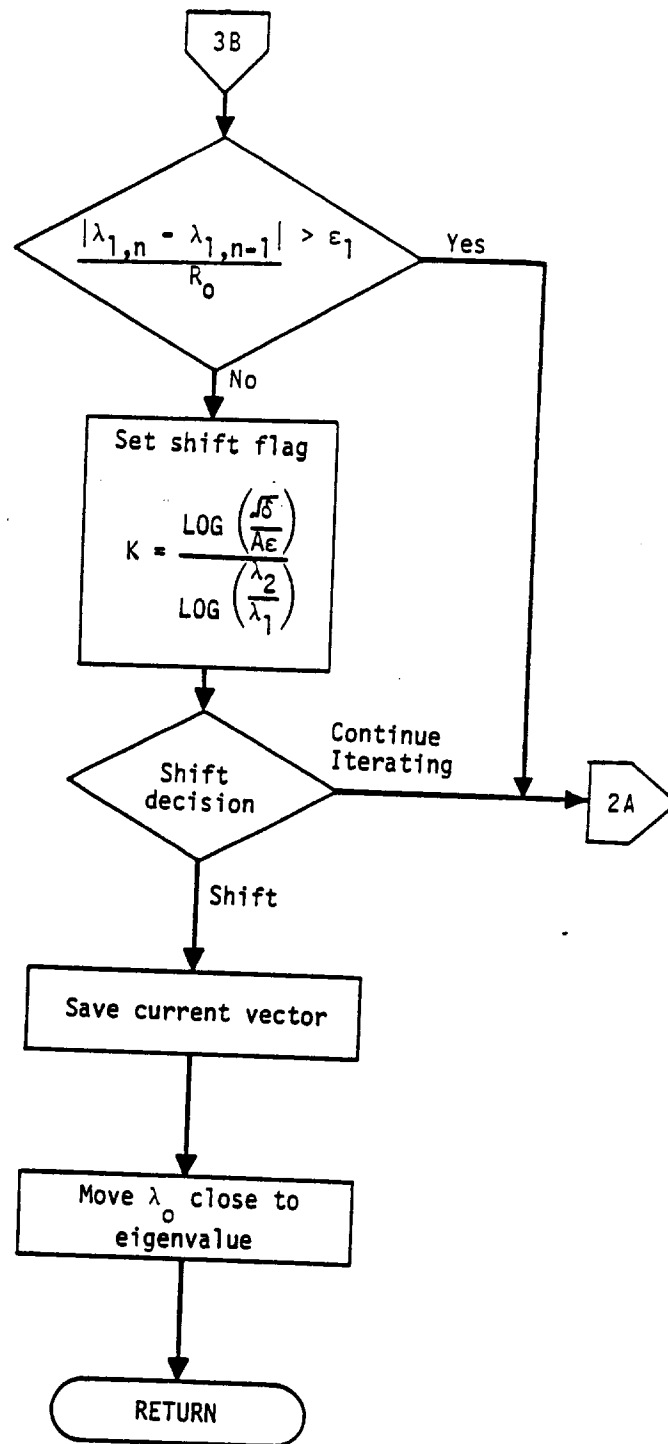


Figure 1.(f) Flowchart for module READ

4.49 FUNCTIONAL MODULE DSMG1 (DIFFERENTIAL STIFFNESS MATRIX GENERATOR - PHASE 1)

4.49.1 Entry Point: DSMG1

4.49.2 Purpose

To generate the differential stiffness matrix, $[K_{gg}^d]$, which is used in two Rigid Formats: Static Analysis with Differential Stiffness and Buckling Analysis.

4.49.3 DMAP Calling Sequence

DSMG1 CASECC,GPTT,SIL,EDT,UGV,CSTM,MPT,ECPT,GPCT,DIT/KDGG/V,N,DSCØSET \$

4.49.4 Input Data Blocks

CASECC - Case Control Data Table.
 GPTT - Grid Point Temperature Table.
 SIL - Scalar Index List.
 EDT - Element Deformation Table.
 UGV - Initial Approximation to the Displacement Vector - g set.
 CSTM - Coordinate System Transformation Matrices.
 MPT - Material Properties Table.
 ECPT - Element Connection and Properties Table.
 GPCT - Grid Point Connection Table.
 DIT - Direct Input Tables.

Notes:

1. A fatal error exists if CASECC is purged.
2. A fatal error exists if there is a temperature load and the GPTT is purged.
3. A fatal error exists if there is a temperature load and the SIL is purged.
4. A fatal error exists if an element deformation set is requested by the user in case control and EDT is purged.
5. A fatal error exists if UGV is purged.
6. CSTM can be purged. However, if some grid point of the model is not in basic coordinates and the CSTM is purged, a fatal error occurs.

7. If the MPT is purged and some element references a material property, a fatal error occurs.
8. A fatal error occurs if the ECPT is purged.
9. A fatal error occurs if the GPCT is purged.
10. If some material property is temperature dependent, DIT cannot be purged.

4.49.5 Output Data Blocks

KDGG - Partition of differential stiffness matrix - g set.

Note: KDGG cannot be pre-purged. A fatal error occurs if it is.

4.49.6 Parameters

DSCØSET - Output-integer-no default value. The set identification number of a DSFACT bulk data card chosen by the user in his Case Control Deck. If no such set was specified by the user, DSCØSET is set to -1.

4.49.7 Method

The module driver, DSMG1, is a very short routine whose only function is to call the two principal subroutines of the module, DS1 and DS1A, which accomplish the two phases of the module. The first phase of the module is incorporated in subroutine DS1. This routine creates the scratch file ECPTDS (GINØ file number 301) by appending to each element in the ECPT data block for which differential stiffness is defined an element deformation, an average element loading temperature for all elements except the conical shell element and the isoparametric solid elements (a loading temperature at each grid point is appended for these elements) and the proper components of the displacement vector, UGV. It should be noted that although element deformations are defined only for rods, tubes, beams, and bars, an element deformation is attached to each element written on the ECPTDS scratch data block. The elements admissible to the ECPTDS scratch data block are defined by the common block GPTA1.

The flow of DS1 is given in the following steps:

1. The length of variable core is determined, buffers are defined at the bottom of open core, and zero pointers to, and lengths of, subarrays in open core are initialized.

FUNCTIONAL MODULE DSMG1 (DIFFERENTIAL STIFFNESS MATRIX GENERATOR PHASE 1)

2. The Case Control data block, CASECC, is read to determine the element deformation set number, the loading temperature set number, and the differential stiffness coefficient set number.
3. If there is no temperature load, go to step 4. If there is a temperature load, the Grid Point Temperature Table data block, GPTT, is positioned to the proper thermal record.
4. If a non-zero element deformation set number was read from CASECC, the element deformation set is read into open core.
5. The first (and only) record of the displacement vector file, UGV, is read into open core, and then the ECPTDS and ECPT files are opened and positioned correctly.
6. The ECPT data block is read record by record, and the ECPTDS scratch data block is generated for use by subroutine DS1A. The procedure for each record is as follows. The pivot point is read. For each element in the current ECPT record, it is determined if the element type is admissible to the ECPTDS data block. If it is not admissible, the next element entry is read; if it is admissible, a test is made to determine if the element type is a TRIA1, QUAD1 or conical shell element. If it is a TRIA1, QUAD1 or a conical shell element and if this element has zero membrane thickness (which implies the element has bending properties only) and since differential stiffness is not defined for plate elements, then the next element entry is read from the current ECPT record. When the first element of the ECPT record which belongs to the differential stiffness set is encountered, the pivot point is written on the ECPTDS data block. Note that if the element is a BAR, the ECPT entry is rearranged to conform with the ECPT entry for the BEAM so that the DBAR subroutine may be called in subroutine DS1A. The element type is written on the ECPTDS record. Then an element deformation number, an average element loading temperature (grid point temperatures for the conical shell and isoparametric solid elements), and the displacement vector components are appended in core to the ECPT entry for the element. Finally this appended ECPT entry is written on ECPTDS. It should be noted that the 2nd through the $(n+1)^{st}$ words (n being the number of grid points of the element) of every ECPT entry, since they are scalar index numbers, are direct pointers into the displacement vector, UGV. Note further that for elements for which differential stiffness is defined (save for the BAR, the BEAM, the QUADTS, the TRIATS and the conical shell element) only the three translational components of the displacement vector at each grid point are appended. For the BAR, BEAM, QUADTS and

MODULE FUNCTIONAL DESCRIPTIONS

TRIATS all six displacement components at each grid point are appended. For the special case of the conical shell element, only the six displacement components at each grid point associated with the zeroth harmonic are appended.

When an end-of-record is sensed for an ECPT record, an indicator is interrogated to determine whether or not any of the elements in the record were written on the ECPTDS data block. If there weren't any, a -1 is written on ECPTDS with an end-of-record mark; if there was at least one element, an end-of-record is written on ECPTDS. In either case, after this step has been done, the next ECPT record is processed identically. When an end-of-file is sensed on the ECPT, the files for ECPT and ECPTDS are closed, and the routine returns to DSMG1.

The module driver, DSMG1, tests the argument of DS1. If it is equal to zero, this implies that no element of the ECPT was a member of the set of elements for which differential stiffness is defined, and hence a fatal error exit occurs. If the argument is greater than zero, subroutine DS1A is called.

In the second phase of the module, subroutine DS1A processes the scratch file ECPTDS to produce the differential stiffness matrix, $[K_{gg}^d]$. The logic of this processing is very similar to that used in subroutine SMA1A (of module SMA1, the stiffness matrix generation routine). See the Module Functional Description for SMA1 (section 4.27) for details.

4.49.8 Subroutines

All subroutines defined below except DS1 are necessary to accomplish the second phase of the module, which is the processing of the ECPTDS scratch file in order to generate $[K_{gg}^d]$. The auxiliary routines PRETRD, PREMAT, GMMATD and INVERD are used similarly to module SMA1.

The data (an ECPTDS entry) input to an element differential stiffness matrix generation routine (e.g., DRØD) are communicated to the routine via /DS1AET/, which fact is not expressly stated in the subroutine descriptions given below.

4.49.8.1 Subroutine Name: DS1

1. Entry Point: DS1
2. Purpose: See discussion above.
3. Calling Sequence: CALL DS1 (IARG)

IARG - Initially set to zero in subroutine DS1. This variable is set to 1 in DS1 every time an element in the ECPT is encountered which is in the set of elements for which differential stiffness is defined. If IARG is still zero upon DS1's return to DSMG1, DSMG1 will terminate the job by calling MESSAGE and PEXIT.

4.49.8.2 Subroutine Name: DS1A

1. Entry Point: DS1A
2. Purpose: See discussion above.
3. Calling Sequence: CALL DS1A

4.49.8.3 Subroutine Name: DS1B

1. Entry Point: DS1B
2. Purpose: This routine, called by the module's element matrix generation routines such as DRØD, DBEAM, etc., adds a double precision 6 by 6 element differential stiffness matrix to the "submatrix" corresponding to the current pivot point. This same function is performed in module SMA1 by subroutine SMA1B, in SMA2 by SMA2B and in PLA4 by PLA4B.

MODULE FUNCTIONAL DESCRIPTIONS

3. Calling Sequence: CALL DS1B (KE,J)

KE - Row-stored double precision 6 by 6 matrix to be added to the "submatrix" corresponding to the current pivot point. Double precision; input.

J - The column index of the $[K_{gg}^d]$ matrix which corresponds to the first column of the KE matrix. Integer; input.

4.49.8.4 Block Data Subprogram Name: DS1ABD

1. Entry Point: DS1ABD

2. Purpose: This block data program sets 1) GINØ file numbers; 2) GINØ OPEN, WRITE and CLØSE option parameters; 3) differential stiffness element routine (e.g., DRØD, DBAR) overlay parameters; and 4) an array which defines the number of words to be read for each element from the ECPTDS scratch file. The common block /DS1AAA/ is used for the second phase of the module.

4.49.8.5 Subroutine Name: DRØD

1. Entry Point: DRØD

2. Purpose: To generate the element differential stiffness matrix for a RØD, CØNRØD or TUBE element.

3. Calling Sequence: CALL DRØD

4.49.8.6 Subroutine Name: DBAR

1. Entry Point: DBAR

2. Purpose: To generate the element differential stiffness matrix for a BAR or BEAM element.

3. Calling Sequence: CALL DBAR

4.49.8.7 Subroutine Name: DSHEAR

1. Entry Point: DSHEAR

2. Purpose: To generate the element differential stiffness matrix for a SHEAR (panel) element.

FUNCTIONAL MODULE DSMG1 (DIFFERENTIAL STIFFNESS MATRIX GENERATOR PHASE 1)

3. Calling Sequence: CALL DSHEAR

4.49.8.8 Subroutine Name: DTRMEM

1. Entry Point: DTRMEM

2. Purpose: To generate the element differential stiffness matrix for a TRMEM element.

3. Calling Sequence: CALL DTRMEM (IARG)

$$IARG = \begin{cases} 0 = \text{Called from DSMG1} \\ 1 = \text{Called from DQDMEM} \\ 2 = \text{Called from DTRIA} \\ 3 = \text{Called from DQUAD} \end{cases}$$

4.49.8.9 Subroutine Name: DQDMEM

1. Entry Point: DQDMEM

2. Purpose: To generate the element differential stiffness matrix for a QDMEM element.

3. Calling Sequence: CALL DQDMEM

4.49.8.10 Subroutine Name: DCONE

1. Entry Point: DCONE

2. Purpose: To generate the element differential stiffness matrix for a conical shell element.

3. Calling Sequence: CALL DCONE

4.49.8.11 Subroutine Name: DTRIA (IARG)

1. Entry Point: DTRIA

2. Purpose: To generate the element differential stiffness matrix for a TRIA1 and TRIA2 element.

3. Calling Sequence: CALL DTRIA (IARG)

$$IARG = \begin{cases} 1 = \text{TRIA1 element} \\ 2 = \text{TRIA2 element} \end{cases}$$

4.49.8.12 Subroutine Name: DQUAD (IARG)

1. Entry Point: DQUAD
 2. Purpose: To generate the element differential stiffness matrix for a QUAD1 and QUAD2 element.
 3. Calling Sequence: CALL DQUAD (IARG)
- IARG = $\begin{cases} 1 = \text{QUAD1 element} \\ 2 = \text{QUAD2 element} \end{cases}$

4.49.8.13 Subroutine Name DTRBSC (IARG, NPIVOT)

1. Entry Point: DTRBSC
 2. Purpose: Used by subroutines DTRIA and DQUAD to generate differential stiffness matrices for elementary triangles.
 3. Calling Sequence: CALL DTRBSC (IARG, NPIVOT)
- IARG = $\begin{cases} 1 = \text{Called from DTRIA} \\ 2 = \text{Called from DQUAD} \end{cases}$
- NPIVOT = 0,1,2, or 3 = Point on triangle used as pivot point. (0 = no pivot)

4.49.8.14 Subroutine Name: DQUADS

1. Entry Point: DQUADS
2. Purpose: To generate the element differential stiffness matrix for the QUADTS element.
3. Calling Sequence: CALL DQUADS

4.49.8.15 Subroutine Name: DTRIAS

1. Entry Point: DTRIAS
2. Purpose: To generate the element differential stiffness matrix for the TRIATS element.
3. Calling Sequence: CALL DTRIAS

4.49.8.16 Subroutine Name: DIHEX

1. Entry Point: DIHEX

MODULE FUNCTIONAL DESCRIPTIONS

2. Purpose: To generate the element differential stiffness matrix for the isoparametric solid elements.

3. Calling Sequence: CALL DIHEX (TYPE)

$$\text{TYPE} = \left\{ \begin{array}{l} 1 - \text{IHEX1} \\ 2 - \text{IHEX2} \\ 3 - \text{IHEX3} \end{array} \right\} - \text{integer-input}$$

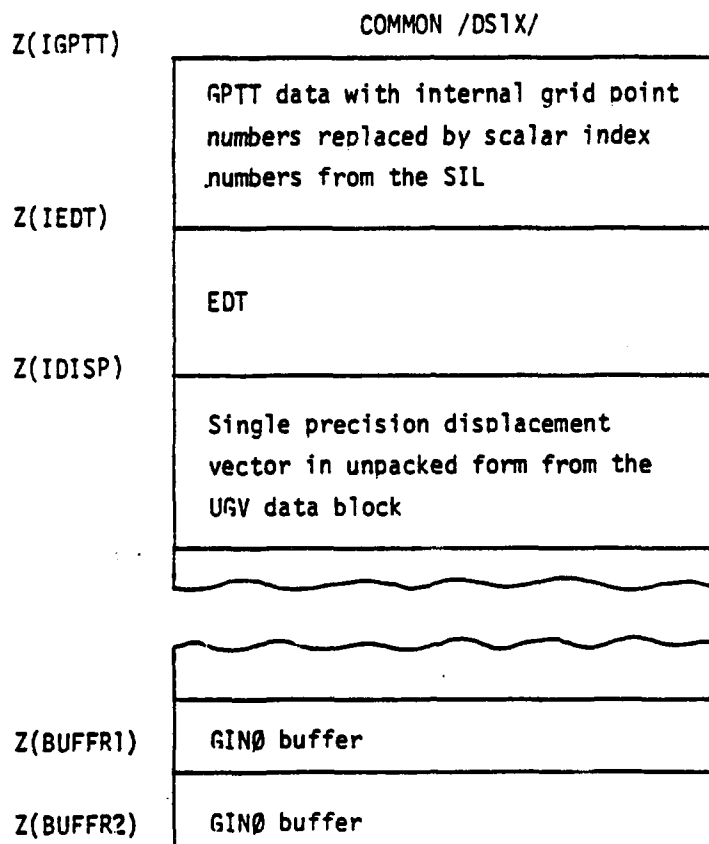
4.49.8.17 Subroutine Name: IHEXSD

See section 4.27.8.54.

4.49.9 Design Requirements

4.49.9.1 Open Core Design

During phase one of the module's operation, the maximum core storage requirements are given in the following diagram, in which open core, the Z array, is defined at /DS1X/.



During phase two of the module's operation, the open core design is the same as that for module SMA1, except that only 3 GINØ buffers are needed instead of 5, one for ECPTDS, one for GPCT and one for KDGG. Open core for phase two is defined at /DS1AXX/.

4.49.9.2 Common Storage Requirements

During phase one, there are no special common storage requirements.

During phase two, /DS1AAA/ takes the place of /SMA1IØ/, /SMA1BK/ and /SMA1CL/; /DS1AET/ corresponds to /SMA1ET/; and /DS1ADP/ corresponds to /SMA1DP/. See the common storage requirements section of the SMA1 Module Functional Description for details on /SMA1IØ/, /SMA1BK/, /SMA1CL/, /SMA1ET/ and /SMA1DP/ (section 4.27.9.3).

4.49.9.3 Arithmetic Considerations

All floating point arithmetic operations are carried out in double precision. $[K_{gg}^d]$ is a real double precision symmetric matrix.

4.49.10 Diagnostic Messages

During phase one the fatal messages 2029 and 2030 can be called if the GPTT is not in the proper format. If a BAR element is encountered whose coupled bending inertia property, I_{12} , is non-zero, then it is set to zero and the user is warned of this fact with message 2111. User messages 2081 and 2083 are fatal messages indicating a null differential stiffness matrix and a null displacement vector respectively.

For phase two diagnostics, the reader is referred to the diagnostic message section of the Module Functional Description for SMA1.

4.50 FUNCTIONAL MODULE SMP2 (STRUCTURAL MATRIX PARTITIONER - PHASE 2)

4.50.1 Entry Point: SMP2

4.50.2 Purpose

To perform the following matrix operations:

$$[K_{ff}^d] \Rightarrow \begin{bmatrix} \bar{K}_{aa}^d & K_{ao}^d \\ (K_{ao}^d)^T & K_{oo}^d \end{bmatrix}, \quad (1)$$

$$[K_{aa}^d] = [\bar{K}_{aa}^d] + [K_{ao}^d] [G_o] + ([K_{ao}^d] [G_o])^T + [G_o]^T [K_{oo}^d] [G_o]. \quad (2)$$

4.50.3 DMAP Calling Sequence

SMP2 {HUSET}, {HGO}, {HRFF}, {HRAA}
 {USET}, {GO}, {HBFF}, {HBAA} / \$
 {KDFF}, {KDAA}

4.50.4 Input Data Blocks

HUSET }
 USET } - Displacement set definitions table.

GØ - Structural matrix partitioning transformation matrix.

HGØ - Heat matrix partitioning transformation matrix.

KDFF - Partition of differential stiffness matrix - f set.

HRFF - Partition of radiation matrix - f set.

HBFF - Partition of capacity matrix - f set.

4.50.5 Output Data Blocks

HRAA - Partition of radiation matrix - a set.

KDAA - Partition of differential stiffness matrix - a set.

HBAA - Partition of capacity matrix - a set.

4.50.6 Parameters

None

MODULE FUNCTIONAL DESCRIPTIONS

4.50.7 Method

The matrix $[K_{ff}^d]$ is partitioned as in Equation 1 using USET (UF, UA, UØ) and matrix subroutine ELIM (see section 3.5.22 for details) is called to perform the operations in Equation 2.

4.50.8 Subroutines

SMP2 has no auxiliary subroutine.

4.50.9 Design Requirements

See design requirements for subroutine UPART and its entry point MPART which perform the symmetric partition of Equation 1; also see design requirements for subroutine ELIM. These are in sections 3.5.9 and 3.5.22 respectively.

FUNCTIONAL MODULE DSMG2 (DIFFERENTIAL STIFFNESS MATRIX GENERATOR - PHASE 2)

4.51 FUNCTIONAL MODULE DSMG2 (DIFFERENTIAL STIFFNESS MATRIX GENERATOR - PHASE 2).

4.51.1 Entry Point: DSMG2

4.51.2 Purpose

This module performs the following matrix operations:

$$[K_{\ell\ell}^b] = [K_{aa}] + \beta[K_{aa}^d], \quad (1)$$

$$[K_{fs}^b] = [K_{fs}] + \beta[K_{fs}^d], \quad (2)$$

$$[K_{ss}^b] = [K_{ss}] + \beta[K_{ss}^d], \quad (3)$$

$$\{P_{\ell}^b\} = \beta\{P_{\ell}\}, \quad (4)$$

$$\{P_s^b\} = \beta\{P_s\}, \quad (5)$$

$$\{Y_s^b\} = \beta\{Y_s\}, \quad (6)$$

$$\{u_o^{ob}\} = \beta\{u_o^o\}. \quad (7)$$

The value of β is on a DSFACT bulk data card whose set identification number is specified by the input parameter DSCØSET. The particular value on that card to be used for β on any pass through the DMAP loop in the Static Analysis with Differential Stiffness Rigid Format is controlled by the parameter NDSKIP (see below).

4.51.3 DMAP Calling Sequence

```
DSMG2  MPT,KAA,KDAA,KFS,KDFS,KSS,KDSS,PL,PS,YS,UØØV/KBLL,KBFS,KBSS.PBL,PBS,YBS,UBØØV/
      V,N,NDSKIP/V,N,REPEATD/V,N,DSCØSET  $
```

4.51.4 Input Data Blocks

MPT - Material Property Table.
 KAA - Partition of stiffness matrix - a set.
 KDAA - Partition of differential stiffness matrix - a set.

MODULE FUNCTIONAL DESCRIPTIONS

- KFS - Partition of stiffness matrix after single-point constraints have been removed.
- KDFS - Partition of differential stiffness matrix after single-point constraints have been removed.
- KSS - Partition of stiffness matrix after single-point constraints have been removed - s set.
- KDSS - Partition of differential stiffness matrix after single-point constraints have been removed - s set.
- PL - Partition of the load vector matrix giving static loads on l set.
- PS - Partition of the load vector matrix giving loads in s set.
- YS - Constrained displacement vector - s set.
- U00V - Partition of the displacement vector matrix giving displacements in the o set.

Notes:

1. A fatal error occurs if MPT is purged.
2. KAA and KDAA cannot be purged.
3. KFS and KDFS must be both purged or both non-purged.
4. KSS and KDSS must be both purged or both non-purged.
5. A fatal error occurs if PL is purged.
6. PS, YS and U00V can be purged.

4.51.5 Output Data Blocks

- KBLL - Partition of the stiffness matrix of the first order approximation to large displacements - l set.
- KBFS - Partition of the stiffness matrix of the first order approximation to large displacements.
- KBSS - Partition of the stiffness matrix of the first order approximation to large displacements - s set.
- PBL - Partition of the load vector of the first order approximation to the large displacements - l set.

FUNCTIONAL MODULE DSMG2 (DIFFERENTIAL STIFFNESS MATRIX GENERATOR - PHASE 2)

- PBS - Partition of the load vector of the first order approximation to the large displacements - l set.
- YBS - Partition of the constrained displacement vector of the first order approximation to the large displacement vector - s set.
- UB00V - Partition of the displacement vector of the first order approximation to the large displacement problem - o set.

Notes:

1. KBLL must not be purged or a fatal error exists.
2. KBFS can be purged if and only if both KFS and KDFS are purged.
3. KBSS can be purged if and only if both KSS and KDSS are purged.
4. PL must not be purged or a fatal error exists.
5. PBS, YBS and UB00V can be purged if and only if PS, YS and UB00V respectively are purged.

4.51.6 Parameters

- NDSKIP - Input-integer - must be set to zero before the DMAP loop for Static Analysis with Differential Stiffness is initiated. This parameter is used as a loop counter for the DMAP loop in the Static Analysis with Differential Stiffness Rigid Format. It enables this module to skip the proper number of words on the proper DSFACT card (see definition of the parameter DSC0SET below) to fetch the value of β for the scalar multiplications defined above.
- REPEATD - Output-integer-no default value. This parameter is set to +1 if another pass is to be made through the Static Analysis with Differential Stiffness DMAP loop (and hence through the DSMG2 module). It is set to -1 if the module determines that the current pass through the module will be the final one. This latter condition is met if (1) no differential stiffness coefficient set number was specified by the user in his Case Control Deck (if this condition is true, module DSMG1 sets the parameter DSC0SET (see below) equal to -1 and a value of 1.0 is used for β); or if (2) the current β is the last one on the user specified DSFACT card.
- DSC0SET - Input-integer-no default value. DSC0SET is the differential stiffness coefficient

set number of a DSFACT bulk data card chosen by the user in his Case Control Deck. If no such set was specified by the user, DSCØSET = -1 and this module will set $\beta = 1.0$ and set the REPEATD parameter to -1. If DSFACT > 0, DSFACT is used to search the MPT for the DSFACT bulk data card chosen by the user.

4.51.7 Method

REPEATD is set to one and NDSKIP is incremented by one. If DSCØSET = -1, REPEATD is set to -1 and the three matrix additions in Equations 1, 2 and 3 are carried out using the matrix subroutine SSG2C. It should be noted that if DSCØSET = -1, it is assumed that DMAP equivalences have been made between PL and PBL, PS and PBS, YS and YBS, and UØØV and UBØØV and hence Equations 4 through 7 are not calculated.

If DSCØSET \neq -1, the MPT is searched until a match is found between DSCØSET and a DSFACT bulk data card image on the MPT. If a match is found, the parameter NDSKIP is used to find the correct value of β for this pass through the DMAP loop. If a match is not found, a fatal error occurs. A match having been found, the following operations are performed.

1. REPEATD is set to -1 if it is determined that this β is the last one on the DSFACT card.
2. The four scalar multiplications in Equations 4 through 7 are performed using matrix subroutine SSG2C.
3. The three matrix additions in Equations 1 through 3 are performed using matrix subroutine SSG2C.

4.51.8 Subroutines

DSMG2 has no auxiliary subroutines. A description of SSG2C can be found in section 3.5.11.

4.51.9 Design Requirements

If DSCØSET = -1, it is assumed that DMAP equivalences have been made between the data blocks corresponding to the matrices in Equations 4 through 7.

Two GINØ buffers are needed and open core is defined at /DSMG2X/.

4.51.10 Diagnostic Messages

Fatal error message 2084 may occur. Included with the message is a flag that has the following definitions:

Flag = 3	Unable to open MPT
= 4	Unable to locate DSFACT card
= 5, 7, 9, 11, 13	End of record encountered on MPT
= 6, 8, 10, 12, 14	End of file encountered on MPT

1

2

3

4

5

6

7

8

9

10

FUNCTIONAL MODULE PLA1 (PIECEWISE LINEAR ANALYSIS - PHASE 1)

4.52 FUNCTIONAL MODULE PLA1 (PIECEWISE LINEAR ANALYSIS - PHASE 1)

4.52.1 Entry Point: PLA1

4.52.2 Purpose

PLA1 is a pre-processor for the modules unique to the Piecewise Linear Analysis Rigid Format. PLA1 extracts the linear elements from the ECPT data block (an element is defined to be linear if its modulus of elasticity (E on a MAT1 bulk data card) is not referenced as a stress-strain tabular function defined on a TABLES1 bulk data card. PLA1 extracts the nonlinear element entries from the ECPT data block to form ECPTNL, and separates the linear and nonlinear element entries in the EST to form ESTL and ESTNL. The linear elements are used to generate the $[K_{gg}^{xl}]$ matrix.

4.52.3 DMAP Calling Sequence

```
PLA1    CSTM,MPT,ECPT,GPCT,DIT,CASECC,EST/KGGXL,ECPTNL,ESTL,ESTNL/V,N,KGGLPG/V,N,NPLALIM/  
        V,N,ECPTNLPG/V,N,PLSETNØ/V,N,NØNLSTR/V,N,PLFACT $
```

4.52.4 Input Data Blocks

CSTM	-	Coordinate System Transformation Matrices.
MPT	-	Material Properties Table.
ECPT	-	Element Connection and Properties Table.
GPCT	-	Grid Point Connection Table.
DIT	-	Direct Input Tables.
CASECC	-	Case Control Data Table.
EST	-	Element Summary Table.

Notes: 1. The CSTM may be purged. However, if it is, and some grid point is not in basic coordinates, then a fatal error exists.

2. If an element references a material property and the MPT is purged, a fatal error exists.

3. If any of the data blocks ECPT, GPCT, DIT, CASECC or EST is purged, a fatal error exists.

MODULE FUNCTIONAL DESCRIPTIONS

4.52.5 Output Data Blocks

- KGXGL - Stiffness matrix of linear elements exclusive of general elements - g set.
- ECPTNL - Element Connection and Properties Table for Nonlinear Elements.
- ESTL - Element Summary Table for Linear Elements.
- ESTNL - Element Summary Table for Nonlinear Elements.

Note: None of the output data blocks may be pre-purged.

4.52.6 Parameters

- KGGLPG - Output-integer-no default value. Purge flag for the KGXGL matrix. If all elements are nonlinear, the KGXGL matrix is the zero matrix, and the module sets KGGLPG = -1. If a linear element is found, KGGLPG will be set to +1.
- NPLALIM - Output-integer-no default value. NPLALIM is the number of load increments on the PLFACT card chosen by the user in his Case Control Deck. This parameter controls the number of steps in the DMAP loop of the Piecewise Linear Analysis (PLA) Rigid Format.
- ECPTNLPG - Output-integer-no default value. Purge flag for the ECPTNL data block as well as a fatal error condition flag with respect to the PLA Rigid Format. If the module finds that all elements are linear, ECPTNLPG is set = -1, and, upon completion of the module, a JUMP to an error condition message writer will occur, and then an EXIT will be executed. If at least one element is nonlinear, ECPTNLPG is set to +1, and the flow through DMAP sequence continues.
- PLSETNØ - Output-integer-no default value. PLSETNØ is the set number on a PLFACT card which is chosen by the user in his Case Control Deck. It is used in modules PLA3 and PLA4 to find the proper PLFACT card in the MPT data block.
- NØNLSTR - Output-integer-no default value. NØNLSTR is a flag used to control the calling of the PLA3 module, which outputs stresses, in ØFP (Output File Processor) format, for nonlinear elements. If either (a) the user does not request the output of element stresses, or (b) the user has requested

element stress output for a set of elements all of whose members are linear, then NØNLSTR is set to -1. If there is a stress output request for some nonlinear element, then NØNLSTR is set = +1 and PLA3 will be called each time through the PLA DAMP loop.

PLFACT - Output-complex-no default value. The first load increment factor to be used the first time through the PLA DAMP loop.

4.52.7 Method

The routine is divided into two phases. Phase 1 processes the ECPT data block in a fashion similar to module SMA1 (see Section 4.27). For each pivot point, every element is examined to determine whether it is linear or non linear. This is accomplished by calling subroutine MAT (see Section 3.4.36) with the second word, INFLAG, of the common block /MATIN/ equal to 5. If the element is linear, the proper element stiffness matrix generator routine such as KRØD, KBAR, etc. is called. The element routine will, in turn, call subroutine SMA1B to add its contribution to the 6 (or fewer, if the pivot point is a scalar point or there is not enough core storage available-see Module Functional Description for SMA1, Section 4.27-) rows of the KGGXL matrix currently being generated. If the MAT routine determines that the element is nonlinear, then the ECPT entry for that element, along with words needed subsequently in module PLA4, is appended (see data block description for ECPTNL in Section 2.3.34.4 for details). The number of words appended depends upon element type. This appended ECPT entry is written onto the ECPTNL data block.

Phase 2 of this routine reads the first record of CASECC into core, and the MPT data block is searched to find the set number on a PLFACT bulk data card (if the default is not used) requested by the user in CASECC. If the default value is specified by the user as described for the PLCØ card of Section 2.3 of the User's Manual, a single value of 1.0 will be generated. Parameters PLFACT and PLASETNØ are set. The EST data block is then processed. The logic here is similar to Phase 1. For each element, it is determined if the element is linear or nonlinear. If the element is linear, its EST entry is copied onto the ESTL data block. If the element is non-linear and stress output is requested, the EST entry along with words needed subsequently in module PLA3 are appended. The number of words appended depends upon element type and is not the same number of words appended to the ECPT entry to create the ECPTNL. If the element nonlinear and stress output is not requested, the EST entry for the element is written onto the ESTL data block.

4.52.8 Subroutines

PLA1 has no auxiliary subroutines as such. However, it uses all the structural element routines of module SMA1 to generate $[K_{gg}^{xl}]$, which in turn use the common blocks of SMA1 as well as the "insertion" routine, SMA1B. See Module Functional Description for SMA1, section 4.27.

4.52.9 Design Requirements

For phase 1 of PLA1, the design requirements are the same as those for SMA1. For phase 2, the first record of CASECC must be held in open core.

4.52.10 Diagnostic Messages

For phase 1, see diagnostic messages for module SMA1, section 4.27.10. For phase 2, a user fatal message, 3032, occurs if the PLFACT bulk data card which was chosen by the user in his Case Control Deck could not be found in the MPT.

4.53 FUNCTIONAL MODULE PLA2 (PIECEWISE LINEAR ANALYSIS - PHASE 2)

4.53.1 Entry Point: PLA2

4.53.2 Purpose

To add the incremental displacement vector, the incremental load vector, and the incremental vector of single-point forces of constraint for the current pass through the Piecewise Linear Analysis Rigid Format DMAP loop to the current running sum of these vectors:

$$\{u_{g_{i+1}}\} = \{u_{g_i}\} + \{\Delta u_{g_i}\} , \quad (1)$$

$$\{P_{g_{i+1}}\} = \{P_{g_i}\} + \{\Delta P_{g_i}\} , \quad (2)$$

$$\{q_{g_{i+1}}\} = \{q_{g_i}\} + \{\Delta q_{g_i}\} . \quad (3)$$

4.53.3 DMAP Calling Sequence

PLA2 DELTAUGV,DELTAPG,DELTAQG/UGV1,PGV1,QG1/V,N,PLACØUNT \$

4.53.4 Input Data Blocks

DELTAUGV - Incremental displacement vector in Piecewise Linear Analysis - g set.

DELTAPG - Incremental load vector in Piecewise Linear Analysis - g set.

DELTAQG - Incremental vector of single-point forces of constraint in Piecewise Linear Analysis - g set.

Note:

1. DELTAUGV and DELTAPG cannot be pre-purged.
2. DELTAQG may be pre-purged.

4.53.5 Output Data Blocks

UGV1 - Matrix of successive sums of incremental displacement vectors - g set.

PGV1 - Matrix of successive sums of incremental load vectors - g set.

QG1 - Matrix of successive sums of incremental vectors of single-point forces of constraint - g set.

Notes:

1. UGV1 and PGV1 cannot be purged.
2. QG1 may be purged if DELTAQG is purged.

4.53.6 Parameters

PLACQUNT - Input and output-integer - this parameter must be set to 1 outside the Piecewise Linear Analysis Rigid Format DMAP loop. This is done using the PARAM module rather than through the Module Properties List (MPL).

4.53.7 Method

If PLACQUNT = 1, that is, this is the first time PLA2 has been called in the Piecewise Linear Analysis Rigid Format DMAP loop, then the DELTAUGV data block is copied onto the UGV1 data block. If PLACQUNT > 1, then PLACQUNT is used as a counter to determine how many records (running sum displacement vectors) to skip on the file containing UGV1 so that the most recently computed running sum displacement vector can be read into open core for the vector addition. Once this vector is read into open core, the incremental displacement vector is read and interpreted using subroutines INTPK and ZNTPKI, and the vector addition given in Equation 1 is carried out element-by-element.

Equations 2 and 3 are computed using the method described in the above paragraph.

4.53.8 Subroutines

PLA2 has no auxiliary subroutines.

4.53.9 Design Requirements

Open core is defined at /PLA2X/.

4.53.10 Diagnostic Messages

User message 2127 or 2128 is output if either DELTAUGV (DELTAPG) or UGV1 (PGV1) is purged.

4.54 FUNCTIONAL MODULE PLA3 (PIECEWISE LINEAR ANALYSIS - PHASE 3)

4.54.1 Entry Point: PLA3

4.54.2 Purpose

To compute element stresses for nonlinear elements (see definition of linear elements in section 4.52.2) for which the user has requested stress output. It also updates the ESTNL data block so that the output data block, ESTNL1, contains up-to-date element stress information.

4.54.3 DMAP Calling Sequence

PLA3 CSTM,MPT,DIT,DELTAUGV,ESTNL,CASECC/ØNLES,ESTNL1/V,N,PLACØUNT/V,N,PLSETNØ \$

4.54.4 Input Data Blocks

CSTM - Coordinate System Transformation Matrices.
 MPT - Material Properties Table.
 DIT - Direct Input Tables.
 DELTAUGV - Current incremental displacement vector.
 ESTNL - Element Summary Table for Nonlinear Elements.
 CASECC - Case Control Data Table.

Notes:

1. CSTM can be purged. However, if some grid point of the model is not in basic coordinates and the CSTM is purged, a fatal error occurs.
2. A fatal error occurs if either MPT, DIT, DELTAUGV, ESTNL or CASECC is purged.

4.54.5 Output Data Blocks

ØNLES - Nonlinear element stresses (to be processed by the Output File Processor).
 ESTNL1 - Element Summary Table for Nonlinear Elements - Updated.

Note: Neither output data block may be purged.

4.54.6 Parameters

PLACØUNT - Input-integer-no default value. This is the Piecewise Linear Analysis (PLA)

Rigid Format DMAP loop counter. It is used in this routine to find the proper loading factors on the PLFACT bulk data card specified by the user (see PLSETNØ below).

PLTSETNØ - Input-integer-no default value. PLSETNØ is the set identification number of some PLFACT bulk data card chosen by the user in his Case Control Deck. It is used to find this PLFACT card in the MPT data block.

4.54.7 Method

The module driver, PLA3, is a short routine whose only function is to call subroutines PLA31 and PLA32 which accomplish phase 1 and phase 2 of the task of the module respectively. Subroutine PLA31 reads the incremental displacement vector into core and appends to each element entry of the ESTNL data block the components of the incremental displacement vector corresponding to the grid points of each element. This merged information is written on the scratch data block ESTNLS, GINØ file number 301. In PLA32, the ESTNLS data block is read, and the proper element routine is called to compute element stresses which are prepared in ØFP (Output File Processor) format. Each element routine also updates incremental stress data. The ESTNL data for each element with the updated stress information (but without the components of the displacement vector) are written on ESTNL1.

In PLA31, for TRMEM and QDMEM elements, only the three translational components of the displacement vector at each grid point of the element are appended to the ESTNL entry. Other elements for which Piecewise Linear Analysis is defined use all six components at each grid point.

In PLA32, the difference quotients γ^* and γ , which are the previous and current (with respect to the DMAP loop in the PLA Rigid Format) load increment ratios, are computed as follows. Let P_1, P_2, P_3, \dots , be the loading factors on a PLFACT bulk data card. Define $P_0 = 0$. Define

$$\alpha_i = P_i - P_{i-1} \quad , \quad (1)$$

for $i \geq 1$. Then, define $\gamma_1^* = 0$, and

$$\gamma_i^* = \frac{\alpha_i}{\alpha_{i-1}} \quad , \quad (2)$$

for $i > 1$, and define

$$\gamma_i = \frac{\alpha_{i+1}}{\alpha_i}, \quad (3)$$

for $i \geq 1$. These difference quotients are stored in /PLA32C/ for communication to the module's element routines so that they can compute the estimated next strain. The details of the element calculation are given in section 4.87. The input parameter PLACØUNT, being the counter for the PLA Rigid Format DMAP loop, controls the computation of γ^* and γ . However, the module's design assumes (1) PLACØUNT is set to one outside the PLA DMAP loop and (2) module PLA2, which increments PLACØUNT by one, will be executed prior to every DMAP call to PLA3. Hence, the proper choice for the subscript i in Equations 1, 2 and 3 is one less than the value of PLACØUNT. The difference PLACØUNT-1 is stored in /PLA32C/ as IPASS.

4.54.8 Subroutines

PLA3 uses, for element routine calculations, the utility routines PRETRS, PREMAT, GMMATS and element drivers. Communication of an appended ESTNL element entry to an element routine during phase 2 of PLA3 is accomplished via /PLA32E/, which is 100 words in length. This fact is not explicitly stated below.

The element drivers PSTRM, PSQDM, PSTRI1, PSTRI2, PSQAD1, and PSQAD2, use a) /PLA3ES/, which is 300 words in length, as a communication link for the element subroutines which they call; and b) /PLA3UV/, which is 25 words in length, as a communication link for displacement vectors between the driver and their subroutines. PLA32 will call the element drivers listed above (plus PSRØD and PSBAR); the other subroutines described below (in sections 4.54.8.11 through 4.54.8.18) are only used (directly or indirectly) by the element drivers.

4.54.8.1 Subroutine Name: PLA31

1. Entry Point: PLA31
2. Purpose: To perform phase 1 of the module's operations as described above.
3. Calling Sequence: CALL PLA31

4.54.8.2 Subroutine Name: PLA32

1. Entry Point: PLA32
2. Purpose: To perform phase 2 of the module's operation as described above.
3. Calling Sequence: CALL PLA32

4.54.8.3 Subroutine Name: PSRØD

1. Entry Point: PSRØD
2. Purpose: To compute element stresses and to update the ESTNL entry for a RØD, CØNRØD or TUBE element. Note that for a TUBE element, the ESTNL entry is rearranged and elementary transformations are performed in PLA32 so that the PSRØD routine may compute element stresses for a TUBE.
3. Calling Sequence: CALL PSRØD

4.54.8.4 Subroutine Name: PSBAR

1. Entry Point: PSBAR
2. Purpose: To compute element stresses and to update the ESTNL entry for a BAR element.
3. Calling Sequence: CALL PSBAR

4.54.8.5 Subroutine Name: PSTRM

1. Entry Point: PSTRM
2. Purpose: To calculate the material properties matrix, arrange the flow of element stress calculations and update the ESTNL entry for the TRMEM element.
3. Calling Sequence: CALL PSTRM

4.54.8.6 Subroutine Name: PSQDM

1. Entry Point: PSQDM
2. Purpose: To calculate the material properties matrix, arrange the flow of element stress calculations and update the ESTNL entry for the QDMEM element.
3. Calling Sequence: CALL PSQDM

4.54.8.7 Subroutine Name: PSTRI1

1. Entry Point: PSTRI1
2. Purpose: To calculate the material properties matrix, arrange the flow of element stress calculations and update the ESTNL entry for the TRIA1 element.
3. Calling Sequence: CALL PSTRI1

4.54.8.8 Subroutine Name: PSTRI2

1. Entry Point: PSTRI2
2. Purpose: To calculate the material properties matrix, arrange the flow of element stress calculations and update the ESTNL entry for the TRIA2 element.
3. Calling Sequence: CALL PSTRI2

4.54.8.9 Subroutine Name: PSQAD1

1. Entry Point: PSQAD1
2. Purpose: To calculate the material properties matrix, arrange the flow of element stress calculations and update the ESTNL entry for the QUAD1 element.
3. Calling Sequence: CALL PSQAD1

4.54.8.10 Subroutine Name: PSQAD2

1. Entry Point: PSQAD2
2. Purpose: To calculate the material properties matrix, arrange the flow of element stress calculations and update the ESTNL entry for the QUAD2 element.
3. Calling Sequence: CALL PSQAD2

4.54.8.11 Subroutine Name: PSTRM1

1. Entry Point: PSTRM1
2. Purpose: To generate element stress matrices for the TRMEM element, and the membrane portion of TRIA1 and TRIA2 elements, and perform subcomputations for the PSQDM1 routine.
3. Calling Sequence: CALL PSTRM1 (NTYPE)

NTYPE $\begin{cases} 0 = \text{TRMEM, TRIA1, or TRIA2} \\ 1 = \text{Subcomputations for the PSQDM1 subroutine} \end{cases}$

4.54.8.12 Subroutine Name: PSQDM1

1. Entry Point: PSQDM1
2. Purpose: To generate element stress matrices for the QDMEM element and the membrane portions of QUAD1 and QUAD2 elements.
3. Calling Sequence: CALL PSQDM1

4.54.8.13 Subroutine Name: PSTQ1

1. Entry Point: PSTQ1
2. Purpose: To generate element stress matrices for the TRIA1, TRIA2, QUAD1, and QUAD2 elements.
3. Calling Sequence: CALL PSTQ1 (NTYPE)

NTYPE $\begin{cases} 1 = \text{TRIA1} \\ 2 = \text{TRIA2} \\ 3 = \text{QUAD1} \\ 4 = \text{QUAD2} \end{cases}$

4.54.8.14 Subroutine Name: PSTRB1

1. Entry Point: PSTRB1
2. Purpose: To generate element stress matrices for subcalculations of basic bending triangles for the plate portion of TRIA1, TRIA2, QUAD1 and QUAD2 elements.
3. Calling Sequence: CALL PSTRB1 (IØPT)

IØPT $\begin{cases} 1 = \text{Subcalculations for PSQPL1} \\ 2 = \text{Subcalculations for PSTPL1} \end{cases}$

4.54.8.15 Subroutine Name: PSTPL1

1. Entry Point: PSTPL1
2. Purpose: To generate the element stress matrices for the plate portion of TRIA1 and

TRIA2 elements.

3. Calling Sequence: CALL PSTPL1

4.54.8.16 Subroutine Name: PSQPL1

1. Entry Point: PSQPL1

2. Purpose: To generate element stress matrices for the QUAD1 and QUAD2 elements.

3. Calling Sequence: PSQPL1

4.54.8.17 Subroutine Name: PSTRQ2

1. Entry Point; PSTRQ2

2. Purpose: To perform final stress computations for TRMEM and QDMEM elements.

3. Calling Sequence: CALL PSTRQ2 (NTYPE)

NTYPE $\begin{cases} 1 = \text{TRMEM element} \\ 2 = \text{QDMEM element} \end{cases}$

4.54.8.18 Subroutine Name: PSTQ2

1. Entry Point: PSTQ2

2. Purpose: To perform final stress computations for the TRIA1, TRIA2, QUAD1, and QUAD2 elements.

3. Calling Sequence: CALL PSTQ2 (NPTS)

NPTS $\begin{cases} 3 = \text{TRIA1 and TRIA2 elements} \\ 4 = \text{QUAD1 and QUAD2 elements} \end{cases}$

4.54.9 Design Requirements

1. The module was designed so that phase 1 and phase 2 can be executed in separate overlay segments.

2. Open core for phase 1 is defined at /PLA31X/ and for phase 2 at /PLA32X/. Open core requirements for both phases are minimal. In phase 1, the single precision incremental displacement vector in unpacked form must be able to be contained in open core. In phase 2, the CSTM and MPT data blocks, tables in the DIT referenced on MATS1 bulk data cards, and

the first record (and only record since a PLA problem allows only one CASECC record) of CASECC must be able to be contained in open core.

3. In addition to the common blocks mentioned above, PLA32 uses /PLA32S/, which is 325 words in length, as scratch storage for the module's element routines, and /SOUT/, which is 30 words in length, as a storage buffer for computed element stresses.

4. One scratch file is used, and all arithmetic operations are performed in single precision.

4.54.10 Diagnostic Messages

During phase 1, the following diagnostic messages may appear. If the incremental displacement vector is null, user fatal error 3005 will be given. Two system fatal "fail-safe" error messages, 2091 and 2092, may be implemented if the ESTNL input data block was incorrectly constructed in PLA1 or was incorrectly updated during the previous execution of the PLA3 module.

During phase 2, error messages 3001, 3002 or 3003 may occur if the proper loading factors P_j cannot be found on the PLFACT bulk data card image in the MPT. If the ECPTDS scratch file is not in the prescribed format, system fatal message 2091 will occur.

If the minimal core storage requirements in either phase 1 or phase 2 are not met, the usual fatal error 3008 will occur.

4.55 FUNCTIONAL MODULE PLA4 (PIECEWISE LINEAR ANALYSIS - PHASE 4)

4.55.1 Entry Point: PLA4

4.55.2 Purpose

To generate the stiffness matrix for nonlinear elements, $[K_{qq}^{nl}]$, and to update the Element Connection and Properties Table for Nonlinear Elements, ECPTNL, so that it contains up-to-date element stress information.

4.55.3 DMAP Calling Sequence

PLA4 CSTM,MPT,ECPTNL,GPCT,DIT,DELTAUGV/KGGNL,ECPTNL1/V,N,PLACOUNT/V,N,PLSETNØ/
 V,N,PLFACT \$

4.55.4 Input Data Blocks

CSTM - Coordinate System Transformation Matrices.
MPT - Material Properties Table.
ECPTNL - Element Connection and Properties Table for Nonlinear Elements.
GPCT - Grid Point Connection Table.
DIT - Direct Input Tables.
DELTAUGV - Current incremental displacement vector.

Notes:

1. CSTM may be purged. However, if some grid point of the model is not in basic coordinates and the CSTM has been purged, a fatal error will occur.
2. A fatal error occurs if either MPT, ECPTNL, GPCT, DIT or DELTAUGV is purged.

4.55.5 Output Data Blocks

KGGNL - Stiffness matrix of nonlinear elements - g set.
ECPTNL1 - Element Connection and Properties Table for Nonlinear Elements - updated.

Note: Neither KGGNL or ECPTNL1 may be purged.

4.55.6 Parameters

- PLACOUNT - Input-integer-no default value. Loop counter for the Piecewise Linear Analysis (PLA) Rigid Format DMAP loop. The module uses this parameter to find the correct loading factors on the PLFACT bulk data card chosen by the user.
- PLSETNO - Input-integer-no default value. Set identification number of a PLFACT bulk data card chosen by the user in his Case Control Deck. The module uses this parameter to search the MPT for this card.
- PLFACT - Output-complex-no default value. The difference of loading factors to be used during the next pass of the PLA Rigid Format DMAP loop.

4.55.7 Method

The module driver PLA4 is a short routine whose only function is to call subroutines PLA41 and PLA42 which accomplish phase 1 and phase 2 of the task of the module respectively. Subroutine PLA41 reads the incremental displacement vector into core and appends to each element entry of the ECPTNL data block the components of the incremental displacement vector corresponding to the grid points of each element. This merged information is written on the scratch data block ECPTS, GINØ file number 301. In PLA42, the ECPTS data block is processed in a fashion similar to the processing of the ECPT data block in module SMA1 (see the Module Functional Description for SMA1, section 4.27).

In PLA41, for all elements except the BAR element, only the three translational components of the displacement vector at each grid point of an element are appended to the ECPTNL element entry. For a BAR element, all six components of the displacement vector at each grid point are appended.

The logic of the processing of the scratch data block, ECPTS, in PLA42 is very similar to that used in subroutine SMA1A (of SMA1, the stiffness matrix generation module - see the Module Functional Description for SMA1, section 4.27). The similarities are not enumerated here, but notable differences are the following.

1. Before PREMAT is called to read into open core the MPT data block and tables from the DIT data block referenced on MATS1 bulk data cards, the MPT is read in subroutine

PLA42 to compute γ^* and γ as in Equations 1, 2 and 3 in section 4.54, and the real part of the output DMAP parameter PLFACT is set to the value of α_{i+1} in Equation 3 in section 4.54. The imaginary part of PLFACT is set to zero. The reason for PLFACT being complex is that it is an input parameter to the DMAP module ADD during the next pass of the PLA Rigid Format DMAP loop, and ADD (see section 4.78) requires its parameters to be complex.

2. When PREMAT is called, the last argument is set negative to signal PREMAT that this is a PLA problem and hence that special processing will be required.

3. Subsequent to the call of an element routine, the element type and the updated ECPT entry are written onto the ECPTNL1 data block.

4.55.8 Subroutines

PLA4 uses PRETRD, PRETRS, PREMAT, INVERS, INVERD, GMMATS, and GMMATD as utility routines. The common block /PLA42E/ is the means of communicating a) the element entry of the ECPTS from PLA42 to an element stiffness matrix generation routine and b) the ECPTS element entry with updated stress information from the element routine back to PLA42 upon completion of element matrix generation. This fact is not explicitly stated in the descriptions of the element routines (e.g., PKRQD) given below.

The element drivers PKTRM, PKQDM, PKTRI1, PKTRI2, PKQAD1, and PKQAD2 use a) /PLA4ES/, which is 300 words in length, and b) /PLA4UV/, which is 25 words in length, as communication links with the subroutines that they call. PLA42 will call the drivers listed above which will use (directly and indirectly) the subroutines described below in sections 4.55.8.12 through 4.55.8.22.

4.55.8.1 Subroutine Name: PLA41

1. Entry Point: PLA41
2. Purpose: See discussion above.
3. Calling Sequence: CALL PLA41

4.55.8.2 Subroutine Name: PLA42

1. Entry Point: PLA42
2. Purpose: See discussion above.

3. Calling Sequence: CALL PLA42

4.55.8.3 Subroutine Name: PLA4B

1. Entry Point: PLA4B

2. Purpose: To add a double precision 6 by 6 element stiffness matrix to the "submatrix" corresponding to the current pivot point. This routine performs the same function as, and is modeled after, subroutine SMA1B of module SMA1.

3. Calling Sequence: CALL PLA4B (KE,J)

KE - Row-stored double precision 6 by 6 matrix to be added to the submatrix in core - input.

J - The column index of the KGGNL matrix which corresponds to first column of the KE matrix - integer - input.

4.55.8.4 Subroutine Name: PKRØD

1. Entry Point: PKRØD

2. Purpose: To generate the element stiffness matrix for a RØD element and to update the ECPTNL element entry for a RØD element.

3. Calling Sequence: CALL PKRØD

4.55.8.5 Subroutine Name: PKBAR

1. Entry Point: PKBAR

2. Purpose: To generate the element stiffness matrix for a BAR element and to update the ECPTNL element entry for a BAR element.

3. Calling Sequence: CALL PKBAR

4.55.8.6 Subroutine Name: PKTRM

1. Entry Point: PKTRM

2. Purpose: To calculate the material properties matrix, update the ECPTNL entry, and arrange the flow of element stiffness calculations for the TRMEM element.

3. Calling Sequence: PKTRM

4.55.8.7 Subroutine Name: PKQDM

1. EEntry Point: PKQDM
2. Purpose: To calculate the material properties matrix, update the ECPTNL entry, and arrange the flow of element stiffness calculations for the QDMEM element.
3. Calling Sequence: CALL PKQDM

4.55.8.8 Subroutine Name: PKTRI1

1. Entry Point: PKTRI1
2. Purpose: To calculate the material properties matrix, update the ECPTNL entry, and arrange the flow of element stiffness calculations for the TRIA1 element.
3. Calling Sequence: CALL PKTRI1

4.55.8.9 Subroutine Name: PKTRI2

1. Entry Point: PKTRI2
2. Purpose: To calculate the material properties matrix, update the ECPTNL entry, and arrange the flow of element stiffness calculations for the TRIA2 element.
3. Calling Sequence: CALL PKTRI2

4.55.8.10 Subroutine Name: PKQAD1

1. Entry Point: PKQAD1
2. Purpose: To calculate the material properties matrix, update the ECPTNL entry, and arrange the flow of element stiffness calculations for the QUAD1 element.
3. Calling Sequence: CALL PKQAD1

4.55.8.11 Subroutine Name: PKQAD2

1. Entry Point: PKQAD2
2. Purpose: To calculate the material properties matrix, update the ECPTNL entry, and arrange the flow of element stiffness calculations for the QUAD2 element.
3. Calling Sequence: CALL PKQAD2

4.55.8.12 Subroutine Name: PKTRM1

1. Entry Point: PKTRM1
2. Purpose: To generate element stress matrices for the TRMEM, TRIA1 and TRIA2 elements, and perform subcomputations for the PKQDM1 routine.
3. Calling Sequence: CALL PKTRM1 (NTYPE)

NTYPE $\left\{ \begin{array}{l} 0 = \text{TRMEM, TRIA1 or TRIA2} \\ 1 = \text{Subcomputations for the PKQDM1 routine} \end{array} \right.$

4.55.8.13 Subroutine Name: PKQDM1

1. Entry Point: PKQDM1
2. Purpose: To generate element stress matrices for the QDMEM, QUAD1 and QUAD2 elements.
3. Calling Sequence: CALL PKQDM1

4.55.8.14 Subroutine Name: PKTQ1

1. Entry Point: PKTQ1
2. Purpose: To generate element stress matrices for the TRIA1, TRIA2, QUAD1, and QUAD2 elements.
3. Calling Sequence: CALL PKTQ1 (NTYPE)

NTYPE $\left\{ \begin{array}{l} 1 = \text{TRIA1} \\ 2 = \text{TRIA2} \\ 3 = \text{QUAD1} \\ 4 = \text{QUAD2} \end{array} \right.$

4.55.8.15 Subroutine Name: PKTRQ2

1. Entry Point: PKTRQ2
2. Purpose: To perform final stress computations for the TRMEM and QDMEM elements.
3. Calling Sequence: CALL PKTRQ2 (NTYPE)

NTYPE $\left\{ \begin{array}{l} 1 = \text{TRMEM element} \\ 2 = \text{QDMEM element} \end{array} \right.$

4.55.8.16 Subroutine Name: PKTQ2

1. Entry Point: PKTQ2
2. Purpose: To perform final stress computations for the TRIA1, TRIA2, QUAD1, and QUAD2 elements.
3. Calling Sequence: CALL PKTQ2 (NPTS)

$$\text{NPTS} \begin{cases} 3 = \text{TRIA1 or TRIA2 elements} \\ 4 = \text{QUAD1 or QUAD2 elements} \end{cases}$$

4.55.8.17 Subroutine Name: PKTRMS

1. Entry Point: PKTRMS
2. Purpose: To generate the element stiffness matrix for the TRMEM element and sub-computations for the PKQDMS routine.
3. Calling Sequence: CALL PKTRMS (NTYPE)

$$\text{NTYPE} \begin{cases} 0 = \text{TRMEM} \\ 1 = \text{Sub-computations for PKQDMS} \end{cases}$$

4.55.8.18 Subroutine Name: PKQDMS

1. Entry Point: PKQDMS
2. Purpose: To generate the element stiffness matrix for the QDMEM element.
3. Calling Sequence: CALL PKQDMS

4.55.8.19 Subroutine Name: PKTRQD

1. Entry Point: PKTRQD
2. Purpose: To generate the element stiffness matrix for the TRIA1, TRIA2, QUAD1, or QUAD2 elements.
3. Calling Sequence: CALL PKTRQD (NTYPE)

NTYPE { 1 = TRIA1
2 = TRIA2
3 = QUAD1
4 = QUAD2

4.55.8.20 Subroutine Name: PKTRBS

1. Entry Point: PKTRBS
2. Purpose: To generate the element stiffness matrix subcalculations for the PKTRPL and PKQDPL routines.
3. Calling Sequence: CALL PKTRBS (IØPT)

IØPT { 1 = Subcomputations for PKQDPL
2 = Subcomputations for PKTRPL

4.55.8.21 Subroutine Name: PKTRPL

1. Entry Point: PKTRPL
2. Purpose: To generate the element stiffness matrix for the TRIA1 and TRIA2 elements.
3. Calling Sequence: CALL PKTRPL

4.55.8.22 Subroutine Name: PKQDPL

1. Entry Point: PKQDPL
2. Purpose: To generate the element stiffness matrix for the QUAD1 and QUAD2 elements.
3. Calling Sequence: CALL PKQDPL

4.55.9 Design Requirements

The module was designed so that phase 1 and phase 2 can be executed in separate overlay segments.

Open core for phase 1 is defined at /PLA41X/ and for phase 2 at /PLA42X/. In phase 1 the single precision incremental displacement vector in unpacked form must be able to be contained in core. In phase 2, the open core requirements are the same as those for module SMA1 (see section 4.27.9.1) except that only four GINØ buffers are required during the principal loop of phase 2,

which processes the ECPTS and GPCT in a complementary manner. One GINØ buffer is defined for each of KGGNL, ECPTNL1, ECPTS and GPCT.

In addition to /PLA42E/, which is 100 words in length, subroutine PLA42 uses the following common blocks: a) /PLA42D/, which is 300 double precision words in length, and is used as a scratch storage for the module's element routines; b) /PLA425/, which is 325 single precision words in length, and is used as scratch storage for the module element routines; and c) /PLA42C/, which is a communication region for phase 2 of the task of the module. /PLA42C/ is defined as follows: CØMMØN/PLA42C/NPVT,GAMMA,GAMMAS,IPASS,ICSTM,NCSTM,IGPCT,NGPCT,IPØINT,NPØINT,I6X6K,N6X6K,CSTM,MPT,ECPTS,GPCT,DIT,KGGNL,ECPTØ,INRW,ØUTRW,EØR,NEØR,CLSRW,JMAX,FRØWIC,LØWIC,NRØWSC,NLINKS,NWØRDS(40),IØVRLY(40),LINK(40),NØGØ

- | | | |
|--|---|--|
| GAMMA,GAMMAS | - | The load increment ratios as defined in Equations 2 and 3 in section 4.54. |
| IPASS | - | Number of the current pass through the PLA DMAP loop. |
| NPVT,ICSTM,NCSTM,IGPCT,
NGPCT,IPØINT,NPØINT,
I6X6K,N6X6K | } | - As defined in section 4.27.9. |
| CSTM,MPT,ECTPS,GPCT,
DIT,KGGNL | } | - GINØ file numbers for their corresponding data blocks. |
| ECPTØ | - | GINØ file number for the ECTPNL1 data block. |
| INRW,ØUTRW,...,
IØVRLY(40),LINK(40),NØGØ | } | - As defined in section 4.27.9. |

The variables a) corresponding to GINØ file numbers, b) GINØ parameter options (e.g., INRW, ØUTRW), and c) NLINKS, IØVRLY, and NWØRDS, and NØGØ are set in the block data subprogram PLA4BD.

One scratch file is used, and all operations associated with stiffness matrix calculations are performed in double precision.

4.55.10 Diagnostic Messages

During phase 1, if the incremental displacement vector is null, user fatal error 2083 will occur.

During phase 2, error messages 3001, 3002, or 3003 may occur if the proper loading factors cannot be found on the PLFACT bulk data card image in the MPT. Other diagnostic messages for phase 2 are the same as those for module SMA1 (see section 4.27.10).

FUNCTIONAL MODULE CASE (SIMPLIFY CASE CONTROL)

4.56 FUNCTIONAL MODULE CASE (SIMPLIFY CASE CONTROL)

4.56.1 Entry Point: CASE

4.56.2 Purpose

To remove looping considerations from later dynamics modules.

4.56.3 DMAP Calling Sequence

CASE CASECC,PSDL/CASEXX/C,N,APPROACH/V,N,REPEAT/V,N,LOOP \$

4.56.4 Input Data Blocks

CASECC - Case Control Data Table.

PSDL - Power Spectral Density List.

Note: PSDL is used only if APPROACH = FREQRESP and Random Analysis is selected in CASECC.

4.56.5 Output Data Blocks

CASEXX - Case Control data table for dynamics problems.

Note: CASEXX cannot be purged.

4.56.6 Parameters

APPROACH - Input-BCD-no default. Defines the approach to be used for looping criteria.

<u>BCD Value</u>	<u>LOOP</u>
STATICS	NONE
REIGEN	NONE
DSO	NONE
DS1	NONE
FREQRESP	DIRECT INPUT MATRICES OR TRANSFER FUNCTIONS
TRANRESP	LOADS
BLK0	NONE
BLK1	NONE
CEIGEN	DIRECT INPUT MATRICES OR TRANSFER FUNCTIONS

MODULE FUNCTIONAL DESCRIPTIONS

Γ

<u>BCD Value</u>	<u>LOOP</u>
PLA	NONE

REPEAT - Input and output-integer-set equal to zero outside of the DMAP loop by the PARAM module. -1 if no additional loops; + loop count if loops.

LOOP - Output-integer-default = -1. -1 if this is not a looping problem, 0 if this is a looping problem.

4.56.7 Method

The method of operation depends upon the input parameter APPROACH.

4.56.7.1 Transient Response

If APPROACH = TRANRESP, CASECC is skipped over REPEAT records. If REPEAT = 0, REPEAT is set to 1. One record of CASECC is read and copied onto CASEXX. An attempt is made to read another record. If no more records exist, REPEAT is set to -1. Also, if this is the first entry to CASE (i.e., REPEAT = 1), LOOP is set to -1. If additional records exist, REPEAT and LOOP are set to 1.

4.56.7.2 Complex Eigenvalue Analysis

If APPROACH = CEIGEN, REPEAT records are skipped in CASECC. If REPEAT = 0, REPEAT is set to 1. One record of CASECC is read and copied onto CASEXX. The names of the Direct Input Matrices and Transfer Functions sets are saved. An attempt is made to read another record. If no more exist, REPEAT is set to -1. Also if this is the first entry (i.e., REPEAT = 1) LOOP is set to -1. If additional records exist, their Direct Input Matrices and Transfer Functions sets are compared to those saved. If they all agree, this record is copied onto CASEXX and the process is repeated. If they do not agree, REPEAT is incremented by 1, LOOP is set to 1, and CASE returns.

4.56.7.3 Frequency Response

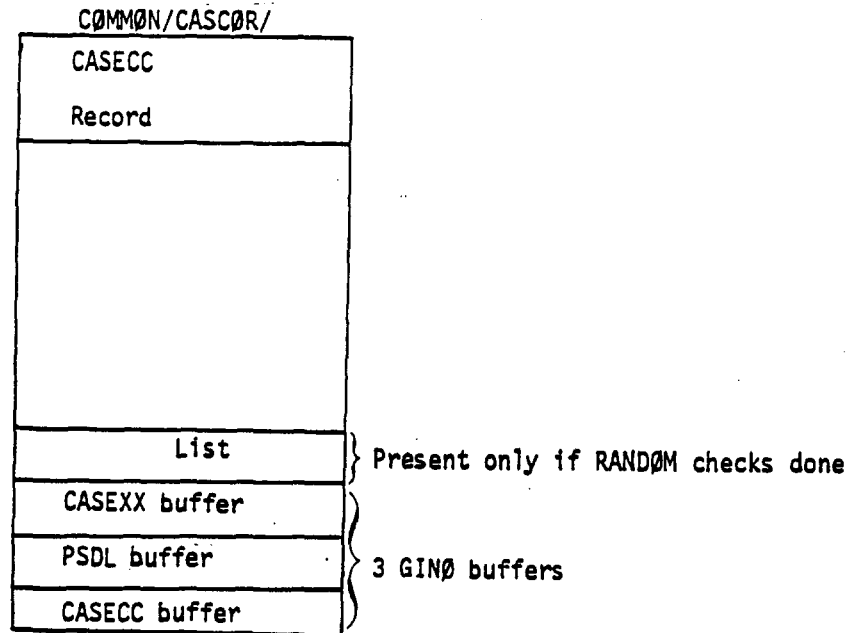
If APPROACH = FREQRESP, the method used is the same as Complex Eigenvalue Analysis except a test is also made for frequency set selection changes. In addition, if RANDPS cards are selected, the selected set is read from PSDL and the unique subcase "id's" referenced are stored. Each subcase id copied onto CASEXX is compared to this list, and the entry is marked as found. If at the completion of CASE unmarked entries exist, the routine terminates with message 3033.

4.56.8. Subroutines

No auxiliary subroutines are used by CASE.

4.56.9 Design Requirements

Open core is defined at /CASCØR/.



4.56.10 Diagnostic Messages

If a case control record cannot be held in core, CASE will issue error message 3008.
 Message 3033 may be issued by CASE as outlined above.

FUNCTIONAL MODULE MTRXIN (MATRIX INPUT)

4.57 FUNCTIONAL MODULE MTRXIN (MATRIX INPUT)

4.57.1 Entry Point: MTRXIN

4.57.2 Purpose

MTRXIN has two purposes: (1) to provide a capability for direct input matrices as may occur in control systems in the dynamics Rigid Formats and, (2) to provide the DMAP user a capability of converting matrices input on DMIG bulk data cards to NASTRAN matrix format.

4.57.3 DMAP Calling Sequences

1. Dynamics Rigid Formats:

```
MTRXIN      { CASEXX } { BDP00L } , EQDYN, TFP00L / { K2DPP } { M2DPP } { B2PP } / V, N, LUSETD / V, N, N0MAT1 /  
            { CASECC } { MATP00L } , { K2PP } { M2PP } { HB2PP } / V, N, LUSETD / V, N, N0MAT1 /  
            V, N, N0MAT2 / V, N, N0MAT3 $
```

2. DMAP Approach:

```
MTRXIN,      , MATP00L, EQEXIN, , / NAME1, NAME2, NAME3 / V, N, LUSETD / V, N, N0MAT1 / V, N, N0MAT2 / V, N, N0MAT3 $
```

4.57.4 Input Data Blocks

CASECC - Case Control.

CASEXX - Case Control data table for dynamics problems.

MATP00L - Data block containing matrices input on DMIG bulk data cards.

BDP00L - Hydroelastic boundary matrix tables.

EQDYN - Equivalence between external numbers and internal numbers, dynamics.

TFP00L - Transfer Function Pool.

EQEXIN - Equivalence between external numbers and internal numbers.

Notes:

1. If CASECC is purged, the second purpose is assumed by MTRXIN.
2. EQDYN, EQEXIN, SIL and SILD may not be purged.

4.57.5 Output Data Blocks

K2DPP - Direct input differential stiffness matrix - p set.

K2PP } - Direct input mass matrix - p set.
HK2PP }

M2DPP - Direct input differential mass matrix - p set.

M2PP - Direct input mass matrix - p set.

B2PP } - Direct input damping matrix - p set.
HB2PP }

NAME1 } - The same names that appear on the DMIG cards, i.e., the DMIG matrix called
NAME2 } - NAME1 will be output on data block NAME1.
NAME3 }

Note: Any output data block may be purged.

4.57.6 Parameters

LUSET - Input-integer-no default. Degrees of freedom in the g set. Used with EQEXIN and SIL.

LUSETD - Input-integer-no default. Degrees of freedom in the p set. Used with EQDYN and SILD.

NØMATi - Output-integer-no default. +1 if the ith output data block is generated, -1 otherwise.

4.57.7 DMAP Example

Assume the bulk data contain two DMIG matrices named M1 and M2 which reference grid and/or scalar points only. The following set of DMAP instructions will generate these two matrices in NASTRAN matrix format, multiply them together and print the result.

```
BEGIN
GP1      GEOM1,GEØM2/GPL,EQEXIN,GPDT,CSTM,BGPDT,SIL/V,N,LUSET/C,N,Ø/C,N,Ø $
SAVE     LUSET $
MTRXIN,  ,MATPØØL,EQEXIN,  ,/M1,M2,/V,N,LUSET/V,N,NØM1/V,N,NØM2/C,N,Ø $
SAVE     NØM1,NØM2 $
CØND     EXIT,NØM1 $
MPYAD    M1,M2,/PRØDUCT/C,N,Ø/C,N,1/C,N,1 $
MATPRN   PRODUCT,,,,// $
LABEL    EXIT $
END
```

4.57.8 Method

The first logical record in the Case Control data block is read into core, and the names of the requested DMIG matrices are fetched. If the Case Control data block is purged, FNAME is called to determine the names of the DMIG matrices from the names of the output data blocks. If the Case Control record was read, the transfer function set selection is fetched. If transfer matrices are requested, the TFP00L data block is opened, and the file is positioned to the requested set. Each transfer function matrix for which a corresponding direct input matrix exists is written on a scratch file. If no direct input matrix exists corresponding to a transfer function matrix, the transfer function matrix is written directly on the appropriate output data block. The transfer function matrices are written in NASTRAN matrix format by decoding the row and column numbers and calling BLDPK.

Upon completion of the writing of the transfer function matrices (if any), the second record of EQEXIN or EQDYN is read into core. The second word of each entry is converted into a scalar index number by dividing by 10. The MATP00L data block is opened. The following processing occurs:

1. The header information for the DMIG matrix is read. If an end-of-file is encountered, step (5) is executed. If the matrix is not requested, the remainder of the record is skipped and step (1) is repeated. Otherwise, step (2) is executed.
2. Each term in the matrix is read. The grid identification and component code are converted to a scalar index value by performing a binary search in EQEXIN or EQDYN in core. The scalar index forms a row position of the matrix. The row and column number (packed in one word) and the value for the term are stored in core. If core storage is exceeded, the terms are written on a scratch file.
3. When all terms have been read, converted and stored, the matrix is sorted by SORT. The matrix is now written in NASTRAN format by BLDPK.
4. If a transfer function is to be added to the DMIG matrix, the ADD routine is called to accomplish the matrix addition.
5. A test is made to determine if all requested matrices have been processed. If not, an error message is queued, and PEXIT is called. Otherwise, the module makes a normal exit.

4.57.9 Design Requirements

4.57.9.1 Allocation of Core Storage

Storage is required to hold the EQDYN or EQEXIN table (2 words per point in the problem) plus five GINØ buffers. Complete spill logic is provided for processing the DMIG matrices.

4.57.9.2 Environment

The module MTRXIN consists of one subroutine, MTRXIN. Calls are made to the utility routine SØRT and matrix operation ADD. Open core is defined by /MTRXXX/. Seven scratch files are used.

4.57.10 Diagnostic Messages

The following messages may be issued by MTRXIN:

2065, 2070, 2074.

4.58 FUNCTIONAL MODULE GKAD (GENERAL K ASSEMBLER DIRECT)

4.58.1 Entry Point: GKAD

4.58.2 Purpose

To assemble the dynamic stiffness, damping and mass matrices.

4.58.3 DMAP Calling Sequence

GKAD {HUSED},GM,{HGØ},{HKA}, {HBAA},{HRAA}, K4AA,{HK2PP}, M2PP,{HB2PP}/
 {USED}, {GØ}, {KAA}, {BAA}, {MAA},
 {HKDD},{HBDD},{HRDD},GMD,{HGØD},{HK2DD},{HM2DD},{HB2DD},
 {KDD},{BDD},{MDD},

4.58.4 Input Data Blocks

- HUSED - Displacement set definitions table dynamics.
- USED
- GM - Multipoint constraint transformation matrix - m set.
- GØ - Structural matrix partitioning transformation matrix.
- KAA - Partition of stiffness matrix - a set.
- BAA - Partition of damping matrix - a set.
- MAA - Partition of mass matrix - a set.
- K4AA - Partition of structural damping matrix - a set.
- K2PP - Direct input stiffness matrix - p set.
- M2PP - Direct input mass matrix - p set.
- B2PP - Direct input damping matrix - p set.
- HGØ - Heat matrix partitioning transformation matrix.
- HKA - Partition of the conductivity matrix - a set.
- HBAA - Partition of the capacity matrix - a set.
- HRAA - Partition of the radiation matrix - a set.
- HK2PP - Direct input conductivity matrix - p set.
- HB2PP - Direct input capacity matrix - p set.

MODULE FUNCTIONAL DESCRIPTIONS

- Notes:
1. USETD cannot be purged.
 2. GM cannot be purged if $MPCF1 \geq 0$.
 3. GØ cannot be purged if $ØMIT \geq 0$.
 4. KAA cannot be purged if $KDEKA \geq 0$.
 5. BAA cannot be purged if $NØBGG \geq 0$.
 6. MAA may be purged.
 7. K4AA cannot be purged if $NØK4GG \geq 0$.
 8. K2PP cannot be purged if $NØK2PP \geq 0$.
 9. M2PP cannot be purged if $NØM2PP \geq 0$.
 10. B2PP cannot be purged if $NØB2PP \geq 0$.

4.58.5 Output Data Blocks

- KDD - Dynamic stiffness matrix - d set.
- BDD - Dynamic damping matrix - d set.
- MDD - Dynamic mass matrix - d set.
- GMD - Multipoint constraint transformation matrix - dynamics.
- HGØD
GØD - Omitted coordinate transformation matrix - dynamics.
- K2DD - Direct input stiffness matrix - d set.
- M2DD - Direct input mass matrix - d set.
- B2DD - Direct input damping matrix - d set.
- HKDD - Dynamic conductivity matrix - d set.
- HBDD - Dynamic capacity matrix - d set.
- HRDD - Dynamic radiation matrix - d set.
- HK2DD - Direct input conductivity matrix - d set.
- HM2DD - Direct input radiation matrix - d set.
- HB2DD - Direct input capacity matrix - d set.

- Notes:
1. GMD cannot be purged if $MPCF \geq 0$.
 2. GØD cannot be purged if $ØMIT \geq 0$.
 3. K2DD cannot be purged if $NØK2PP \geq 0$.
 4. M2DD cannot be purged if $NØM2PP \geq 0$.
 5. B2DD cannot be purged if $NØB2PP \geq 0$.

FUNCTIONAL MODULE GKAD (GENERAL K ASSEMBLER DIRECT)

4.58.6 Parameters

- TYPE - Input-BCD-no default. If TYPE = TRANSIENT the transient equations are used; otherwise the frequency response equations are used.
- APP - Input-BCD-no default. If APP = FØRCE the p set = d set; otherwise p's are reduced to d's by removing m's, s's, and o's.
- FØRM - Input-BCD-no default. If FØRM = MØDAL, KDD and BDD are not computed. MDD is not computed unless MØDACC \geq 0.
- G - Input-real-default = 0.0. G is the coefficient of K4DD if TYPE \neq TRANSIENT. G/W3 is coefficient of K1DD if TYPE = TRANSIENT.
- W3 - Input-real-default = 0.0. If TYPE = TRANSIENT G/W3 is the coefficient of K1DD. If W3 = 0.0 K1DD is not used.
- W4 - Input-real-default = 0.0. 1.0/W4 is the coefficient of K4DD if TYPE = TRANSIENT. If W4 = 0.0 K4DD is not used.
- NØK2PP - Input-integer-no default. NØK2PP \geq 0 indicates presence of K2PP.
- NØM2PP - Input-integer-no default. NØM2PP \geq 0 indicates presence of M2PP.
- NØB2PP - Input-integer-no default. NØB2PP \geq 0 indicates presence of B2PP.
- MPCF1 - Input-integer-no default. MPCF1 \geq 0 indicates presence of GM.
- SINGLE - Input-integer-no default. SINGLE \geq 0 indicates presence of single-point constraints.
- ØMIT - Input-integer-no default. ØMIT \geq 0 indicates presence of GØ.
- NØUE - Input-integer-no default. NØUE \geq 0 indicates presence of extra points.
- NØK4GG - Input-integer-no default. NØK4GG \geq 0 indicates presence of K4AA.
- NØBGG - Input-integer-no default. NØBGG \geq 0 indicates presence of BAA.
- KDEKA - Input-integer-no default. KDEKA \geq 0 indicates presence of MAA and KAA.
- MØDACC - Input-integer-default = -1. MØDACC \geq 0 requests computation of MDD (meaningful only if FØRM = MØDAL)

MODULE FUNCTIONAL DESCRIPTIONS

4.58.7 Method

If extra points are present ($N\emptyset UE \geq 0$) and multipoint constraints or omitted coordinates are present ($MPCF1 \geq 0$ or $\emptyset MIT \geq 0$), then

$$GM \Rightarrow GMD, \quad (1)$$

and

$$G\emptyset \Rightarrow G\emptyset D. \quad (2)$$

Subroutine GKAD1A performs these tasks.

If direct input matrices are present and m's, s's or o's are present, the direct input matrices are reduced from the p set to the d set. Let $[D_{pp}^2]$ be a direct input matrix,

$$[D_{pp}^2] = [K_{pp}^2], [M_{pp}^2] \text{ or } [B_{pp}^2]$$

1. If m's are present,

$$[D_{pp}^2] \Rightarrow \begin{bmatrix} D_{nn}^2 & D_{nm}^2 \\ D_{mn}^2 & D_{mm}^2 \end{bmatrix}. \quad (3)$$

(The e coordinates are included with the n coordinates). Then compute:

$$[D_{nn}^2] = [D_{nn}^2] + [D_{nm}^2] [G_m^d] + [G_m^d]^T [D_{mn}^2] + [G_m^d]^T [D_{mm}^2] [G_m^d].$$

2. If s's are present,

$$[D_{ff}^2] \Rightarrow \begin{bmatrix} D_{ff}^2 & D_{fs}^2 \\ D_{sf}^2 & D_{ss}^2 \end{bmatrix}, \quad (5)$$

where only $[D_{ff}^2]$ is saved. The e coordinates are included with the f coordinates.

3. If o's are present, first partition $[D_{ff}^2]$

$$[D_{ff}^2] \rightarrow \left[\begin{array}{c|c} D_{dd}^2 & D_{do}^2 \\ \hline D_{od}^2 & D_{oo}^2 \end{array} \right], \quad (6)$$

then:

$$[D_{dd}^2] = [D_{dd}^2] + [D_{do}^2] [G_o^d] + [G_o^d]^T [D_{od}^2] + [G_o^d]^T [D_{oo}^2] [G_o^d]. \quad (7)$$

Steps 1 through 3 are done for K2PP, M2PP and B2PP, using subroutines GKAD1C and GKAD1D.

If FØRM = MØDAL and MØDACC < 0, GKAD is done. If not, the a set matrices are expanded to the d set by adding zeros at extra points. Let $[D_{aa}]$ be an a set matrix. Then,

$$\left[\begin{array}{c|c} D_{aa} & 0 \\ \hline 0 & 0 \end{array} \right] \Rightarrow [D_{dd}^1] \quad (8)$$

The above step is done for KAA, BAA, MAA, and K4AA and is performed in subroutine GKAD1B.

Compute KDD, BDD and MDD.

1. For Frequency Response or Complex Eigenvalue Analysis (TYPE ≠ TRAN),

$$[K_{dd}] = (1+iG) [K_{dd}^1] + [K_{dd}^2] + i [K_{dd}^{14}], \quad (9)$$

$$[B_{dd}] = [B_{dd}^1] + [B_{dd}^2], \quad (10)$$

$$[M_{dd}] = [M_{dd}^1] + [M_{dd}^2]. \quad (11)$$

MODULE FUNCTIONAL DESCRIPTIONS

2. For Transient Analysis (TYPE = TRAN),

$$[K_{dd}] = [K_{dd}^1] + [K_{dd}^2] , \quad (12)$$

$$[B_{dd}] = [B_{dd}^1] + [B_{dd}^2] + \frac{G}{W3} [K_{dd}^1] + \frac{1.0}{W4} [K_{dd}^4] , \quad (13)$$

$$[M_{dd}] = [M_{dd}^1] + [M_{dd}^2] . \quad (14)$$

If W3 or W4 is zero, the corresponding matrices are ignored.

4.58.8 Subroutines

GKAD uses matrix utility routines SSG2B, SSG2C, CALCV, MERGE, UPART, and MPART. Descriptions for these routines can be found in Section 3.

4.58.8.1 Subroutine Name: GKAD1A

1. Entry Point: GKAD1A
2. Purpose: To expand GM or GØ to d size matrices:

$$[G_m : o] \Rightarrow [G_m^d] \quad (15)$$

3. Calling Sequence: CALL GKAD1A (USETD,GØ,GØD,SCR1,UE,UA,UNE)

USETD - GINØ file number of USETD - integer - input.
 GØ - GINØ file number of GØ - integer - input.
 GØD - GINØ file number of GØD - integer - input.
 SCR1 - GINØ file number of scratch file - integer - input.
 UE - Pointer to UE bit in USETD word - integer - input.
 UA - Pointer to UA bit in USETD word - integer - input.
 UNE - Pointer to UNE bit in USETD word - integer - input.

4.58.8.2 Subroutine Name: GKAD1B

1. Entry Point: GKAD1B
2. Purpose: To expand a set matrices to d set size.
3. Calling Sequence: CALL GKAD1B (USETD,KAA,MAA,BAA,K4AA,K1DD,M1DD,B1DD,K41DD,UA,UE,UD,SCR1)

USETD - GINØ file number of USETD - integer - input.
 KAA - GINØ file number of KAA - integer - input.
 MAA - GINØ file number of MAA - integer - input.
 BAA - GINØ file number of BAA - integer - input.
 K4AA - GINØ file number of K4AA - integer - input.

MODULE FUNCTIONAL DESCRIPTIONS

K1DD - GINØ file number of K1DD - integer - input.
M1DD - GINØ file number of M1DD - integer - input.
B1DD - GINØ file number of B1DD - integer - input.
K41DD - GINØ file number of K41DD - integer - input.
SCR1 - GINØ file number of scratch file - integer - input.
UA - Pointer to UA bit in USETD word - integer - input.
UE - Pointer to UE bit in USETD word - integer - input.
UD - Pointer to UD bit in USETD word - integer - input.

4.58.8.3 Subroutine Name: GKAD1C

1. Entry Point: GKAD1C
2. Purpose: To initialize GKAD1D.
3. Calling Sequence: CALL GKAD1C (GMD,GØD,SCR1,SCR2,SCR3,SCR4,SCR5,SCR6,USETD)
GMD,GØD,USETD are GINØ file numbers of their respective data blocks - integer - input.
SCR1,...,SCR6 are GINØ file numbers of six scratch files - integer - input.

4.58.8.4 Subroutine Name: GKAD1D

1. Entry Point: GKAD1D
2. Purpose: To reduce "2PP" matrices to "2DD" matrices.
3. Calling Sequence: CALL GKAD1D (K2PP,K2DD)
K2PP - GINØ file number of input matrix - integer - input.
K2DD - GINØ file number of reduced matrix - integer - input.

4.58.9 Design Requirements

Six scratch files are necessary. Open core for GKAD1A and GKAD1B is defined at /GKADA1/.
Open core for GKAD1C and GKAD1D is defined at /GKADC1/.

4.58.10 Diagnostic Messages

None

FUNCTIONAL MODULE CEAD (COMPLEX EIGENVALUE ANALYSIS - DISPLACEMENT)

4.59 FUNCTIONAL MODULE CEAD (COMPLEX EIGENVALUE ANALYSIS - DISPLACEMENT)

4.59.1 Entry Point: CEAD

4.59.2 Purpose

To solve the equation

$$([M]p^2 + [B]p + [K])\{u\} = \{0\} \quad (1)$$

for the eigenvalues p and the associated eigenvectors $\{u\}$ where $[M]$, $[B]$ and $[K]$ are mass, damping and stiffness matrices respectively.

4.59.3 DMAP Calling Sequence

CEAD KDD,BDD,MDD,EED,CASECC/PHID,CLAMA,ØCEIGS,PHIDL/V,N,NFØUND \$

4.59.4 Input Data Blocks

KDD - Dynamic stiffness matrix - d set.
BDD - Dynamic damping matrix - d set.
MDD - Dynamic mass matrix - d set.
EED - Eigenvalue Extraction Data.
CASECC - Case Control Data Table.

Notes:

1. EED must be present.
2. CASECC must be absent when a substructure modal reduce is being performed. In all other cases it must be present.
3. At least one of KDD, BDD and MDD must be present.

4.59.5 Output Data Blocks

PHID - Complex eigenvectors in the d set.
CLAMA - Complex eigenvalue table.
ØCEIGS - Complex eigenvalue summary table.
PHIDL - Left complex eigenvectors in the d set.

MODULE FUNCTIONAL DESCRIPTIONS

Notes:

1. PHID, CLAMA and ØCEIGS must be present.
2. PHIDL may be purged if the left-hand vectors are not desired.

4.59.6 Parameters

NFØUND - Output-integer-no default. NFØUND indicates the number of eigenvalues found. If none were found, NFØUND is set to -1.

4.59.7 Method

The Complex Eigenvalue Analysis Module calculates the eigenvalues and eigenvectors for a general system which may have complex terms in the mass, damping, and stiffness matrices. The eigenvectors are scaled according to the user-requested normalization scheme. Modal masses are not calculated since they will, in general, be complex, and their value is rather dubious. The form of the problem solved by the Complex Eigenvalue Analysis Module is given in Equation 1.

The eigenvalues p and the eigenvectors $\{u\}$ are always treated as complex. These data are related to the u_d displacements if a direct formulation is used or are related to the u_h displacements if a modal formulation is used. The method to be used and the necessary data are selected by calling for one ID number in the EED data block. A set of EED data which defined either the Determinant Method, the Inverse Power Method, the Tridiagonal Reduction Method or the Hessenburg Method must be used. Subroutine CDETM, CFEER, CINVPR, or HESS1 is called to solve the eigenvalue problem (see subroutine descriptions below for method details. The eigenvalues and associated vectors are sorted by the magnitude of the imaginary part of the eigenvalue with all positives listed ahead of all negatives. (Subroutine CEAD1A).

4.59.8 Subroutines

The subroutines used by CEAD can be divided into six groups: 1) those used by CEAD; 2) those used for the Inverse Power Method; 3) those used by the Determinant Method; 4) those used by the Hessenburg Method; 5) those used by the Tridiagonal Reduction (FEER) Method; and 6) general utility routines. The descriptions of the utility routines can be found in Section 3.

FUNCTIONAL MODULE CEAD (COMPLEX EIGENVALUE ANALYSIS - DISPLACEMENT)

<u>CEAD</u>	<u>Determinant</u>	<u>Hessenberg</u>	<u>Inverse Power</u>		<u>FEER</u>		<u>General</u>
CEAD1A	CDETM	ALLMAT	CDIFBS	CMTIMU	CFCNTL	CFE1MY	ADD
CLVEC	CDETM2	CFACR	CDIVID	CNØRM	CFEER1	CFE2MY	CDCØMP
	CDETM3	CFBSØR	CINFBS	CNØRM1	CFEER2	CF1FBS	PRELØC
	CDTFBS	HESS1	CINVPR	CSQRTX	CFEER3	CF2FBS	
	CSQRTN	HESS2	CINVPR	CSUB	CFEER4	CFNØR1	
	CSUMM	MERGED	CINVPR	CXTRNY	CFER3D	CFNØR2	
			CINVPR	ØRTHØ	CFER3S	CF1ØRT	
					CFE1AØ	CF2ØRT	
					CFE2AØ		

4.59.8.1 Subroutine Name: CEAD1A

1. Entry Point: CEAD1A
 2. Purpose: To sort the eigenvalues and the right and left eigenvectors.
 3. Calling Sequence: CALL CEAD1A (LAMAI,PHII,PHIIL,LAMAØ,PHIØ,PHIØL,NFØUND,NVECT,CAPP)
- LAMAI - GINØ file number of unsorted eigenvalues - integer - input.
 PHII - GINØ file number of unsorted eigenvectors - integer - input.
 PHIIL - GINØ file number of unsorted left eigenvectors (Inverse Power Method only) - integer - input.
 LAMAØ - GINØ file number of data block CLAMA - integer - input.
 PHIØ - GINØ file number of data block PHID - integer - output.
 PHIØL - GINØ file number of data block PHIDL - integer - output.
 NFØUND - Number of eigenvalues found - integer - input.
 NVECT - Number of eigenvectors found - integer - input.
 CAPP - Method - BCD - input.

4.59.8.2 Subroutine Name: CINVPR

1. Entry Point: CINVPR
 2. Purpose: CINVPR is the main driver for the Complex Inverse Power Method of eigenvalue extraction.
 3. Calling Sequence: CALL CINVPR (EED,METHØD,NFØUND)
- COMMON / CINVPR / K(7),M(7),B(7),LAM(7),PHI(7),EIGSUM,SCRFIL(11),NØREG,EPS,REG(7,10),PHIDL
 COMMON / CINVX / Z(1)

MODULE FUNCTIONAL DESCRIPTIONS

- K,M,B - Input matrix control blocks for the stiffness, mass, and damping matrices (K), [M], and [B].
- LAM,PHI - Matrix control blocks for the output eigenvalue and eigenvector files.
- EIGSUM - The output eigenvalue summary file.
- SCRFIL(11) - Eleven scratch files available to Inverse Power
- NØREG - Number of regions input to CINVPR.
- EPS - Convergence criterion.
- REG(7,10) - Storage space for up to 10 region parameters.
- PHIDLI - GINØ file number of unsorted left eigenvector file.
- Z(1) - Open core for CINVPR.

MODULE FUNCTIONAL DESCRIPTIONS

EED - GINØ file number for this input data block.

METHØD - ID of an EIGC card for the Inverse Power Method.

NFØLND - Number of eigenvalues found.

4. Method: Complex Inverse Power was, in general, designed identically to Real Inverse Power. The notable exceptions are in the iteration equation and the orthogonalization with respect to previously extracted eigenvectors. With these points in mind, the basic flow can be taken from READ in section 4.48. Theoretical development is given in the Theoretical Manual.

4.59.8.3 Subroutine Name: CINVP1

1. Entry Point: CINVP1

2. Purpose: To generate calling sequences to ADD to form

$$[A] = [K] + \lambda[B] + \lambda^2[M]. \quad (2)$$

3. Calling Sequence: CALL CINVP1

COMMON /CINVPX/ K(7),M(7),B(7),DUM(15),A

COMMON /CINVIX/Z(1)

COMMON /CINVXX/ LAMBDA

K,M,B - Matrix control blocks for the input matrices.

A - GINØ file number for the output matrix.

Z(1) - Area of open core available to ADD.

LAMBDA - Complex double precision scalar multiplier.

4.59.8.4 Subroutine Name: CINVP2

1. Entry Point: CINVP2

2. Purpose: To initialize and call CDCØMP for subroutine CINVPR.

FUNCTIONAL MODULE CEAD (COMPLEX EIGENVALUE ANALYSIS - DISPLACEMENT)

Γ

3. Calling Sequence: CALL CINVP2

COMMON /CINVPX/ DUM(36),A,XX,L,U,SCR1,SCR2,SCR3,LL,UU

COMMON /CINVXX/DUMM(4),SWITCH

COMMON /CINV2X/Z(1)

- A - GINØ file number for the input matrix.
- L,U - GINØ file number for the lower and upper triangular factors output from CDCØMP.
- SCR1,SCR2,SCR3 - Three scratch files used by CDCØMP.
- LL,UU - GINØ file numbers for alternate storage of L and U.
- SWITCH - $\left\{ \begin{array}{l} 0, \text{ store factors on L and U.} \\ 1, \text{ store factors on LL and UU.} \\ -204, \text{ store factors on LL and UU, and recompute length of open core.} \\ \text{(Left-hand eigenvector calculations only)} \end{array} \right.$
- Z(1) - Area of open core used by CDCØMP.

4.59.8.5 Subroutine Name: CINVP3

1. Entry Point: CINVP3

2. Purpose: To solve for a complex eigenvalue and eigenvector using the Inverse Power Method.

3. Calling Sequence: CALL CINVP3

COMMON /CINVPX/K(7),M(7),B(7),LAM(7),PHI(7),XXX,SCRFIL(11)

COMMON /CINV3X/Z(1)

See section 4.59.8.2 above for details on /CINVPX/.

Z(1) - Area of open core available in CINVP3.

4. Method: The logic flow and the mathematical equations are essentially identical to INVP3, with the following exceptions. The eigenvalues and eigenvectors are found corresponding to the matrix equation

$$(\lambda^2[M] + \lambda[B] + [K]) [\phi] = [0] \quad (3)$$

where the iteration equation is given by

$$(\lambda_0^2[M] + \lambda_0[B] + [K]) \{w_n\} = -([B] + \lambda_0[M]) \{u_{n-1}\} - [M] \{v_{n-1}\} \quad (4)$$

with

$$\{\bar{u}_n\} = \frac{1}{c_n} \{w_n\}, \quad (5)$$

$$\{\bar{v}_n\} = \lambda_0 \{\bar{u}_n\} + \frac{1}{c_n} \{u_{n-1}\}, \quad (6)$$

$$\{u_n\} = \{\bar{u}_n\} - \sum_1 \alpha_1 \{\phi_1\}, \quad (7)$$

$$\{v_n\} = \{\bar{v}_n\} - \sum_1 \alpha_1 \lambda_1 \{\phi_1\}, \quad (8)$$

and

$$\alpha_1 = \frac{\{\phi_1\}^T [\lambda_1[M] \{\bar{u}_n\} + [M] \{\bar{v}_n\} + [B] \{\bar{u}_n\}]}{\{\phi_1\}^T (2\lambda_1 [M] + [B]) \{\phi_1\}}. \quad (9)$$

Γ

where

ϕ_i = Previously extracted right-hand vector,

$\bar{\phi}_i$ = Previously extracted left-hand vector,

C_n = Largest element (in magnitude) of $\{W_n\}$, and

λ_i = Previously extracted eigenvalue.

The above equations replace Equations 19 through 22 in section 4.48. The calculation of the remaining equations remains the same except for the use of complex arithmetic. The left eigenvector is obtained by decomposing Equation 3 with $\lambda_0 - \lambda_i$ and using CDIFBS to make the appropriate substitution using the factors from CDCØMP.

5. Design Requirements: CINVPS requires fourteen complex double precision vectors in core plus four GINØ buffers.

4.59.8.6 Subroutine Name: CNØRM

1. Entry Point: CNØRM

2. Purpose: To normalize successive iterated vectors such that the maximum element is equal to unity, and to return the normalizing divisor.

3. Calling Sequence: CALL CNØRM (X,DIV)

X - Input vector to be normalized.

DIV - Divisor which was used to normalize the vector corresponding to the argument X.

4.59.8.7 Subroutine Name: CNØRM1

1. Entry Point: CNØRM1

2. Purpose: To normalize a complex vector such that the largest magnitude of an element is equal to one.

3. Calling Sequence: CALL CNØRM (X,N)

X - Vector to be normalized.

N - Length of the vector (complex terms).

4.59.8.8 Subroutine Name: CINFBS

1. Entry Point: CINFBS
2. Purpose: To perform the forward-backward substitution necessary to solve an iteration of the Inverse Power Method.
3. Calling Sequence: CALL CINFBS (X,Y,BUF)
COMMON /CINFBX/L(7),U(7)
L,U - Matrix control blocks for the factors output from CDCOMP.
X - Complex double precision input vector.
Y - Complex double precision solution vector.
BUF - GINØ buffer.
4. Method: CINFBS is a stripped down version of GFBS. Both vectors reside in core, and only complex double precision arithmetic is used.

4.59.8.9 Subroutine Name: CDIFBS

1. Entry Point: CDIFBS
2. Purpose: To perform the forward-backward substitution necessary to solve for the left eigenvector.
3. Calling Sequence: CALL CDIFBS (X,BUF)
COMMON /CINVPX/DUM(41),UPRTRI,XXX,LØWTRI
UPRTRI,LØWTRI - Files containing the upper and lower triangular factors output from CDCOMP.
X - The output complex double precision left eigenvector.
BUF- GINØ buffer used by CDIFBS.
4. Method: CDIFBS actually solves the system of equations

$$[A]^T \{x\} = \{y\}, \quad (10)$$

where [A] has been decomposed into [A] = [L] [U]. To solve the transpose problem we have that

Γ

$$[A]^T = ([L] [U])^T = [U]^T [L]^T \quad (11)$$

so that

$$[U]^T [L]^T \{x\} = \{y\}. \quad (12)$$

CDIFBS is a modified form of GFBS which does the forward pass on [U] and the backward pass on [L]. All arithmetic operations are complex double precision.

4.59.8.10 Subroutine Name: CMTIMU

1. Entry Point: CMTIMU
2. Purpose: To pre-multiply a vector {y} by a matrix to obtain a vector {x}.
3. Calling Sequence: CALL CMTIMU (Y,X,FILE,BUF)

COMMON /CINVPX/DUM(7),M(7)

FILE - If FILE = 0, form {x} = [M]{y}.

FILE ≠ 0, form {x} = [A] {y}, where [A] is the matrix on FILE.

X,Y - Complex double precision vectors.

BUF - GINØ buffer.

4.59.8.11 Subroutine Name: CXTRNY

1. Entry Point: CXTRNY
2. Purpose: To form the inner product of two complex vectors, {x} and {y}

$$a = \{x\}^T \{\bar{y}\}, \quad (13)$$

where $\{\bar{y}\}$ denotes a vector all of whose components are the complex conjugates of {y}.

3. Calling Sequence: CALL CXTRNY (X,Y,A)

COMMON /CINVPX/XX,N

N - Length of the vectors.

X,Y - Complex double precision vectors.

A - Complex double precision value of the inner product of {x} and {y}.

4.59.8.12 Subroutine Name: CSUB

1. Entry Point: CSUB
2. Purpose: To evaluate the vector equation

$$\{z\} = a\{x\} - b\{y\}, \quad (14)$$

where $\{x\}$, $\{y\}$, a and b may be complex.

3. Calling Sequence: CALL CSUB (X,Y,Z,A,B)

COMMON /CINVPX/XXX,N

N - Length of the vectors $\{x\}$ and $\{y\}$.

X,Y,Z - Complex double precision vectors.

A,B - Complex double precision scalar multipliers.

4.59.8.13 Subroutine Name: ORTHO

1. Entry Point: ORTHO
2. Purpose: To orthogonalize a vector with respect to all previously extracted vectors.
3. Calling Sequence: CALL ORTHO (U,V,X1,X2,X3,X4,X5,NZ,BUF1,BUF2,BUF3,BUF4)

COMMON /CINVPX/K(7),M(7),B(7),LAMBDA(7),PHI(7),XXX,SCRFIL(10)

COMMON /CINVXX/DUM(19),NRPTS

See section 4.59.8.2 for /CINVPX/ details.

NRPTS - Number of eigenvectors already extracted.

U,V - Input-current vectors - Output - orthogonalized vectors.

X1,...,X5- Storage space for five complex double precision vectors.

NZ - The number of words of core available to ORTHO.

BUF1, $\left\{ \begin{array}{l} \vdots \\ \text{BUF4} \end{array} \right\}$ - Four GIN buffers.

4. Method: ORTHO solves the equations

$$\{u_n\} = \{u_n\} - \sum_1 \alpha_1 \lambda_1 \{\phi_1\}, \quad (15)$$

$$\{v_n\} = \{v_n\} - \sum_i \alpha_i \lambda_i \{\phi_i\}, \quad (16)$$

where

$$\alpha_i = \frac{\{\bar{\phi}_i\}^T [\lambda_i [M] \{u_n\} + [M] \{v_n\} + [B] \{u_n\}]}{\{\bar{\phi}_i\}^T [2 \lambda_i [M] + [B]] \{\phi_i\}} \quad (17)$$

and

$\{\bar{\phi}_i\}$ = Previously found left eigenvectors.

$\{\phi_i\}$ = Previously found right eigenvectors.

λ_i = Previously found eigenvalues.

Note that the demoninator of equation 17 is constant with respect to the current iterate u_n and v_n . Thus it is computed once for each vector and saved on the left vector scratch file in place of the left vector.

MODULE FUNCTIONAL DESCRIPTIONS

4.59.8.14 Subroutine Name: CDETM

1. Entry Point: CDETM
 2. Purpose: To solve the complex eigenvalue problem by the Determinant Method.
 3. Calling Sequence: CALL CDETM (METHOD,EED,M,B,K,LAMA,PHID,ØCEIGS,NFØUND,SCR1,SCR2,SCR3,SCR4,SCR5,SCR6,SCR7,SCR8)
- METHOD - ID of an EIGC card for the Determinant Method - integer - input.
- EED,ØCEIGS, } - GINØ file numbers of their respective data blocks - integer - input.
M,B,K
- LAMA - GINØ file number of temporary eigenvalue storage file - integer - input.
- PHID - GINØ file number of temporary eigenvector storage file - integer - input.
- NFØUND - Number of eigenvalues found - integer - output.
- SCR1,SCR2, } - GINØ file numbers of 8 scratch files - integer - input.
...,SCR8
4. Method: The overall flow and theoretical considerations of the Determinant Method are explained in section 4.88. Two refinements are made in CDETM. The first is the handling of multiple search regions, which allows the user to control the distribution of starting points in the complex plane. See the EIGC bulk data card description in section 2 of the User's Manual for further details. The second is the use of the EIGP card to define poles which will be swept from the determinant as if they were previously accepted eigenvalues. This allows the user to prevent convergence to known or already extracted eigenvalues.
 5. Design Requirements: CDETM requires two complex double precision d set vectors plus one GINØ buffer in core.

4.59.8.15 Subroutine Name: CDETM2

1. Entry Point: CDETM2
 2. Purpose: To arrange 3 starting points in order of the magnitude of the determinant.
 3. Calling Sequence: CALL CDETM2(P,D,IP,PR,PI,DR,DI,IPS)
- P - Three starting point values - input-complex double precision.

FUNCTIONAL MODULE CEAD (COMPLEX EIGENVALUE ANALYSIS - DISPLACEMENT)

┌

- D - Scaled determinants at P - input-complex double precision.
- IP - Scale factors for D - input - integer.
- PR - Real parts of the reordered starting points - output-double precision.
- PI - Imaginary parts of the reordered starting points - output-double precision.
- DR - Real parts of the reordered determinants - output-double precision.
- DI - Imaginary parts of the reordered determinants - output-double precision.
- IPS - Scale factors of the reordered determinants - output - integer.

4.59.8.16 Subroutine Name: CSUMM

1. Entry Point: CSUMM
2. Purpose: To add two scaled complex numbers together.
3. Calling Sequence: CALL CSUMM (D1,D2,ID1,D3,D4,ID2,D5,D6,ID3)

The arguments are defined in the following equation:

$$(D1,D2) \times 10^{ID1} + (D3,D4) \times 10^{ID2} = (D5,D6) \times 10^{ID3}, \quad (18)$$

where all Di's are double precision.

┌

4.59.8.17 Subroutine Name: CDTFBS

1. Entry Point: CDTFBS
 2. Purpose: To solve for the eigenvector given the decomposed impedance matrix.
 3. Calling Sequence: CALL CDTFBS (F,EV,BUFFER(1),FU,NRØW)
- F - Applied complex load vector - input-complex double precision.
 - EV - Eigenvector - output- complex double precision.
 - BUFFER(1) - GINØ buffer.
 - FU - Matrix control block for [U] - integer - input.
 - NRØW - Order of problem - integer - input.

MODULE FUNCTIONAL DESCRIPTIONS

4.59.8.18 Subroutine Name: CDETM3

1. Entry Point: CDETM3
2. Purpose: To rescale a scaled complex number.
3. Calling Sequence: CALL CDETM3(D1,D2,ID1)

Let $\overline{D1}$, $\overline{D2}$, $\overline{ID1}$ be the input values of D1, D2, ID1. On return from CDETM3

$$(D1,D2) \times 10^{\overline{ID1}} = (\overline{D1},\overline{D2}) \times 10^{\overline{ID1}}, \quad (20)$$

and

$$1.0 \leq |(D1,D2)| \leq 10.0, \quad (21)$$

where all D1's are double precision.

4.59.8.19 Subroutine Name: CDIVID

1. Entry Point: CDIVID
2. Purpose: To divide a complex vector by a complex number.
3. Calling Sequence: CALL CDIVID (DIV,V,V1,NV)

where V is a complex D.P. vector of length NV to be divided by DIV and the answer put in V1.

4.59.8.20 Subroutine Name: HESS1

1. Entry Point: HESS1
2. Purpose: HESS1 is the overall driver for the upper Hessenburg method of complex eigenvalue extraction. CEAD will call HESS1 if sufficient core exists ($6N^2 + 8N$) to use this method where N is the order of the reduced problem. Otherwise it will select complex inverse power. If the B matrix is not null $N = 2*N$.
3. Calling Sequence: CALL HESS1 (KDD,MDD,CLAMA,PHID,ØCEIGS,NFØUND,NVECD,BDD,SCR1,SCR2,SCR3,SCR4,SCR5,SCR6,SCR7,EED,METHØD)

where

KDD, MDD, CLAMA, PHID, ØCEIGS, BDD, SCRI-7, and EED are the GINØ file numbers of their respective data blocks.

NFØUND - integer-output-NFØUND is the number of eigenvalues found.

NVECD - integer-output-NVECD is the number of eigenvalues output.

METHØD - integer-input-METHØD is the EIGC Id selected in CASECC by the CMETHØD card.

4. Method: HESS1 transforms the eigenvalue problem $[P^2[MDD] + P[BDD] + [KDD]] \{\phi_D\} = \{0\}$ into $[k^2 I + A]\{\phi_{D1}\} = \{0\}$ according to the following procedures:

If BDD = 0

$$[A] = [MDD]^{-1} [KDD]$$

$$P = \sqrt{k^2} \text{ (with positive Im)}$$

$$\{\phi_D\} = \{\phi_{D1}\}$$

If BDD \neq 0

$$A = \left[\begin{array}{c|c} 0 & -I \\ \hline MDD^{-1} KDD & MDD^{-1} BDD \end{array} \right]$$

$$P = k^2$$

$$\{\phi_{D1}\} = \left\{ \begin{array}{c} \phi_D \\ 0 \end{array} \right\}$$

MDD must be non-singular and a special case exists if MDD is the identity (Form = 8).

The [A] matrix is put into core and subroutine ALLMAT is called.

5. Subroutines Called: CFBSØR, MERGED, CFACTØR, ALLMAT, and HESS2

6. Design Requirements:

1. Open core must be available at /HESS1X/
2. All computations are done single precision.

4.59.8.21 Subroutine Name: HESS2

1. Entry Point: HESS2

MODULE FUNCTIONAL DESCRIPTIONS

2. Purpose: To form an NRØW X NRØW identity matrix and a proper partitioning vector to insert this identity matrix in a larger matrix.
3. Calling Sequence: CALL HESS2 (NRØW,IDEN,IVP)

where:

NRØW is the order of the identity matrix to be written on IDEN

IDEN - GINØ file number of the Identity matrix

IVP - GINØ file number of the partitioning vector, {IPV} =

$$\left\{ \begin{array}{c} 1.0 \\ \vdots \\ 0 \\ \vdots \\ 0 \end{array} \right\} \begin{array}{l} \text{NRØW} \\ \\ \text{NRØW} \end{array}$$

4.59.8.22 Subroutine Name: MERGED

1. Entry Point: MERGED
2. Purpose: To set up and call the matrix utility routine MERGE to perform the following merge

$$\{CP\} \left[\begin{array}{c|c} A11 & A12 \\ \hline A21 & A22 \end{array} \right] \Rightarrow [A]$$

← RP →

3. Calling Sequence:

CALL MERGED (A11,A12,A21,A22,A,RP,CP,N1,N2)

where:

- A11, A12, A21, A22, A, RP, and CP are the GINØ file numbers of their respective data blocks. If any partition does not exist its GINØ file number should be 0. All input data blocks should have matrix trailers. RP or CP may be 0. N1 and N2 are the orders of RP and CP if either is zero.
4. Method: MERGED sets up /PARMEG/ and calls MERGE. It also writes trailer on A.
 5. Design Requirements: Open core exists at /MRGEDX/

FUNCTIONAL MODULE CEAD (COMPLEX EIGENVALUE ANALYSIS - DISPLACEMENT)

4.59.8.23 Subroutine Name: CFACTR

1. Entry Point: CFACTR
2. Purpose: CFACTR will decompose a complex matrix A into its two factors LLL and ULL.
3. Calling Sequence: CALL CFACTR (A,LLL,ULL,SCR1,SCR2,SCR3,IØPT)

where:

A, LLL, ULL, SCR1, SCR2, and SCR3 are the GINØ file numbers of their respective data blocks.

IØPT is output according to the following table:

<u>Decomp Method</u>	<u>IØPT</u>
CSSP	4
CSDP	2
CUSP	1

4. Method: The various decomposition methods are chosen based on /SYSTEM/(55) and the trailer of A. CFACTR will write a trailer on the factors.
5. Subroutines Called: CFACTR may call SCDCMP, CSPSDC, or CDCØMP.
6. Design Requirements: Open core must be at /CFACTX/

4.59.8.24 Subroutine Name: CFBSØR

1. Entry Point: CFBSØR
2. Purpose: The purpose of CFBSØR is to solve the complex matrix equation $[A][X] = [B]$ given the factors of $[A] = [LLL][ULL]$.
3. Calling Sequence: CALL CFBSØR (LLL,ULL,B,X,IØPT)

where:

LLL,ULL, B and X are the GINØ file names of their respective matrices.

IØPT is an input parameter set by CFACTR (See Section 4.59.8.24).

4. Method: The trailers of [LLL], [ULL] and [B] are used to prepare for FBS. A trailer is written on [X].
5. Subroutines Called: CXFBS or GFBS may be called based on IØPT.
6. Design Requirements: Open core at /CFBSRX/.

MODULE FUNCTIONAL DESCRIPTIONS

4.59.8.25 Subroutine Name: ALLMAT

1. Entry Point: ALLMAT

2. Purpose: The purpose of ALLMAT is to compute the eigenvalues (All) and eigenvectors (number requested) of an arbitrary complex matrix by use of the QR algorithm and the Wielandt inverse power method for vectors. This is essentially the routine distributed via SHARE as SDA 3441.

3. Calling Sequence: CALL ALLMAT (A,LAMBDA,H,HL,VECT,MULT,INTH,M,NCAL,IØPT1)

where:

A - is the M X M complex input matrix. On return A contains the complex eigenvectors.

LAMBDA - is a complex list of eigenvalues.

H - is a complex M X M working array.

HL - is a complex M X M working array

VECT - is a complex M order array

MULT - is a complex M order array

INTH and INT are real M order arrays

M - is the problem order.

NCAL - is the number of eigenvectors to compute.

IØPT1 - is not used.

4. Method: The input matrix A is reduced to an upper Hessenberg matrix, H, by a sequence of elementary triangular and permutation matrices which make up a matrix P such that $P^{-1}AP = H$. The QR algorithm is made use of in ALLMAT by applying unitary similarity transformations to Hessenberg matrices, $H_1: H_1 = P^{-1}AP$. $H_s = (h_{ij}^{(s)}) = Q_s^H H_{s-1} Q_s = Q_s^H Q_{s-1}^T Q_{s-1} H_{s-1} Q_{s-1} = T_s Q_s$ where Q_s^H is the product of plane rotations, chosen so that T_s is upper triangular. This process makes $h_{n,n-1}^{(s)}$ converge to zero and therefore $h_{nn}^{(s)}$ converges to an eigenvalue of A. When convergence is met ($h_{n,n-1}^{(s)}$ negligible) the Hessenberg matrix, H_s , is deflated (i.e., last row and column eliminated) and ALLMAT proceeds with its leading principal submatrix (a new H_1) of order one less. If $h_{n-1,n-2}^{(s)}$ becomes negligible, the eigenvalue of the lower right hand matrix of order two are calculated and ALLMAT proceeds with the leading principal matrix of order two less. It can be shown that convergence is accelerated by judiciously subtracting scalar matrices from the H_s matrices. ALLMAT actually replaces H_s by $H_s - k_s I$ such that k_s

FUNCTIONAL MODULE CEAD (COMPLEX EIGENVALUE ANALYSIS - DISPLACEMENT)

is one of the eigenvalues p_s or q_s of the lower right hand 2×2 matrix of H_s . The choice of p_s or q_s is made on the basis of whether $|h_{nn}^{(s)} - p_s|$ or $|h_{nn}^{(s)} - q_s|$ is a minimum. The shift technique is applied at each iteration.

Two passes of the Wielandt inverse power method are used to calculate the eigenvectors, y_i , of H . Very little work is required for the second pass since the necessary elementary triangular and permutation matrices are stored in MULT and INTH (storage areas internal to ALLMAT). Finally, the eigenvectors of A , Py_i , are calculated. The matrix P resides in INT and the lower part of H (INT and H are internal to ALLMAT).

References: The theory and a complete description of the algorithms appear in the first reference.

- (1) J. H. Wilkinson (1965): The Algebraic Eigenvalue Problem, Oxford.
- (2) A. S. Householder (1964): The Theory of Matrices in Numerical Analysis, Blaisdell.

Authors

R. E. Punderlic, J. RinzoI, Scientific Programming Department, Central Data Processing Facility, Union Carbide Corporation, Nuclear Division, ORGDP, Oak Ridge, Tennessee.

5. Design requirements: The eigenvalues of A are not necessarily calculated in any absolute algebraic order.

Ten iterations per eigenvalue are allowed and examples exist for which convergence will not occur in ALLMAT (e.g., most lower triangular matrices with all equal eigenvalues; a matrix with ones on the lower diagonal, one as the N^{th} component of the first row and zeros elsewhere). In the case of non-convergence, ALLMAT will return a value for NCAL less than N and it is suggested that the user experiment with arbitrary shifts of the input matrix (i.e., add a constant to the diagonal of A) which will sometimes eliminate the difficulty (e.g., second example just stated).

If overflows or detrimental underflows occur, scaling A such that its largest element is in modulus about one will probably eliminate the difficulty.

The accuracy obtaining in computing the eigenvalues of A , $\lambda_i(A)$, is usually related to the spectral radius, $\rho(A)$, of the matrix A or more generally to some norm of A times the norm of its inverse. Hence the greater $\rho(A)/\min|\lambda_i|$ the fewer significant digits the smaller eigenvalues may have. Accuracy also decreases as the order of the matrix increases. Close eigenvalues are usually calculated with less accuracy than well separated ones. In most cases ALLMAT has yielded roots and vectors accurate to about six significant digits.

MODULE FUNCTIONAL DESCRIPTIONS

4.59.8.26 Subroutine Name: CFCNTL

1. Entry Point: CFCNTL

2. Purpose: CFCNTL is the main driver for the complex version of the Tridiagonal Reduction (FEER) method.

3. Calling Sequence: CALL CFCNTL (EED,METHØD,NFØUND)

EED - GINØ file number for file containing EIGC information (Integer, input).

METHØD - Eigenvalue extraction method 'FEER' (Hollerith, input).

NFØUND - Accumulated number of acceptable eigensolutions (Integer, output).

COMMON/FEERAA/IK(7),IM(7),IB(7),ILAM(7),IPHI(7),IDMPFL,ISCR(11),REG(7,10),MCBLT(7),MCBUT(7),
MCBVEC(7),MCBLMB(7)

IK,IM,IB - Matrix control blocks for the input stiffness, mass and damping matrices, respectively.

ILAM,IPHI - Matrix control blocks for the output eigenvalue and eigenvector files.

IDMPFL - GINØ file number for the eigenvalue summary file (integer).

ISCR - GINØ file numbers for eleven scratch files (integer).

REG - Array of information obtained from the EIGC bulk data card and continuation cards.

MCBLT - Lower triangular matrix, [L], control block.

MCBUT - Upper triangular matrix, [U], control block.

MCBVEC - Orthogonal vector file control block.

MCBLMB - Matrix control block for $\lambda_0[M] + [B]$.

COMMON/FEERXC/LAMBDA(2),SYMMET,MREDUC,NØRD,IDIAG,EPS,NØRTHØ,NØRD2,NØRD4,NØRDP1,NSWP,JSKIP,

NØB,IT,TEN2MT,TENMHT,NSTART,QPR,JREG,NØREG,NZERØ,TENMTT,MINØPN,NUMØRT,NUMRAN

LAMBDA - Point of interest (i.e., center of neighborhood) in the complex plane (double precision).

SYMMET - Indicator for symmetric dynamic matrix (logical).

MREDUC - Size of the reduced problem (set internally).

NØRD - Problem size (set internally using the dimension of the stiffness matrix).

IDIAG - DIAG 12 output indicator (set internally).

EPS - The user specified (or default) desired theoretical accuracy of the eigenvalues expressed as a percentage (double precision).

FUNCTIONAL MODULE CEAD (COMPLEX EIGENVALUE ANALYSIS - DISPLACEMENT)

- NØRTHØ - Number of orthogonal vectors in present set (includes previously computed vectors).
- NØRD2 - $2 \times NØRD$.
- NØRD4 - $4 \times NØRD$.
- NØRDP1 - $NØRD + 1$.
- NSWP - Vector size for sweep algorithm.
- JSKIP - CFEER4 logic bypass indicator.
- NØB - Indicator for absence of damping matrix [B] (logical).
- IT - Number of decimal digits of accuracy, t , for the computer.
- TEN2MT - 10^{2-t} , where t is defined by IT; reorthogonalization accuracy criterion.
- TENMHT - $10^{-t/2}$, where t is defined by IT; accuracy criterion for disjoint tridiagonal matrix.
- NSTART - Number of initial reorthogonalization attempts (for the current point of interest).
- QPR - Indicator for very detailed printout (logical).
- JREG - Number (positive integer) of the current neighborhood.
- NØREG - Total number of neighborhoods, or points of interest, in the complex plane, to be processed.
- NZERØ - Number of previously obtained eigenvectors.
- TENMTT - $10^{-t/3}$, where t is defined by IT; rigid body root criterion.
- MINØPN - Minimum open core not used by the complex FEER process, in single precision words.
- NUMØRT - Total number of reorthogonalizations of all the trial vectors employed.
- NUMRAN - Total number of random starting and restart vectors used by the complex FEER process for all neighborhoods.

COMMON/FEERZC/Z(1)

- Z - Area of open core used by CFCNTL.

4. Design Requirements: The complex Feer method requires sufficient core for three GINØ buffers and five pairs of (complex) vectors of size NØRD ($2 \times NØRD$ when NØB=FALSE). The method can be performed in single or double precision.

MODULE FUNCTIONAL DESCRIPTIONS

5. Subroutine Glossary for the complex FEER method.

CFCNTL

CFEER1

CFEER2

CFEER3

CFEER4

CFER3S,CFER3D

CFE1A0,CFE2A0

CFE1MY,CFE2MY

CF1FBS,CF2FBS

CFN0R1,CFN0R2

CF10RT,CF20RT

Single and double precision versions.

4.59.8.27 Subroutine Name: CFEER1

1. Entry Point: CFEER1

2. Purpose: To set up the call to SADD to form the dynamic matrix

$$[D] = \lambda_0^2[M] + \lambda_0[B] + [K]$$

and also the sub-factor of the dynamic matrix,

$$[S] = \lambda_0[M] + [B]$$

3. Calling Sequence: CALL CFEER1

COMMON/FEERAA/

COMMON/FEERXC/

See CFCNTL for a description of /FEERAA/ and /FEERXC/ (Section 4.59.8.26)

COMMON/FEERZ1/Z(2)

Z - Area of open core used by CFEER1.

4. Design Requirements: CFEER1 can perform single or double precision operations.

4.59.8.28 Subroutine Name: CFEER2

1. Entry Point: CFEER2

FUNCTIONAL MODULE CEAD (COMPLEX EIGENVALUE ANALYSIS - DISPLACEMENT)

2. Purpose: To set up the call to CDCOMP to decompose the dynamic matrix

$$[D] = [L][U]$$

3. Calling Sequence: CALL CFEER2 (IRET)

IRET - Singularity indicator (output).

COMMON/FEERAA/

COMMON/FEERXC/

See CFCNTL for a description of /FEERAA/ and /FEERXC/ (Section 4.59.8.26).

COMMON/FEERZ2/Z(2)

Z - Area of open core used by CFEER2.

4.59.8.29 Subroutine Name: CFEER3

1. Entry Point: CFEER3

2. Purpose: CFEER3 is the basic driver routine which obtains the reduced tridiagonal matrix for the complex FEER method of eigenvalue extraction.

3. Calling Sequence: CALL CFEER3

COMMON/FEERAA/

COMMON/FEERXC/

See CFCNTL for a description of /FEERAA/ and /FEERXC/ (Section 4.59.8.26).

COMMON/FEERZ3/Z(2)

Z - Area of open core used by CFEER3.

4. Design Requirements: CFEER3 can perform single or double precision operations.

4.59.8.30 Subroutine Name: CFEER4

1. Entry Point: CFEER4

2. Purpose: CFEER4 obtains the physical eigenvalues and eigenvectors of the reduced problem (for the current point of interest) by feeding the reduced tridiagonal matrix to subroutine ALLMAT. In addition, CFEER4 sorts the eigenvalues according to increasing distance from the center of interest, and determines which eigensolutions are acceptable and which are not (according to the default or user-input accuracy criterion).

MODULE FUNCTIONAL DESCRIPTIONS

3. Calling Sequence: CALL CFEER4

COMMON/FEERAA/

COMMON/FEERXC/

See CFCNTL for a description of /FEERAA/ and /FEERXC/ (Section 4.59.8.26).

COMMON/FEERZ4/Z(2)

Z - Area of open core used by CFEER4.

4. Design Requirements: CFEER4 performs only single precision operations.

4.59.8.31 Subroutines CFER3S,CFER3D

1. Entry Points: CFER3S (single precision), CFER3D (double precision).

2. Purpose: To perform the tridiagonal reduction algorithm for CFEER3.

3. Calling Sequence: CALL CFER3S(V1,V1L,V2,V2L,V3,V3L,V4,V4L,V5,V5L,ZB,ZC)

CALL CFER3D (same arguments as for CFER3s).

V1,V1L
V2,V2L
V3,V3L
V4,V4L
V5,V5L } - Working space for five pairs of vectors, where a vector
 - pair consists of a right-hand vector (e.g., V1) and a
 left-hand vector (e.g., V1L).

ZB,ZC - Working space for two GINØ buffers

COMMON/FEERAA/

COMMON/FEERXC/

See CFCNTL for a description of /FEERAA/ and /FEERXC/ (Section 4.59.8.26).

4. Design Requirements: CFER3S and CFER3D are driven by the CFEER3 subprogram, which performs the necessary initialization and termination process. CFEER3 invokes either CFER3S or CFER3D according to the precision of the required input files. All vectors are dimensioned consistent with the initial problem size (NORD2 when NOB=TRUE and NORD4 otherwise). In addition, each left-hand vector must immediately follow its corresponding right-hand vector in core, since this configuration results in more streamlined coding and associated quicker execution.

4.59.8.32 Subroutines CFE1AØ,CFE2AØ

1. Entry Point: CFE1AØ (single precision), CFE2AØ (double precision).

FUNCTIONAL MODULE CEAD (COMPLEX EIGENVALUE ANALYSIS - DISPLACEMENT)

2. Purpose: To perform the eigenmatrix multiplication operation.

3. Calling Sequence: CALL $\begin{matrix} \text{CFE1A0} \\ \text{CFE2A0} \end{matrix}$ (TP0SE,V1,V2,V3,ZB) } Single and double precision versions

TP0SE - Indicator for transpose operation (logical).

V1 - Input vector.

V2 - Output vector.

V3 - Working space for one vector.

ZB - Working space for one GIN0 buffer.

C0MM0N/FEERAA/

C0MM0N/FEERXC/

See CFCNTL for a description /FEERAA/ and /FEERXC/ (Section 4.59.8.26)

4. Design Requirements: All vectors are of dimension consistent with the initial problem size (see Section 4.59.8.31).

4.59.8.33 Subroutines CFE1MY,CFE2MY

1. Entry Points: CFE1MY (single precision), CFE2MY (double precision).

2. Purpose: To perform the standard matrix-times-vector multiply function. The operation is

$$\{X\} = [M]\{Y\}$$

or

$$\{X\} = [M]^T\{Y\}$$

3. Calling Sequence: CALL $\begin{matrix} \text{CFE1MY} \\ \text{CFE2MY} \end{matrix}$ (TP0SE,Y,X,FILE,BUF).

MODULE FUNCTIONAL DESCRIPTIONS

- TPØSE - Indicator for transpose operation (logical).
Y - Input vector
X - Output vector.
FILE(7) - Input matrix control block for the required matrix (integer).
BUF - Working space for one GINØ buffer.

4. Design Requirements: All vectors are of dimension consistent with the initial problem size.

4.59.8.34 Subroutines CF1FBS,CF2FBS

1. Entry Point: CF1FBS (single precision), CF2FBS (double precision).
2. Purpose: To perform the operational inverse algorithm (forward and backward sweeps).
The operation is

$$\{X\} \leftarrow [L][U]\{X\}$$

or

$$\{X\} \leftarrow [U]^T[L]^T\{X\}$$

3. Calling Sequence: CALL CF1FBS (TPØSE,XØUT,IØBUF) Single and double
CF2FBS precision versions

- TPØSE - Indicator for transpose operation (logical).
XØUT - Input vector, which gets transformed to the output vector.
IØBUF - Working space for one GINØ buffer.

CØMMØN/FEERAA/

CØMMØN/FEERXC/

See CFCNTL for a description of /FEERAA/ and /FEERXC/ (Section 4.59.8.26).

4. Design Requirements: CF1FBS and CF2FBS use the direct output of the CDCØMP subprogram.
The vector is dimensioned consistent with half of the initial problem size when the damping matrix is present, and consistent with the initial problem size when the damping matrix is absent.

4.59.8.35 Subroutines CFNØR1,CFNØR2

1. Entry Point: CFNØR1 (single precision), CFNØR2 (double precision).

FUNCTIONAL MODULE CEAD (COMPLEX EIGENVALUE ANALYSIS - DISPLACEMENT)

2. Purpose: To normalize a pair of (right-hand and left-hand) complex vectors to magnitude unity. The operation is

$$\beta = [\{\bar{W}\}^T \{W\}]^{1/2}$$

$$\{V\} = \frac{1}{\beta} \{W\}$$

$$\{\bar{V}\} = \frac{1}{\beta} \{\bar{W}\}$$

where the bar denotes a left-hand vector.

3. Calling Sequence: CALL $\begin{matrix} \text{CFNØR1} \\ \text{CFNØR2} \end{matrix}$ (RIGHT,LEFT,SIZE2,ØPTIØN,RI) } Single and double precision versions

- RIGHT - Input right-hand complex vector, which gets transformed to the output right-hand vector when ØPTIØN = 0.
- LEFT - Input left-hand complex vector, which gets transformed to the output left-hand vector when ØPTIØN = 0.
- SIZE2 - Length of either vector in computer words (integer).
- ØPTIØN - Selects the desired option, as follows:
 - (0) Normalize the input vectors, and output the square root of the inner product in RI.
 - (1) Only output the inner product, in RI.
 - (2) Only output the square root of the inner product, in RI.
- RI - Inner product, or square root of the inner product, or the input factors (output; see ØPTIØN).

COMMON/FEERCX/

See CFCNTL for a description of /FEERCX/ (Section 4.59.8.26).

4.59.8.36 Subroutines CF1ØRT,CF2ØRT

1. Entry Point: CF1ØRT (single precision), CF2ØRT (double precision)
2. Purpose: To perform the reorthogonalization algorithm.
3. Calling Sequence: CALL $\begin{matrix} \text{CF1ØRT} \\ \text{CF2ØRT} \end{matrix}$ (SUCESS,MAXITS,TEN2MT,NZERØ,IØRTHØ,VR,VL,V1L,V2,V2L,ZB)
- SUCESS - Output indicator for successful reorthogonalization (logical).
- MAXITS - Input maximum allowed number of reorthogonalization iterations (integer).

MODULE FUNCTIONAL DESCRIPTIONS

- TEN2MT - Convergence tolerance (input).
- NZERØ - Number of orthogonal vector-pairs (right and left hand) from restart and prior neighborhoods (input).
- IØRTHØ - Number of orthogonal vector-pairs previously computed in the current neighborhood (input).
- VR,VL - Input (and transformed for output) right and left hand vectors, respectively, which are to be reorthogonalized with respect to all pairs of previously computed orthogonal vectors.
- V1,V1L
V2,V2L - Working space for four vectors.
- ZB - Working space for one GINØ buffer.

CØMMØN/FEERAA/

CØMMØN/FEERXC/

See CFCNTL for a description of /FEERAA/ and /FEERXC/ (Section 4.59.8.26).

4. Design Requirements: All vectors are dimensioned consistent with the initial problem size (see Section 4.59.8.31). In addition, V1L must immediately follow V1 in core.

4.59.8.37 Subroutine Name: CLVEC

1. Entry Point: CLVEC
2. Purpose: To generate the left eigenvectors for the Determinant and Hessenburg Methods if requested. (Note: The left eigenvectors are already generated during a normal Inverse Power approach.)
3. Calling Sequence: CALL CLVEC (LAMD,NVECT,PHIDL,IH,IBUF,IBUF1)
 - LAMD - GINØ file number of sorted eigenvalues - integer-input.
 - NVECT - Number of left eigenvectors to be calculated - integer-input.
 - PHIDL - GINØ file number of data block PHIDL - integer-input.
 - IH - Trailer for data block PHIDL - integer array-input,output.
 - IBUF - Open core pointer to GINØ buffer - integer-input.
 - IBUF1 - Open core pointer to GINØ buffer - integer-input.
4. Method: For each eigenvalue, λ_i , a dynamic matrix of the form $M\lambda_i^2 + B\lambda_i + K$ is formed and decomposed into its upper and lower triangular components U and L. An arbitrary load

FUNCTIONAL MODULE CEAD (COMPLEX EIGENVALUE ANALYSIS - DISPLACEMENT)

vector F is generated and the equation $U^t L^t \phi_i = F$ is solved for the left eigenvector ϕ_i . ϕ_i is normalized and packed into output file PHIDL. The process is repeated for the first NVECT eigenvalues on file LAMD. Inverse Power subroutines CINVP1, CINFP2, CDIFBS and CNØRM1 are used in performing the above operations.

4.95.9 Design Requirements

Open core is defined at /CEAD1X/ to process EED. Open core is defined at /CEAD1A/ for use by CEAD1A.

4.59.10 Diagnostic Messages

The following diagnostic messages may be generated: 3001, 3002, 3003, 3005, 3007, 3008, 3025 and 3045.

FUNCTIONAL MODULE VDR (VECTOR DATA RECOVERY)

4.60 FUNCTIONAL MODULE VDR (VECTOR DATA RECOVERY)

4.60.1 Entry Point: VDR

4.60.2 Purpose

VDR formats data blocks for input to the Output File Processor (OFP) and XY plot (XYPLØT) modules to provide a capability for output of vectors in the solution set.

4.60.3 DMAP Calling Sequence

$$\text{VDR} \quad \left\{ \begin{array}{l} \text{CASECC} \\ \text{CASEXX} \end{array} \right\}, \left\{ \begin{array}{l} \text{EQDYN} \\ \text{HEQDYN} \end{array} \right\}, \left\{ \begin{array}{l} \text{USETD} \\ \text{HUSETD} \end{array} \right\}, \left\{ \begin{array}{l} \text{PHID} \\ \text{UDVF} \\ \text{UDVT} \\ \text{PHIH} \\ \text{UHVT} \\ \text{HUDVT} \end{array} \right\}, \left\{ \begin{array}{l} \text{CLAMA} \\ \text{PPF} \\ \text{TØL} \\ \text{HTØL} \end{array} \right\}, \text{XYCDB}, \left\{ \begin{array}{l} \text{PNLD} \\ \text{PNLH} \\ \text{HPNLD} \end{array} \right\}, \left\{ \begin{array}{l} \text{ØPHID} \\ \text{ØUDVC1} \\ \text{ØUDV1} \\ \text{ØPHIH} \\ \text{ØUHVC1} \\ \text{ØUHV1} \\ \text{HØUDV1} \end{array} \right\}, \left\{ \begin{array}{l} \text{ØPNL1} \\ \text{HØPNL1} \end{array} \right\} /$$

$$\text{C,N}, \left\{ \begin{array}{l} \text{TRANRESP} \\ \text{FREQRESP} \\ \text{CEIGN} \end{array} \right\} / \text{C,N}, \left\{ \begin{array}{l} \text{DIRECT} \\ \text{MØDAL} \end{array} \right\} / \text{V,N}, \text{SØRT2/V,N}, \text{ØUTPUT/V,N}, \text{SDR2/V,N}, \text{FMØDE \$}$$

4.60.4 Input Data Blocks

$\left\{ \begin{array}{l} \text{CASECC} \\ \text{CASEXX} \end{array} \right\}$ - Case Control Data Table.

$\left\{ \begin{array}{l} \text{EQDYN} \\ \text{HEQDYN} \end{array} \right\}$ - Equivalence between external and internal number - Dynamics.

USETD - Displacement set definitions table - Dynamics.

$\left\{ \begin{array}{l} \text{PHID} \\ \text{UDVF} \\ \text{UDVT} \\ \text{PHIH} \\ \text{UHVT} \end{array} \right\}$ - Partition of displacement vector.

CLAMA - Complex eigenvalue, table.

PPF - Dynamic loads for frequency response - p set.

$\left\{ \begin{array}{l} \text{TØL} \\ \text{HTØL} \end{array} \right\}$ - Table of output times.

XYCDB - X Y Control Data Block

$\left\{ \begin{array}{l} \text{PNLD} \\ \text{PNLH} \\ \text{HPNLD} \end{array} \right\}$ - Non-Linear Load Vector.

HUSETD - Temperature set definitions table - Dynamics.

HUDVT - Partition of temperature vector.

MODULE FUNCTIONAL DESCRIPTIONS

Notes:

1. CASECC, EQDYN and USETD may not be purged.
2. PP may be purged only if UDV is purged.
3. PNL and XYCDB may be purged.

4.60.5 Output Data Blocks

$\left. \begin{array}{l} \emptyset PHID \\ \emptyset U D V C 1 \\ \emptyset U D V 1 \\ \emptyset P H I H \\ \emptyset U H V C 1 \\ \emptyset U H V 1 \end{array} \right\} - \text{Output displacement requests - Solution set.}$

$\left. \begin{array}{l} \emptyset P N L 1 \\ H \emptyset P N L 1 \end{array} \right\} - \text{Non-Linear Load Vector.}$

$H \emptyset U D V 1 - \text{Output temperature requests - Solution set.}$

Note: Output data blocks may be purged.

4.60.6 Parameters

The first parameter indicates a Rigid Format and must be one of the three names shown above. The second parameter indicates a direct or modal formulation and must be one of the two names shown above.

- $S \emptyset R T 2 - \text{Output-integer-no default. } +1 \text{ if any } S \emptyset R T 2 \text{ output is requested, } -1 \text{ otherwise.}$
- $\emptyset O U T P U T - \text{Output-integer-no default. } +1 \text{ if any output in the solution set is requested, } -1 \text{ otherwise.}$
- $S D R 2 - \text{Output-integer-no default. } +1 \text{ if any requests for output in the physical set are found in CASECC or XYCDB, } -1 \text{ otherwise.}$
- $F M \emptyset D E - \text{Input-integer-no default. If a modal formulation, } F M \emptyset D E = \text{mode number of the first mode. } F M \emptyset D E \text{ is not used in a direct formulation.}$

4.60.7 Method

4.60.7.1 General

VDR is the main control program for the module. VDRA is called to analyze the Case Control (CASECC) and XYCDB data blocks. If any requests for solution set output are found, VDRB is called to assemble the ØUDV1 output data block for processing by the ØFP. If the problem is a transient response problem, VDRB is called a second time to process any requests for non-linear load output.

4.60.7.2 Analysis of the Case Control and XYCDB Data Blocks

VDRA attempts to open the XYCDB data block. If it is purged, a return is given to VDR. Otherwise, the header record and first data record of XYCDB are skipped, and data applying to all subcases are read from the second data record. If no such data exist, a dummy master is created. Otherwise, the master data are reduced to a list of unique pairs. If only master data exist, flags are set appropriately.

For each record in the Case Control data block the following processing occurs:

1. The record is read into core. If no XYCDB subcase corresponds to the Case Control subcase, pointers are set to the master data. Otherwise, the master data and appropriate XYCDB subcase data are merged and reduced to unique pairs.
2. For each request for solution set output in XYCDB, the corresponding request in Case Control is examined. If no request is present in Case Control, the XYCDB request is reduced to a set in Case Control format, and a request for the set is turned on in Case Control. If the Case Control set is "ALL", no further action is taken. If the Case Control request is a set, the set is merged with the XYCDB set, and the request altered to reflect the new set (unless all points in the XYCDB set were already in the Case Control set). A flag is set if any new requests are formed.
3. When all requests for the current Case Control record have been analyzed, the record (as modified) is written on a scratch file.
4. When all Case Control records have been read, the GINØ file name for the Case Control data block is switched to the scratch file (unless no modifications were made to Case Control).

MODULE FUNCTIONAL DESCRIPTIONS

4.60.7.3 Preparation of Solution Set Output

The operations of VDRB are dependent on the Rigid Format being executed. VDRB operates in all six of the dynamics Rigid Formats. The initial operations in VDRB proceed as follows:

1. For a direct solution, or a modal solution with extra points, the second record of EQDYN is read into core. USETD is read into core.
2. If the problem is a direct solution, each entry in EQDYN is processed. The scalar index value (the 2nd word of each entry) is replaced by the scalar index value in the solution set plus a code indicating which components of the point are in the solution set.
3. If the problem is a modal solution with extra points, the scalar index of each extra point in EQDYN is replaced with a scalar index in the solution set. The scalar indices of all other points are replaced with zero.
4. If the problem is a complex eigenvalue problem, a list of mode numbers and complex eigenvalues is read into core from the CLAMA data block.
5. If the problem is a transient response problem, a list of times is read into core from the TØL data block.
6. If the problem is a frequency response problem, a list of frequencies is read into core from the PP data block.
7. The header record on the input file is skipped, and various parameters are initialized for the overall processing.

A record on the Case Control data block is read. The output request is examined. If the output is defined in terms of a set, pointers to the set definition are computed. The vector is unpacked in core (unless the vector is already in core in the case of velocities and accelerations for frequency problems).

Information is assembled to write the identification record on the output data block as follows.

1. For complex eigenvalues, the mode number and eigenvalue are picked up from the list in core.

FUNCTIONAL MODULE VDR (VECTOR DATA RECOVERY)

2. For frequency response, the frequency is picked up from the list in core. A comparison with the ØFREQ selection in Case Control is made. If the current frequency is not marked for output, the remainder of the calculations for the current vector are skipped.

3. For a transient problem, the time is picked up.

The identification record is written. Entries are written in the data record according to the request. The modified EQDYN table in core is used to pick up points in the vector to be output. Conversion to magnitude and phase is made if requested.

When all points in the current request have been processed, post processing occurs depending on the problem type as follows:

1. For complex eigenvalues, a pointer is updated to the next mode number and eigenvalue. If all eigenvectors have not been processed, the steps above are repeated. Otherwise, terminal processing is initiated.
2. For frequency response, if the vector just processed was a displacement vector, the corresponding velocity vector is determined by differentiating with respect to time.

$$\{v\} = i\omega \{u\}. \quad (1)$$

Similarly, if the vector just processed was a velocity vector, the corresponding acceleration vector is formed by differentiating with respect to time:

$$\{a\} = i\omega \{v\}. \quad (2)$$

If all vectors have not been processed, the steps above are repeated. Otherwise, terminal processing is initiated.

3. For transient response, pointers are updated so that the vectors will be processed in the order a) displacement, b) velocity, and c) acceleration. If all vectors have not been processed, the steps above are repeated. Otherwise, terminal processing is initiated.

The terminal processing consists of closing all files, writing a trailer on the output file and exiting.

MODULE FUNCTIONAL DESCRIPTIONS

4.60.8 Subroutines

4.60.8.1 Subroutine Name: VDR

1. Entry Point: VDR
2. Purpose: Main control program for the module.
3. Calling Sequence: CALL VDR

4.60.8.2 Subroutine Name: VDRA

1. Entry Point: VDRA
2. Purpose: To analyze the output requests in the Case Control and XYCDB data blocks.
3. Calling Sequence: CALL VDRA

4.60.8.3 Subroutine Name: VDRB

1. Entry Point: VDRB
2. Purpose: To process requests for solution set output and assemble the output data block.
3. Calling Sequence: CALL VDRB (INFIL, ØUTFL, IREQ)

INFIL - GINØ file name of the data block containing vectors to be output in the solution set.

ØUTFL - GINØ file name of the data block where solution set output will be written.

IREQ - Word position in the Case Control record where solution set output request is defined.

4.60.9 Design Requirements

4.60.9.1 Allocation of Core Storage

FUNCTIONAL MODULE VDR (VECTOR DATA RECOVERY)

The maximum storage requirements for the module are in VDRB. A general picture of core storage is as follows:

COMMON/VDRCDR/Z(1)		
1	EQDYN Table	} 2 words per entry, one entry for each point in the problem.
ILIST	List of eigenvalues, frequencies or times	
ICC+1	Case Control record	} 1, 2 or 3 words per entry, one entry for each eigenvalue, frequency or time.
IVEC	Unpacked Vector	
BUF3	Buffer for input file	} One word for each degree of freedom in the solution set. (two words if complex).
BUF2	Buffer for output file	
BUF1	Buffer for Case Control	

4.60.9.2 Environment

The Block Data program VDRBD initializes /VDRCDM/ with GINØ file names, data defining position of parameters in a Case Control record, data defining rigid formats and problem types, and miscellaneous data. It must be in core when VDR is executed.

The module VDR is designed to be executed as one overlay segment. Open core is defined by /VDRCDR/. Two scratch files are used.

FUNCTIONAL MODULE FRRD (FREQUENCY RESPONSE - DISPLACEMENT APPROACH)

4.61 FUNCTIONAL MODULE FRRD (FREQUENCY RESPONSE - DISPLACEMENT APPROACH)

4.61.1 Entry Point: FRRD

4.61.2 Purpose

To solve the matrix equation

$$[-\omega_i^2 [M] + i\omega [B] + [K]] [X] = [P(\omega_i)]$$

at a given set of frequencies ω_i and loads P (which may be functions of ω_i).

4.61.3 DMAP Calling Sequence

FRRD CASECC, USETD, DLT, FRL, GMD, GØD, {KDD}, {KHH}, {BDD}, {BHH}, {MDD}, {MHH}, PHIDH, DIT/UHV, PS, PD, PP/V, N, APP/
V, N, FØRM/V, N, LUSETD/V, N, MPCF1/V, N, SINGLE/V, N, ØMIT/V, N, NONCUP/V, N, FRQSET/C, Y, DECØMØPT=1 S

4.61.4 Input Data Blocks

CASECC - Case Control Data table.
USETD - Displacement set definitions table dynamics.
DLT - Dynamic Loads Table.
FRL - Frequency Response List.
GMD - Multipoint constraint transformation matrix - dynamics.
GØD - Omitted coordinate transformation matrix - dynamics.
KDD - Modal stiffness matrix - d set.
KHH - Modal stiffness matrix - h set.
BDD - Modal damping matrix - d set.
BHH - Modal damping matrix - h set.
MDD - Modal mass matrix - d set.
MHH - Modal mass matrix - h set.
PHIDH - Transformation matrix from d set to modal coordinates.
DIT - Direct Input Tables.

Notes:

1. CASECC cannot be purged.
2. USETD cannot be purged.
3. DLT cannot be purged.
4. FRL cannot be purged.
5. GMD cannot be purged if MPCF1 ≥ 0 .
6. GØD cannot be purged if ØMIT ≥ 0 .

MODULE FUNCTIONAL DESCRIPTIONS

7. PHIDH cannot be purged if FØRM = MØDAL.
8. DIT cannot be purged if a load uses tables.

4.61.5 Output Data Blocks

- UHV - Displacement vectors.
- PS - Partition of load vector matrix giving loads in s set.
- PD - Load vectors - d set.
- PP - Load vectors - p set.

Notes: 1. UHV, PD, and PP cannot be purged.
2. PS cannot be purged if SINGLE \geq 0.

4.61.6 Parameters

- APP - Input-BCD-no default. APP should be set equal to DISP.
- FØRM - Input-BCD-no default. FØRM = MØDAL implies a modal solution should be used.
- LUSETD - Input-integer-no default. LUSETD indicates length of p set.
- MPCF1 - Input-integer-no default. MPCF1 \geq 0 implies multipoint constraints present.
- SINGLE - Input-integer-no default. SINGLE \geq 0 implies single-point constraints present.
- ØMIT - Input-integer-no default. ØMIT \geq 0 implies omitted coordinates present.
- NØNCUP - Input-integer-no default. NØNCUP = -1 implies noncoupled solution if FØRM = MØDAL.
- FRQSET - Output-integer-no default. FRQSET is the set id of the selected frequency list from CASECC.

4.61.7 Method

4.61.7.1 Overview of the Method

The Frequency Response module for the displacement approach assembles a frequency-dependent load vector and solves for the steady-state, frequency response, displacement vectors. Various load sets are defined as functions of frequency. Combinations of these sets are used with the various specified frequencies. Load vectors for each frequency are formed and reduced to loads on the proper degree of freedom. The solutions for both direct formulation and coupled modal formulation are identical except that different matrices are used. The solution involves a triangular decomposition and back substitution using the type of arithmetic selected by the matrix types for each frequency according to the following table.

FUNCTIONAL MODULE FRRD (FREQUENCY RESPONSE - DISPLACEMENT APPROACH)

<u>Trailer</u>	<u>Complex Terms</u>	<u>Decomp Method</u>	
		<u>CDC</u>	<u>Other</u>
SYM	No	RSSP	RSDP
SYM	Yes	CSSP	CSDP
UNSYM	No	RUDP	RUDP
UNSYM	Yes	CUDP	CUDP

where:

SYM = symmetric dynamic matrix
 UNSYM = unsymmetric dynamic matrix
 RSSP = real symmetric single precision
 RSDP = real symmetric double precision
 RUDP = real unsymmetric double precision
 CUDP = complex unsymmetric double precision
 CSSP = complex symmetric single precision
 CSDP = complex symmetric double precision

The solutions for the uncoupled modal formulation are analytic equations.

4.61.7.2 Logical Phases

1. The load vectors for each desired frequency are assembled from the DLT data block. The DLØAD section of the DLT tells which load sets to use and what scale factors to use in combining the load sets. The data for each load set are given in the RLØAD or TLØAD section of the DLT. This is done in subroutine FRRD1A.
2. The total load vectors are partitioned and manipulated to produce load vectors on the solution coordinates. This is done in subroutine FRRD1B.
3. The matrix equation for displacements is now solved for each load combination and each frequency. The overall dynamic matrix is formed. The matrix is decomposed, and the displacements are formed by back substitution using the various loads. If the formulation is an uncoupled modal system, the displacements are calculated directly. This is done by subroutines FRRD1C and FRRD1D or FRRD1F.

MODULE FUNCTIONAL DESCRIPTIONS

4. The solution vectors are then resorted into load-frequency order. This work is done by subroutine FRRDIE.

4.61.7.3 Algorithms

1. Assembly of Load Vectors:

The frequency set id is extracted from CASECC. This frequency set is placed in core from the FRL and converted from radians to frequency. These frequencies are output into the header of PPF for later output identification. The load id is read from CASECC, found in DLT, and a table is constructed giving a simple id and a scale factor for each component. The DLT data are read for each simple id, and a list of the required tables is extracted. Core is allocated to hold as many load vectors as possible up to the number of frequencies. If tables are present, they are initialized and evaluated for all frequencies in core. The DLT is read, and two types of loads are constructed:

$$1) \quad \text{RL0AD1} \quad P(f) = A[C(f) + iD(f)]e^{i(\theta - 2\pi f\tau)} \quad (2)$$

$$2) \quad \text{RL0AD2} \quad P(f) = AB(f)e^{i(\phi(f) + \theta - 2\pi f\tau)} \quad (3)$$

where A, B, C, D, ϕ , θ and τ are user input constants or tables.

TL0AD loads are computed as follows:

FUNCTIONAL MODULE FRRD (FREQUENCY RESPONSE - DISPLACEMENT APPROACH)

Case 1. TLØAD1 data card, referencing a TABLED1, 2 or 3.

The $P(t)$ is given in terms defined on the above data cards by

$$P_j(t) = A_j Y_T \left(\frac{t - \tau_j - X1}{X2} \right) \quad (1a)$$

Y_T is a piecewise linear table, for the $(N-1)$ intervals $(x_1, x_2) \dots (x_{N-1}, x_N)$. Then,

$$P_j(\omega) = A_j e^{-i\omega\tau_j} X2 \sum_{i=1}^{N-1} \frac{x_{i+1} - x_i}{2} (L_i Y_i + R_i Y_{i+1}) \quad (1b)$$

where

$$L_i = e^{-i\omega(X1 + X2 \cdot x_i)} E_2(-i\omega X2(x_{i+1} - x_i)) \quad (1c)$$

$$R_i = e^{-i\omega(X1 + X2 \cdot x_{i+1})} E_2(i\omega X2(x_{i+1} - x_i)) \quad (1d)$$

$E_2(i\theta)$ is computed by special formulas for large and small θ , as defined in module IFT.

$E_2(-i\theta)$ is the conjugate of $E_2(\theta)$.

Case 2. TLØAD1 with TABLED4.

$$P_j(t) = \begin{cases} A_j \sum_{n=0}^N a_n \left(\frac{t - \tau_j - X1}{X2} \right)^n & X3 < t - \tau_j < X4 \\ 0 & \text{Otherwise} \end{cases} \quad (2a)$$

Then,

$$\tilde{P}_j(\omega) = A_j e^{-i\omega\tau_j} (\text{SUM}) \quad (2b)$$

$$(\text{SUM}) = e^{-i\omega X4} (X4 - X3) \sum_{n=0}^N \frac{\tilde{a}_n}{n+1} E_{n+1}(i\omega(X4 - X3)) \quad (2c)$$

$$\tilde{a}_n = \left(\frac{X4 - X3}{X2} \right)^n \sum_{m=0}^{N-n} \frac{(n+m)!}{n!m!} \left(\frac{X3 - X1}{X2} \right)^m a_{n+m} \quad (2d)$$

MODULE FUNCTIONAL DESCRIPTIONS

The function $E_n(i\theta)$ is computed as follows:

If $0 < \theta < 0.1$, compute for $n=N$

$$\text{Re } E_N(i\theta) = 1 - \frac{\theta^2}{(N+1)(N+2)} + \frac{\theta^4}{(N+1)\dots(N+4)} - + \dots \quad (2e)$$

$$\text{Im } E_N(i\theta) = \frac{\theta}{N+1} - \frac{\theta^3}{(N+1)\dots(N+3)} + - \dots \quad (2f)$$

Stop when last term $< 10^{-9}$.

Then, by recursion for $n=N-1, N-2, \dots, 1$

$$\text{Re } E_n(i\theta) = 1 - \frac{\theta}{n+1} \text{Im } E_{n+1}(i\theta) \quad (2g)$$

$$\text{Im } E_n(i\theta) = \frac{\theta}{n+1} \text{Re } E_{n+1}(i\theta) \quad (2h)$$

If $0.1 < \theta < \infty$, compute for $n=0$

$$\text{Re } E_0(i\theta) = \cos \theta \quad (2i)$$

$$\text{Im } E_0(i\theta) = \sin \theta \quad (2j)$$

Then, by recursion for $n=1, 2, 3, \dots, N$

$$\text{Re } E_n(i\theta) = \frac{n}{\theta} \text{Im } E_{n-1}(i\theta) \quad (2k)$$

$$\text{Im } E_n(i\theta) = \frac{n}{\theta} [1 - \text{Re } E_{n-1}(i\theta)] \quad (2l)$$

Case 3. TLØAD2 with

$$P_j(t) = \begin{cases} A_j \tilde{t}^B e^{c\tilde{t}} \cos(2\pi F\tilde{t} + P) & 0 < \tilde{t} < T_2 - T_1 \\ 0 & \text{Otherwise} \end{cases} \quad (3a)$$

$$\tilde{t} \equiv t - T_1 - \tau_j$$

B will be restricted to an integer ≥ 0 .

$$\tilde{P}_j(\omega) = A_j e^{-i\omega\tau_j} [R_2 + R_1] \left[\frac{(T_2 - T_1)^{B+1}}{2(B+1)} \right] \quad (3b)$$

FUNCTIONAL MODULE FRRD (FREQUENCY RESPONSE - DISPLACEMENT APPROACH)

$$R_2 = e^{\text{power}} E_{B+1}(z) \quad (3c)$$

$$\text{where power} = +iP + (c + i2\pi F)(T_2 - T_1) - i\omega T_2$$

$$\text{and } z = -[c + i2\pi F - i\omega](T_2 - T_1)$$

R_1 = same as R_2 except the sign of P and F are reversed.

$$E_B(z) = \begin{cases} 1 + \frac{z}{B+1} + \frac{z^2}{(B+1)(B+2)} + \dots & |z| < .1 \\ \text{(until last term} < 10^{-9} \text{)} & \\ \frac{B!}{z^B} \left(e^z - \sum_{k=0}^{B-1} \frac{z^k}{k!} \right) & |z| \geq .1 \end{cases} \quad (3d)$$

If all frequencies cannot be evaluated at once, additional passes through the DLT are made until all are evaluated. If additional subcases exist in CASECC, the above steps are repeated for each load.

MODULE FUNCTIONAL DESCRIPTIONS

2. Manipulation of Load Vectors:

The vectors produced in the previous sections are related to the p set. They are reduced by the following steps using data blocks USETD, GMD and GØD.

If MPCF1 \geq 0:

$$\{P_p\} \Rightarrow \left\{ \frac{\bar{P}_{ne}}{P_m} \right\}, \quad (4)$$

$$\{P_{ne}\} = \{\bar{P}_{ne}\} + [G_m^d]^T \{P_m\}. \quad (5)$$

If SINGLE \geq 0:

$$\{P_{ne}\} \Rightarrow \left\{ \frac{P_{fe}}{P_s} \right\} \quad (6)$$

$\{P_s\}$ is output on data block PS.

If ØMIT \geq 0:

$$\{P_{fe}\} \Rightarrow \left\{ \frac{\bar{P}_d}{P_o} \right\}, \quad (7)$$

$$\{P_d\} = \{\bar{P}_d\} + [G_o^d]^T \{P_o\}. \quad (8)$$

$\{P_d\}$ is output on PD.

If FØRM = MØDAL:

$$\{P_h\} = [\phi_{dh}]^T \{P_d\}. \quad (9)$$

3. Solution Phase:

For a direct formulation the equation to be solved is:

$$[-\omega^2 [M_{dd}] + i\omega [B_{dd}] + [K_{dd}]] \{u_d\} = \{P_d(\omega)\} \quad (10)$$

For a coupled formulation the equations to be solved is:

$$[-\omega^2 [M_{hh}] + i\omega [B_{hh}] + [K_{hh}]] \{u_h\} = \{P_h(\omega)\} \quad (11)$$

FUNCTIONAL MODULE FRRD (FREQUENCY RESPONSE - DISPLACEMENT APPROACH)

The left hand matrix is generated by two calls to ADD and decomposed. The normal matrix decomposition checks are relaxed in these solutions. It is expected that the matrices will not pass the triangular decomposition at certain frequencies. The solution will proceed, and only a warning will be issued. The loads at the given frequency are collected from the load file and fed to GFBS for a forward backward substitution solution. If the decomposition failed, a zero vector will result.

For one uncoupled modal formulation the equations to be solved are:

$$\{\epsilon_i\} = \frac{P_i(\omega)}{-m_i\omega^2 + ib_i\omega + k_i} \quad (12)$$

With zero damping the uncoupled modal formulation may produce division by small numbers. This fact is noted and the solution proceeds.

4. Order Phase:

Except for the uncoupled modal approach it may be necessary to reorder the solutions from a frequency / load sort to a load / frequency sort.

4.61.8 Subroutines

Utility subroutines PRETAB,TAB,CALCV,SSG2B,SSG2A,SSG2C,CDCOMP,SCDCMP,CSPSDC,CXFBS,FACTØR,SDCØMP, DECØMP, SSG3A and GFBS are used. See subroutine descriptions, Section 3, for details.

4.61.8.1 Subroutine Name: FRRD1A

1. Entry Point: FRRD1A

2. Purpose: To assemble the user selected loads.

3. Calling Sequence: CALL FRRD1A (DLT,FRL,CASECC,DIT,PP,LUSETD,NFREQ,NLØAD,FRQSET,FØL)

DLT,FRL,CASECC,DIT,PP are GINØ file numbers of their respective data blocks - integer - input.

LUSETD - Length of p set - integer - input.

NFREQ - Number of frequencies in selected frequency set - integer - output.

FØL - the GINØ file number of the output frequency list - may be purged.

MODULE FUNCTIONAL DESCRIPTIONS

NLOAD - Number of loads (records in CASECC) selected - integer - output.

FRQSET - Set id of selected frequency set - integer - output.

4.61.8.2 Subroutine Name: FRRD1B

1. Entry Point: FRRD1B

2. Purpose: To reduce loads from the p to the d (or h) set.

3. Calling Sequence: CALL FRRD1B (PP, USETD, GMD, GØD, MULTI, SINGLE, ØMIT, MØDAL, PHIDH, PD, PS, PH, SCR1, SCR2, SCR3, SCR4)

PP, USETD, GMD, GØD, PHIDH, PD, PS, PH are GINØ file numbers of their respective data blocks - integer - input.

MULTI - $MULTI \geq 0$ implies m's are present - integer - input.

SINGLE - $SINGLE \geq 0$ implies s's are present - integer - input.

ØMIT - $ØMIT \geq 0$ implies o's are present - integer - input.

MØDAL - MØDAL = MØDA implies a modal formulation - BCD - input.

SCR1, ..., SCR4 - GINØ file numbers of 4 scratch files - integer - input.

4.61.8.3 Subroutine Name: FRRD1C

1. Entry Point: FRRD1C

2. Purpose: To form and decompose "left" hand side of the frequency equation.

3. Calling Sequence: CALL FRRD1C (FRL, FRQSET, MDD, BDD, KDD, I, ULL, LLL, SCR1, SCR2, SCR3, SCR4, IGØØD)

FRL, MDD, BDD, KDD, ULL, LLL, SCR1-4 are GINØ file numbers of their respective data blocks - integer - input.

FRQSET - Set id of selected frequency set - integer - output.

I - Current frequency counter - integer - input.

IGØØD - $IGØØD = 1$ implies a singular matrix - integer - output.

4.61.8.4 Subroutine Name: FRRD1D

1. Entry Point: FRRD1D

2. Purpose: To solve for displacements given decomposition factors and loads.

FUNCTIONAL MODULE FRRD (FREQUENCY RESPONSE - DISPLACEMENT APPROACH)

3. Calling Sequence: CALL FRRD1D (PD,ULL,LLL,SCR1,SCR2,UDVP,I,NLOAD,IGOOD,NFREQ)

PD,ULL,LLL,UDVP,SCR1,SCR2 are GINØ file numbers of their respective data blocks - integer - input.

I - Current frequency count - integer - input.

NLOAD - Number of loads - integer - input.

IGOOD - IGOOD = 1 implies a singular matrix - integer - input.

NFREQ - Total number of frequencies - integer - input.

4.61.8.5 Subroutine Name: FRRD1E

1. Entry Point: FRRD1E

2. Purpose: To reorder displacements if necessary.

3. Calling Sequence: CALL FRRD1E (UDVP,UDV,NLOAD,I)

UDVP - GINØ file number of displacements sorted by frequency/load - integer - input.

UDV - GINØ file number of displacements sorted by load/frequency - integer - input.

NLOAD - Number of loads - integer - input.

I - Number of frequencies solved.

4.61.8.6 Subroutine Name: FRRD1F

1. Entry Point: FRRD1F

2. Purpose: To solve the uncoupled modal equations.

3. Calling Sequence: CALL FRRD1F (MHH,BHH,KHH,FRL,FRQSET,NLOAD,NFREQ,PH,UHV)

MHH,BHH,KHH,FRL,PH,UHV are GINØ file numbers of their respective data blocks - integer - input.

FRQSET - Selected frequency set id-integer - input.

NFREQ - Number of frequencies in FRQSET - integer - input.

NLOAD - Number of loads (subcases in current execution) - integer - input.

4.61.8.7 Subroutine Name: FACTRU

1. Entry Point: FACTRU

2. Purpose: To decompose a matrix by invoking real unsymmetric decomposition $[A] \Rightarrow [LL][UL]$.

FUNCTIONAL MODULE FRRD (FREQUENCY RESPONSE - DISPLACEMENT APPROACH)

3. Calling Sequence: CALL FACTRU (\$n,A,LL,UL,SCR1,SCR2,SCR3)

where A,LL,UL,SCR1,SCR2,SCR3 are the GINØ file numbers of their respective data blocks.

n - statement number to return to if A is singular.

4. Design Requirements:

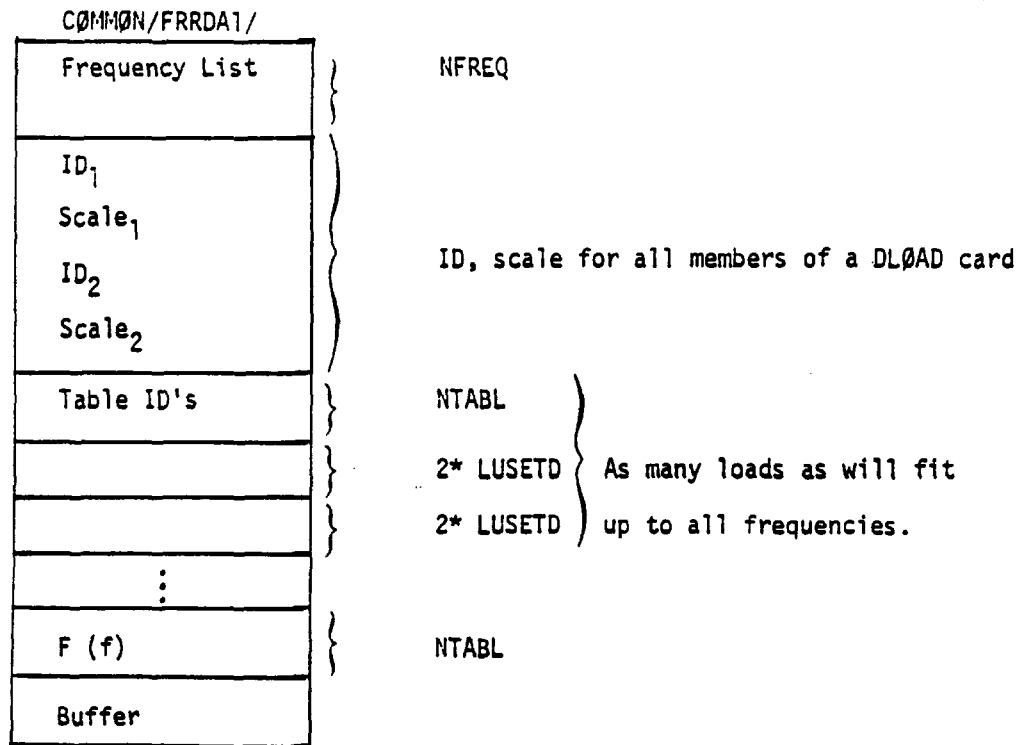
- a) A must have a trailer
- b) A trailer will be written on LL and UL
- c) Open core must be available at /FCTRUX/

4.61.9 Design Requirements

Eight scratch files are used by FRRD.

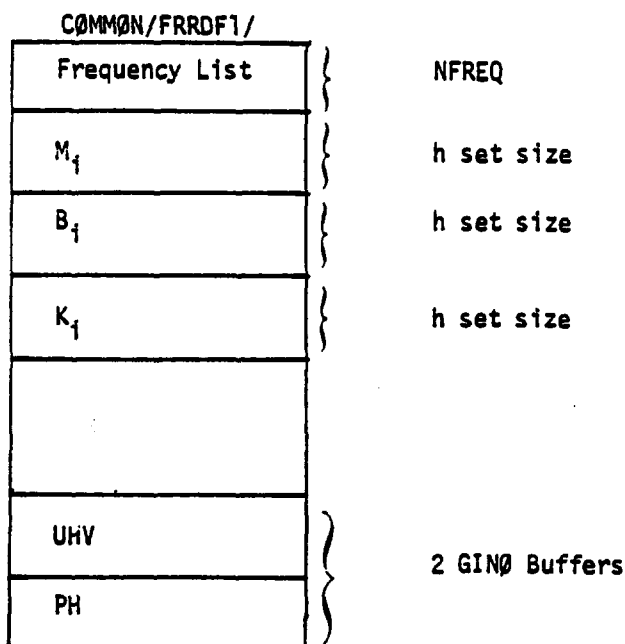
Open core at /FRRDA1/ is used as follows:

MODULE FUNCTIONAL DESCRIPTIONS



Open core at /FRRDB1/, /FRRDC1/, /FRRDD1/ are used by the matrix routines.

Open core at /FRRDF1/ is used as follows:



4.61.10 Diagnostic Messages

Module FRRD may issue the following diagnostic messages:

3005, 3008 and 3045.

FUNCTIONAL MODULE SDR3 (STRESS DATA RECOVERY - PHASE 3 - SØRT1 TO SØRT2 PROCESSOR)

4.62 FUNCTIONAL MODULE SDR3 (STRESS DATA RECOVERY - PHASE 3 - SØRT1 to SØRT2 PROCESSOR)

4.62.1 Entry Point: SDR3

4.62.2 Purpose

To transpose (perform SØRT2) data blocks containing data prepared for output in the form of ELEMENT-ID-SETS or PØINT-ID-SETS versus TIME-STEP or FREQUENCY-STEP to data prepared for output in the form of TIME-STEP-SETS or FREQUENCY-STEP-SETS versus ELEMENT-ID or PØINT-ID.

4.62.2.1 Example of SØRT1 and SØRT2 Output

Below is a table of ØFP printed output of SDR3 input (SØRT1) and output (SØRT2) data blocks.

SDR3 Input Data Block Printed (SØRT1)

TIME = 1.0		D I S P L A C E M E N T S					
POINT-ID		T1	T2	T3	R1	R2	R3
1		0.0	4.53	0.0	0.0	0.0	0.0
2		0.0	5.12	0.0	0.0	0.0	0.0
TIME = 2.0		D I S P L A C E M E N T S					
POINT-ID		T1	T2	T3	R1	R2	R3
1		0.0	4.83	0.0	0.0	0.0	0.0
2		0.0	5.53	0.0	0.0	0.0	0.0
TIME = 3.0		D I S P L A C E M E N T S					
POINT-ID		T1	T2	T3	R1	R2	R3
1		0.0	6.84	0.0	0.0	0.0	0.0
2		0.0	7.96	0.0	0.0	0.0	0.0

SDR3 Output Data Block Printed (SØRT2)

POINT-ID = 1		D I S P L A C E M E N T S					
TIME		T1	T2	T3	R1	R2	R3
1.0		0.0	4.53	0.0	0.0	0.0	0.0
2.0		0.0	4.83	0.0	0.0	0.0	0.0
3.0		0.0	6.84	0.0	0.0	0.0	0.0
POINT-ID = 2		D I S P L A C E M E N T S					
TIME		T1	T2	T3	R1	R2	R3
1.0		0.0	5.12	0.0	0.0	0.0	0.0
2.0		0.0	5.53	0.0	0.0	0.0	0.0
3.0		0.0	7.96	0.0	0.0	0.0	0.0

MODULE FUNCTIONAL DESCRIPTIONS

4.62.3 DMAP Calling Sequence

SDR3 IN1,IN2,IN3,IN4,IN5,IN6/OUT1,OUT2,OUT3,OUT4,OUT5,OUT6/ \$

4.62.4 Input Data Blocks

One to six data blocks in any order desired. Input data blocks to SDR3 which are purged are ignored.

4.62.5 Output Data Blocks

One to six data blocks in corresponding order to that of the input data blocks. If SORT2 is to be performed, there must be an available output data block for the corresponding input data block (Non-Fatal Error if this condition is not met).

4.62.6 Parameters

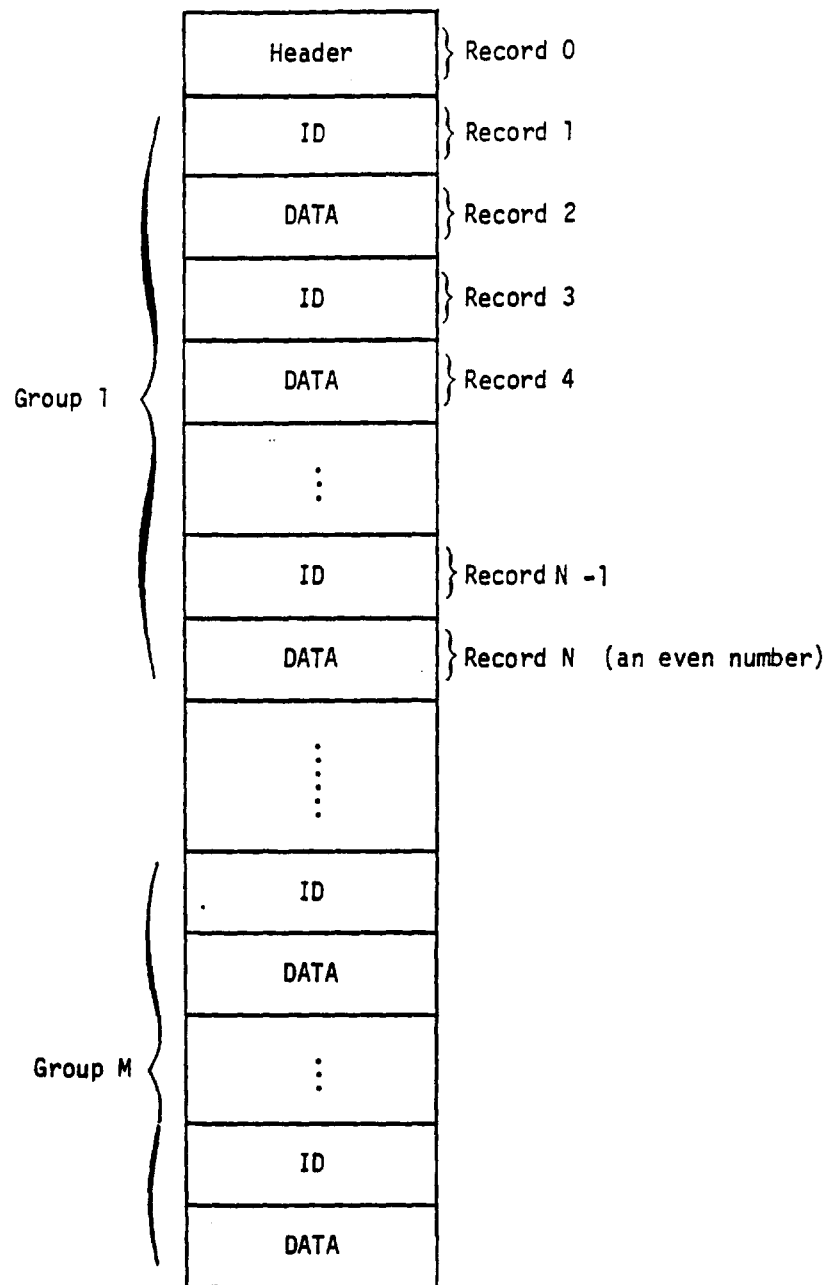
None

4.62.7 Method

4.62.7.1 Input and Output Data Block Record Arrangements

Both the input and output data blocks of SDR3 have the following format:

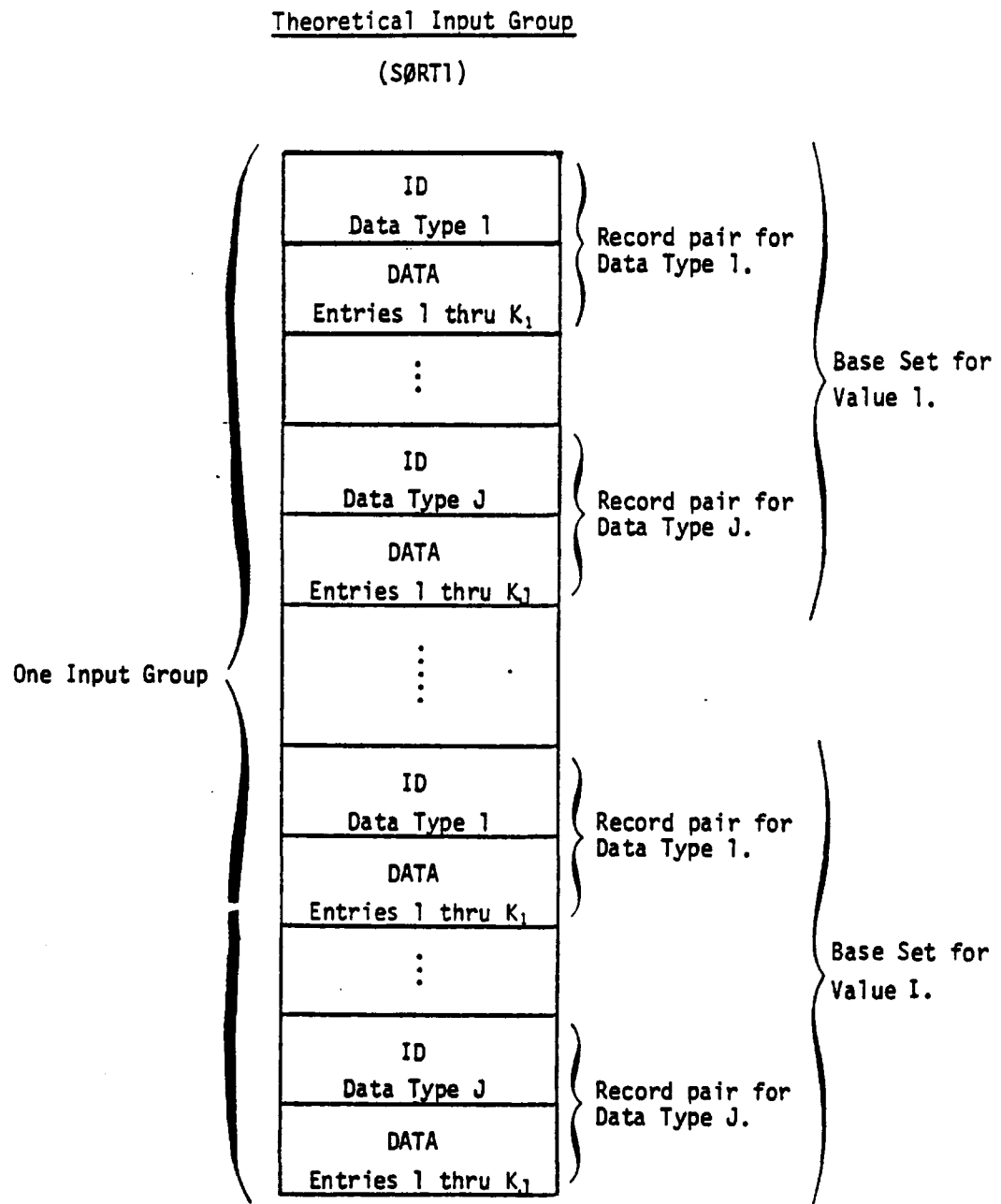
FUNCTIONAL MODULE SDR3 (STRESS DATA RECOVERY - PHASE 3 - SØRT1 TO SØRT2 PROCESSOR)



MODULE FUNCTIONAL DESCRIPTIONS

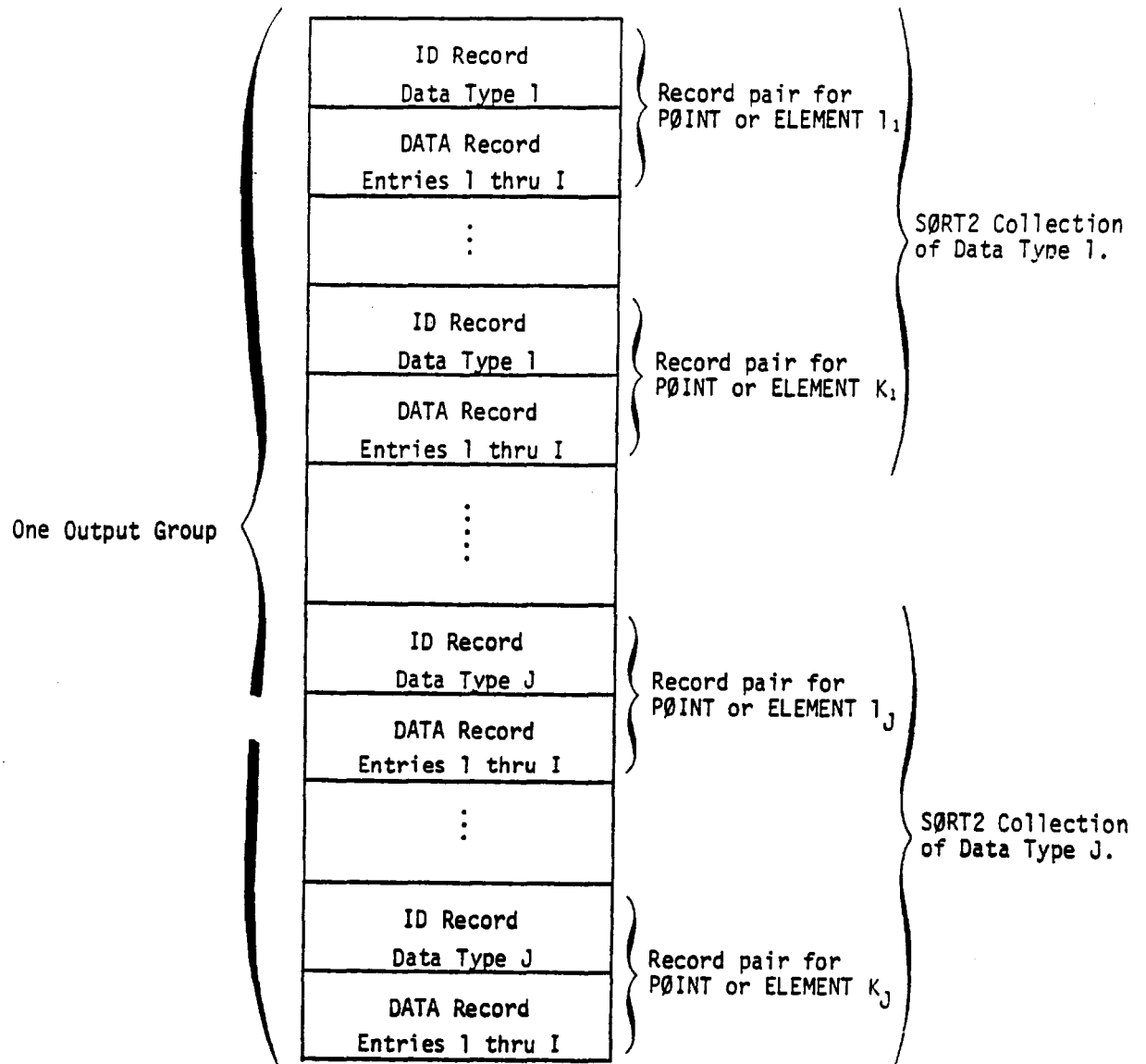
4.62.7.2 Description of a Group

1. An input (SØRT1) data block Group and an output (SØRT2) data block Group are given in the following figures:



Theoretical Output Group

(SØRT2)



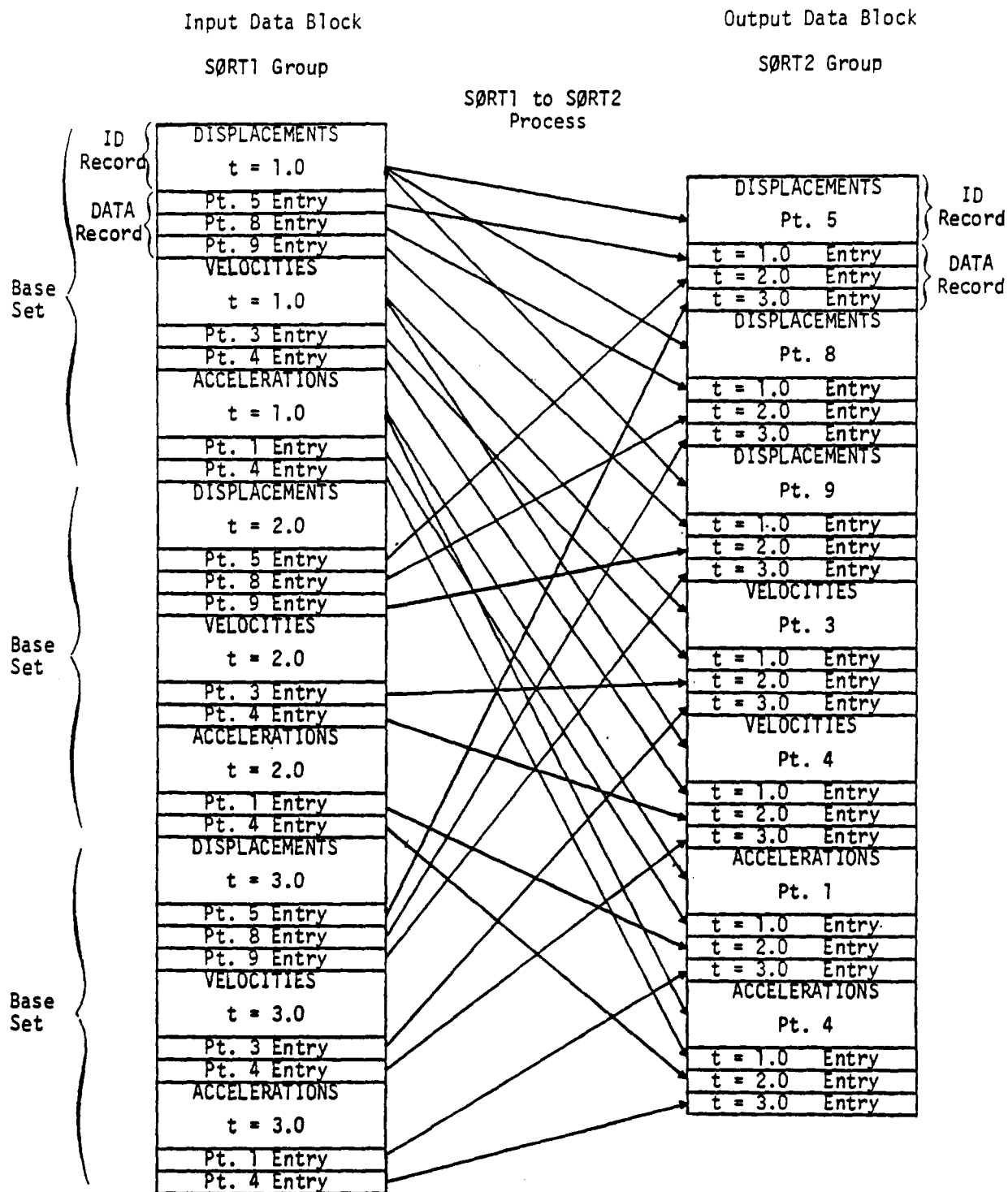
MODULE FUNCTIONAL DESCRIPTIONS

2. In the above figures each Group is independent of any other Group so far as SDR3 need be concerned.
3. A Group is defined as a collection of successive records belonging to the same subcase.
4. An ID-Record is of a fixed size equal to 146 words.
5. A DATA-Record contains multiple Entries with each Entry being of a length in words specified within the immediately preceding ID-Record.
6. I = The number of Values (FREQUENCIES or TIMES) present in the Group.
7. A Base Set is a sub-Group of the Group containing data records for one particular Value.
8. J = The number of different Data Types (DISPLACEMENTS, VELOCITIES, etc.) within a Base Set.
9. K_j = The number of Entries for Data Type j .
10. Respective records of any two Base Sets within an input data block Group are of the same size.
11. Respective Entries within respective DATA Records of all Base Sets of an input data block Group begin with the same ELEMENT-ID or POINT-ID.
12. Most input data blocks will contain only one Group having but one Data Type. There is normally more than one Base Set within any Group.
13. A pictorial representation of a SORT1 to SORT2 process is given on the next page using the following data:

Values = 3 time steps (1.0, 2.0, 3.0)

Data Types = $\begin{cases} 1 - \text{Displacements (3 Entries/Value - points 5, 8 and 9)} \\ 2 - \text{Velocities (2 Entries/Value - points 3 and 4)} \\ 3 - \text{Accelerations (2 Entries/Value - points 1 and 4)} \end{cases}$

FUNCTIONAL MODULE SDR3 (STRESS DATA RECOVERY - PHASE 3 - SØRT1 TO SØRT2 PROCESSOR)



MODULE FUNCTIONAL DESCRIPTIONS

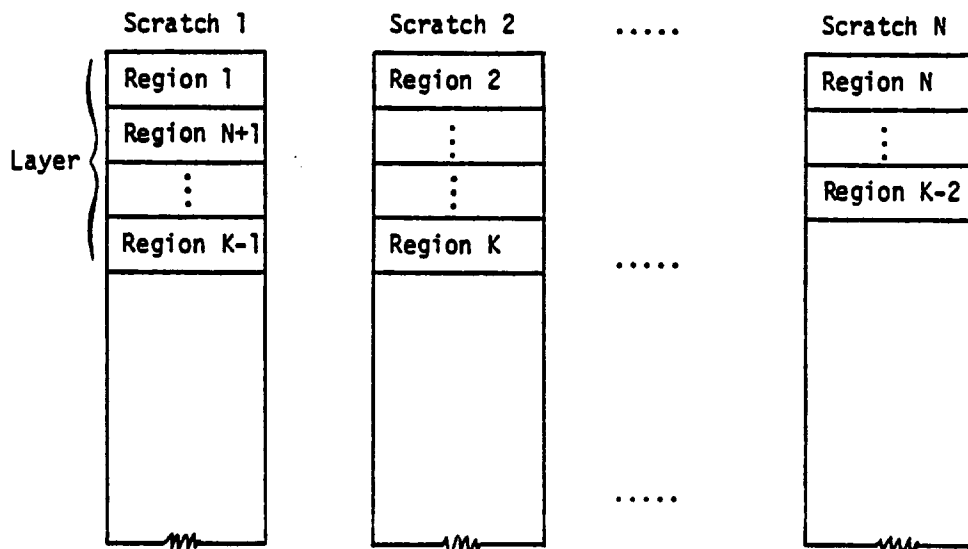
4.62.7.3 Physical Data Processing (SØRT1 to SØRT2)

All emphasis is placed on the Group, and thus in performing SØRT2 a Group pointer always points to the first record of the current Group being processed.

Each Group is processed and completed successively until all Groups have been processed. For each Group a loop of J passes is executed. During the j^{th} pass of this loop, the j^{th} Data Type (note 4.62.7.2) present of the Base Sets will be collected and transposed. The transpose consists of determining how many Entries are present for the current Data Type and then dividing the available core into that many Regions. The Entries of each DATA record for the j^{th} Data Type are distributed in Entry order, one each, to the Regions. At the time each Entry is distributed to a Region, the Entry's first word (PØINT-ID or ELEMENT-ID) is replaced by the Value (FREQUENCY or TIME) in the ID-Record associated with the DATA-Record from which the Entry has come. At the conclusion of each pass of this loop, output to the data block can proceed. For each Region an ID-Record is written. This ID-Record is a copy of the input data block ID-Record in the first Base Set for the j^{th} Data Type, having had the Value (FREQUENCY or TIME) replaced with the PØINT-ID or ELEMENT-ID of the respective Region. The filled portion of the Region is then output as the DATA-Record.

4.62.7.4 Spill Logic

If during the Entry distribution the Regions can hold no more Entries, spill to scratch files is performed. A Layer of records is written, one record for each Region, each time spill is required.



At the output stage, if spill to the scratch files has occurred, the Regions in the scratch files are output before the in-core Regions.

4.62.8 Subroutines

4.62.8.1 Subroutine Name: SDR3A

1. Entry Point: SDR3A
2. Purpose: To perform all SØRT2 operations when called by the driver routine SDR3.
3. Calling Sequence: CALL SDR3A (ØFPFIL)

ØFPFIL - An array of six words, one for each input data block, each of which is set to zero before the CALL and then reset by SDR3A with a traceback positive integer in the event an error for its respective data block occurred.

4.62.9 Design Requirements

1. The design requires that the largest DATA-Record fit in core. If a problem is outputting so many ELEMENT-ID or PØINT-ID Entries for a particular FREQUENCY or TIME that core is insufficient, then more subcases in conjunction with output request sets are recommended.

2. CØMMØN/SDR3ZZ/Z(1)

This common block defines open core for the SDR3 module.

3. SDR3 will open all its scratch files (8).

4.62.10 Diagnostic Messages

All errors within SDR3 are considered non-fatal-User Warning type errors. Any error resulting in termination of the SØRT2 process results in the setting of an SDR3 traceback number, an appropriate message, and a call to the ØFP (Output File Processor) which in turn will output the data block in SØRT1 format. If ØFP is unable to output the data block it in turn will call the TABPRT routine, and the data block will be printed.

FUNCTIONAL MODULE XYTRAN (XY - OUTPUT DATA TRANSLATOR)

4.63 FUNCTIONAL MODULE XYTRAN (XY - OUTPUT DATA TRANSLATOR)

4.63.1 Entry Point: XYTRAN

To read the first record of the XYCDB data block (prepared by subroutine IFPIXY of Executive module IFP1); to set xy-output parameters from the serial specifications of this record; to interpret the user curve requests; to locate in the XYTRAN input data blocks (2 thru 6) the data sets containing the requested curve data; to prepare summary and xy-coordinate data for the requested curves and output them to the system output printer and punch units; and to prepare xy-coordinate data and output them to the XYTRAN output data block for direct plotting by the XYPLØT module of those curve requests specified to be plotted.

4.63.3 DMAP Calling Sequences

4.63.3.1 Static Analysis (Rigid Format 1)

1. Stress data recovery output.

XYTRAN XYCDB,ØPG2,ØQG2,ØUGV2,ØES2,ØEF2/XYPLTT/C,N,TRAN/C,N,PSET/V,N,PFILE/V,N,CARDNØ \$

4.63.3.2 Transient Response - Direct Formulation. (Rigid Format 9)

1. Vector data recovery output.

XYTRAN XYCDB,ØUDVC2,,,/XYPLTFA/C,N,FREQ/C,N,DSET/V,N,PFILE/V,N,CARDNØ \$

2. Stress data recovery output.

XYTRAN XYCDB,ØPPC2,ØQPC2,ØUPVC2,ØESC2,ØEFC2/XYPLTF/C,N,FREQ/C,N,PSET/V,N,PFILE/V,N,CARDNØ \$

3. Random response output.

4.63.3.3 Transient Response - Direct Formulation. (Rigid Format 9)

1. Vector data recovery output.

XYTRAN XYCDB,ØUDV2,ØPNL2,,,/XYPLTTA/C,N,TRAN/C,N,DSET/V,N,PFILE/V,N,CARDNØ \$

2. Stress data recovery output.

XYTRAN XYCDB,ØPP2,ØQP2,ØUPV2,ØES2,ØEF2/XYPLTT/C,N,TRAN/C,N,PSET/V,N,PFILE/V,N,CARDNØ \$

4.63.3.4 Frequency Response - Modal Formulation. (Rigid Format 11)

1. Vector data recovery output.

XYTRAN XYCDB,ØUHVC2,,,/XYPLTFA/C,N,FREQ/C,N,HSET/V,N,PFILE/V,N,CARDNØ \$

2. Stress data recovery output.

XYTRAN XYCDB,ØPPC2,ØQPC2,ØUPVC2,ØESC2,ØEFC2 / XYPLTF / C,N,FREQ / C,N,PSET / V,N,PFILE /
V,N,CARDNØ \$

3. Random Response output

XYTRAN XYCDB,PSDF,AUTØ,,, / XYPLTR / C,N,RAND / C,N,PSET / V,N,PFILE / V,N,CARDNØ \$

4.63.3.5 Transient Response - Modal Formulation (Rigid Format 12)

1. Vector data recovery output.

XYTRAN XYCDB,ØUHV2,ØPNL2,,, / XYPLTTA / C,N,TRAN / C,N,HSET / V,N,PFILE / V,N,CARDNØ \$

4.63.3.6 Aerodynamic - Modal Flutter Analysis (Rigid Format 10)

1. VG curve output.

XYTRAN XYCBD,ØVG,,, / XYPLTCE / C,N,VG / C,N,PSET / V,N,PFILE / V,N,CARDNØ \$

4.63.3.7 Heat - Transient Analysis

1. XYTRAN XYCDB,HØPP2,HØQP2,HØUPV2,,HØEF2/HXYPLTT/C,N,TRAN/C,N,PSET/V,N,PFILE/V,N,CARDNØ \$

4.63.4 Input Data Blocks

XYCDB - XY Output Control Data Block.

ØUDVC2 - Output displacement vector requests (solution set, SØRT2, complex).

ØPPC2 - Output load vector requests (solution set, SØRT2, complex).

ØQPC2 - Output forces of single-point constraint requests (solution set, SØRT2, complex).

ØUPVC2 - Output displacement vector requests (p set, SØRT2, complex).

ØESC2 - Output element stress requests (SØRT2, complex).

ØEFC2 - Output element force requests (SØRT2, complex).

PSDF - Power Spectral Density Table.

AUTØ - Autocorrelation function table.

ØUDV2 - Output displacement vector requests (solution set, SØRT2, real).

ØPNL2 - Output nonlinear load requests (solution set, SORT2, real).

HØPP2 }
ØPP2 } - Output load vector requests (p set, SORT, real).

HØQP2 }
ØQP2 } - Output forces of single-point constraint (p set, SØRT2, real).

ØUPV2 - Output displacement vector requests (p set, SØRT2, real).

ØES2 - Output element stress requests (SØRT2, real).

HØEF2 }
ØEF2 } - Output element force requests (SØRT2, real).

ØUHV2 - Output displacement vector requests (solution set, SØRT2, complex).

FUNCTIONAL MODULE XYTRAN (XY - OUTPUT DATA TRANSLATOR)

- ØUHV2 - Output displacement vector requests (solution set, SØRT2, complex).
- ØVG - Output VG curves (SØRT2
- HØUPV2 - Output temperature vector requests (p set, SØRT2, real).
- ØPG2 - Output load vector requests (g set, SØRT2, real).
- ØQG2 - Output forces of single-point constraint requests (g set, SØRT2, real).
- ØUGV2 - Output displacement vector requests (SØRT2, real).

4.63.5 Output Data Blocks

- XYPLTFA -
 - XYPLTF -
 - XYPLTR -
 - XYPLTTA -
 - XYPLTT -
 - HXYPLTT -
- } XY-Plot output requests prepared by XYTRAN for direct plotting by XYPLØT.

4.63.6 Parameters

- CARDNØ - Input and output-integer-default value = 0. CARDNØ is incremented by one and punched in columns 73-80 of each card punched by XYTRAN.
- PFILE - Input and output-integer-default value = 0. PFILE is incremented by one for each frame XYTRAN defines for output by XYPLØT.
- FREQ - Input-BCD-2-word-constant distinguishes the problem as frequency response.
- TRAN - Input-BCD 2-word-constant distinguishes the problem as transient response.
- RAND - Input-BCD 2-word-constant distinguishes the problem as random response.
- DSET - Input-BCD 2-word-constant distinguishes the input vector as the d set.
- PSET - Input-BCD 2-word-constant distinguishes the input vector as the p set.
- HSET - Input-BCD 2-word-constant distinguishes the input vector as the h set.

MODULE FUNCTIONAL DESCRIPTIONS

4.63.7 Method

4.63.7.1 The following diagram illustrates the process of serially reading through the XYCDB data block's first record and performing the XYTRAN data processing.

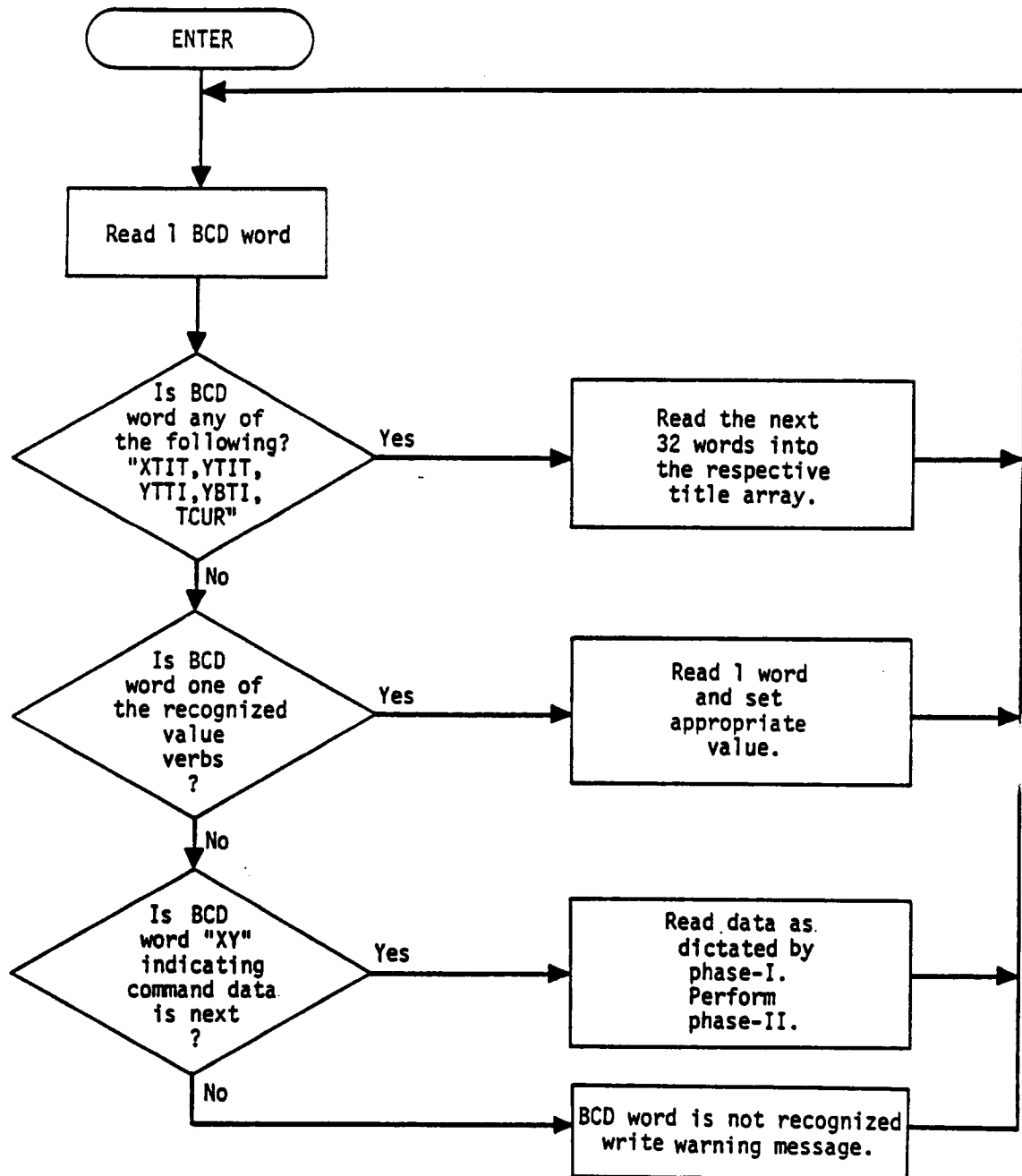


Figure 1. Flowchart for reading the first record of XYCDB

FUNCTIONAL MODULE XYTRAN (XY - OUTPUT DATA TRANSLATOR)

4.63.7.2 Phase I

In Phase I the XYCDB data block is further read to:

1. Determine the type of XY-output curves desired. (Response, Autocorrelation, or Power Spectral Density Function);
2. Determine the type of data (displacements, stresses, etc.,) and subcases desired;
3. Determine which types of XY-output are requested of XYPUNCH, XYPEAK, XYPRINT, XYPAPLOT, and XYPLT (XY-output requests are described in section 4 of the User's Manual).
4. Determine the point-component curve relationships for a frame.

The data for all curves of a given frame (upper and lower, or whole) are then collected and stored in core.

4.63.7.3 Phase II

The operations of Phase II involve the analysis of the curve data in conjunction with the XY-output specifications stored to this point as a set of values, and the computation and setting of dynamic curve limits. When all processing is complete, output to the printer, the punch, and the XYTRAN output data block is accomplished.

4.63.8 Subroutines

4.63.8.1 Subroutine Name: XYDUMP

1. Entry Point: XYDUMP
 2. Purpose: To perform phase II as described above.
 3. Calling Sequence: CALL XYDUMP (IARG, ITYPE).
- IARG - 201, GINØ output data block number.
- ITYPE - 1 for RESPONSE, 2 for PSDF, 3 for AUTØ.

4.63.8.2 Subroutine Name: XYFIND

1. Entry Point: XYFIND
2. Purpose: To position one of the XYTRAN input data blocks (2 thru 6) to the beginning of a data set record for a particular ELEMENT-ID or POINT-ID of a specific

MODULE FUNCTIONAL DESCRIPTIONS

data type.

3. Calling Sequence: CALL XYFIND (\$n₁, \$n₂, \$n₃, MAJID, IDZ)

n₁ = Return taken in the event an end-of-file is sensed when an EOF should not be hit.

n₂ = Return taken in the event an end-of-record is sensed when an end-of-record should not be hit.

n₃ = Return taken if the data requested could not be found.

MAJID = An array of the eleven data type major-IDs.

IDZ = Pointer into the Z array of open core to an ELEMENT-ID or POINT-ID.

4.63.8.3 Subroutine Name: XYØUT

1. Entry Point: XYØUT

2. Purpose: To output to the system printer unit an xy-output summary or to output to the system printer and/or punch unit(s) an xy-output coordinate pair.

3. Calling Sequence: CALL XYØUT (IARG, BUFF)

IARG = $\begin{cases} <0 \text{ implies print summary.} \\ \geq 0 \text{ implies print and/or punch coordinate pair.} \end{cases}$

BUFF = Array containing data to be output.

4.63.8.4 Subroutine Name: XYLØG

1. Entry Point: XYLØG

2. Purpose: To analyze the input arguments V1 and V2 and to reset these arguments to powers of ten bracketing the original values. An example follows.

V1	=	0.5	} Input arguments.
V2	=	5.6	
IARG	=	Undefined	
V1	=	0.1	} Output arguments.
V2	=	10.0	
IARG	=	2	

FUNCTIONAL MODULE XYTRAN (XY - OUTPUT DATA TRANSLATOR)

3. Calling Sequence: CALL XYLØG(V1,V2,IARG)

V1 = Smaller input real variable.

V2 = Large input real variable.

IARG = Number of logarithmic cycles needed to bracket V1 and V2. (Set by XYLØG before return)

4.63.8.5 Subroutine Name: XYTICS

1. Entry Point: XYTICS

2. Purpose: To accept user-specified xy-plot edge-tic specifications and compute actual edge-tic beginning and ending values, their increments to the successive edge-tics, and their scientific values with powers of ten.

3. Calling Sequence: CALL XYTICS (IØUT,ØUT,IARG1,R1,R2,ISKIP,LØG)

IØUT = Integer output array
ØUT = Real output array } One and the same array.

IARG1 = Number of edge-tic divisions desired by user.

R1 = Minimum coordinate value of edge.

R2 = Maximum coordinate value of edge.

ISKIP = Edge-tic skip count indicating which edge-tics are to have a value printed along with the tic-mark.

LØG = Number of logarithmic cycles. If zero, linear scale is to be calculated.

4.63.8.6 Subroutine Name: XYPRPL

1. Entry Point: XYPRPL

2. Purpose: To process the XYPAPLØT request. The XYTRAN output data block is read and a proper plot is generated for each XYPAPLØT request. Frame numbers are printed as well as titles, and the data are scaled to the size of the page width. Log requests should not be used.

3. Calling Sequence: CALL XYPRPL

4.63.8.7 Subroutine Name: XYCHAR

1. Entry Point: XYCHAR

2. Purpose: To store the points to be plotted into the appropriate line of the output buffer.

MODULE FUNCTIONAL DESCRIPTIONS

3. Calling Sequence: CALL XYCHAR(IRØW,ICØL,CURVCH)

IRØW = Y coordinate of the point to be plotted

ICØL = X coordinate of the point to be plotted

CURVCH = Symbol to be used for the point

4.63.8.8 Subroutine Name: XYGRAF

1. Entry Point: XYGRAF

2. Purpose: To print the proper plot for a frame.

3. Calling Sequence: CALL XYGRAF(GRAPH)

GRAPH = Frame border data for the plot

4.63.9 Design Requirements

1. The XYTRAN design requires that for a particular frame all of the curve data for the curves of that frame fit in core. If this condition is not possible, one curve at a time will be cancelled, with a warning message output, until the condition is met for the frame in question.

2. The following CØMMØN blocks are used in the subroutines of module XYTRAN.

a. CØMMØN/XYWØRK/

This common block contains variables required in the processing of the user output requests.

b. CØMMØN/XYTRZZ/

Defines open core for the module

4.63.10 Diagnostic Messages

All XYTRAN diagnostic messages are of a USER-WARNING nature. There are no FATAL type error diagnostics. XYTRAN is in all cases expected to make a normal return.

FUNCTIONAL MODULE RANDØM (RANDOM ANALYSIS MODULE)

4.64 FUNCTIONAL MODULE RANDØM (RANDOM ANALYSIS MODULE)

4.64.1 Entry Point: RANDØM

4.64.2 Purpose

To compute power spectral density functions and autocorrelation functions from frequency response data.

4.64.3 DMAP Calling Sequence

RANDØM XYCDB,DIT,PSDL,ØUPVC2,ØPPC2,ØQPC2,ØESC2,ØEFC2,CASECC/PSDF,AUTØ/V,N,NØRAND \$

4.64.4 Input Data Blocks

XYCDB - XY Plotter Control Data Block.
DIT - Direct Input Tables.
PSDL - Power Spectral Density List.
ØUPVC2 - Output displacement vector requests (p set, SØRT2, complex).
ØPPC2 - Output load vector requests (p set, SØRT2, complex).
ØQPC2 - Output forces of single-point constraint (p set, SØRT2, complex).
ØESC2 - Output element stress requests (SØRT2, complex).
ØEFC2 - Output element force requests (SØRT2, complex).
CASECC - Case Control Data Table.

Notes: 1. If XYCDB is purged, RANDØM returns.
2. DIT cannot be purged if PSDL points to tables in DIT.
3. If PSDL is purged, RANDØM returns.
4. ØUPVC2, ØPP2, ØQP2, ØESC2, ØEFC2 must contain the requested outputs.
5. CASECC cannot be purged.

4.64.5 Output Data Blocks

PSDF - Power Spectral Density Table.
AUTØ - Autocorrelation function table.

Notes: PSDF and AUTØ cannot be purged.

MODULE FUNCTIONAL DESCRIPTIONS

4.64.6 Parameters

NØRAND - Output-integer-no default. NØRAND = -1, if no random analysis is requested;
0, otherwise.

4.64.7 Method

4.64.7.1 Overview of the Method

The Random Analysis Module calculates power spectral density functions, autocorrelation functions and mean deviations for selected displacements, loads, forces of single-point constraint, and element forces and stresses.

4.64.7.2 Module Initialization

The following 4 steps of subroutine RAND7 comprise module initialization.

1. The XYCDB must be present or RANDØM returns.
2. A set of RANDPS Bulk Data cards from PSDL must be selected in CASECC or RANDØM returns.
3. The frequency list is extracted from the first non-empty data file.
4. The selected RANDPS cards are read in and stored. The tables referenced are prepared by subroutine PRETAB. The RANDPS (see section 2.4 of the User's Manual) card defines the functions

$$S_{ab}(f) = (x + iy) F_K(f) \quad (1)$$

where a is the subcase id of the excited load set; b is the subcase id of the applied load set ($a \leq b$); (x,y) is a complex number such that if $a = b$, then y must be 0.0; and K is the table identification number of a TABRND1 Bulk Data card which defines $F_K(f)$, a power spectral density as a tabular function of frequency.

The power spectral density for gust turbulence can also be supplied on a TABRNDG card. Thus

$$S_{ab}(f) = \overline{W_g^2} \frac{2L}{U} \frac{1 + 2(p+1)(kW/U)^2}{[1 + (kW/U)^2]^{p+3/2}}$$

where $\overline{W_g^2}$, L, U, p and k are user supplied data and $W = 2\pi f$.

If on any RANDPS card $a \neq b$, the equations are called coupled, otherwise they are called uncoupled.

4.64.7.3 The Uncoupled Case

The following eight steps are accomplished in subroutine RAND5.

1. The XYCDB is read for a list of requested points. This list is stored in core. (Subroutine RAND6).
2. Core is allocated for as many points as possible at one word per frequency. If all points will not fit in core, another pass will be made on this file.
3. Compute $S_{aa}(f)$ at each load change (subroutine TAB).
4. Read in the data from the SØRT2 data block and compute:

$$S_{ja}(f) = |U_j(f)|^2 S_{aa}(f) , \quad (3)$$

where $U_j(f)$ is the response of the j^{th} point at frequency f .

5. These are summed over all loads to form the power spectral density function:

$$S_j(f) = \sum_a S_{ja}(f) , \quad (4)$$

where 'a' runs over all subcase ID's on the RANDPS cards.

6. When all subcases for the points in core have been processed, the mean response \bar{q}_j is calculated in subroutine RAND3 for each point j :

$$\bar{q}_j = \left\{ \frac{1}{2} \sum_{i=1}^{N-1} [(S_j(f_i) + S_j(f_{i+1})) (f_{i+1} - f_i)] \right\}^{1/2} , \quad (5)$$

where N = number of frequencies. The mean response is output with both the PSDF and the autocorrelation function.

The zero crossing N_0 is also computed and output with the mean response \bar{q}_j . N_0 is defined by

$$N_0 = \frac{1}{2\pi} \left[\int_0^\infty \omega^2 S_j(\omega) d\omega / \int_0^\infty S_j(\omega) d\omega \right] . \quad (6)$$

The integral in the denominator is already calculated in this module, and is related to the "mean square response"

$$\overline{q_j^2} = R_j(0) = \int_0^\infty S_j(f) df ; \quad (7)$$

thus, the numerator must be integrated. Compute

$$\begin{aligned} \overline{r_j^2} &= \int_0^\infty f^2 S_j(f) df , \\ \overline{r_j} &= \left\{ \frac{1}{2} \sum_{i=1}^{N-1} [S_j(f_i) + S_j(f_{i+1})) (f_{i+1} - f_i)] \right\}^{1/2} , \end{aligned} \quad \left. \vphantom{\int_0^\infty} \right\} \quad (8a)$$

1-61

FUNCTIONAL MODULE RANDOM (RANDOM ANALYSIS MODULE)

$$\left. \begin{aligned} \alpha &= (3f_i^2 + 2f_i f_{i+1} + f_{i+1}^2)/6 \\ \beta &= (f_i^2 + 2f_i f_{i+1} + 3f_{i+1}^2)/6 \end{aligned} \right\} \quad (8b)$$

Note that if α and β are 1.0, the sum for $\overline{r_j}$ would become the formula for $\overline{q_j}$. Then $N_0 = r_j/q_j$ is the quantity to be output.

7. If PSDF for point j is requested, one ID and data record are written on the PSDF data block.
8. If an autocorrelation function is requested for point j , the $S_j(f)$ are transformed to the time domain to give the autocorrelation function:

$$\begin{aligned} R_j(\tau_m) &= \sum_{i=1}^{N-1} \left\{ \frac{1}{4\pi^2 \tau_m^2} \left[\frac{S_j(f_{i+1}) - S_j(f_i)}{(f_{i+1} - f_i)} \right] [\cos(2\pi \tau_m f_{i+1}) - \cos(2\pi \tau_m f_i)] + \right. \\ &\quad \left. = \frac{1}{2\pi \tau_m} [S_j(f_{i+1}) \sin(2\pi \tau_m f_{i+1}) - S_j(f_i) \sin(2\pi \tau_m f_i)] \right\} . \end{aligned} \quad (9)$$

FUNCTIONAL MODULE RANDØM (RANDOM ANALYSIS MODULE)

THIS PAGE HAS BEEN LEFT BLANK INTENTIONALLY.

MODULE FUNCTIONAL DESCRIPTIONS

where i is the index of the frequencies; N is highest frequency; τ_m is defined by

$$\tau_m = \tau_0 + \frac{m}{M} (\tau_{\max} - \tau_0), \quad (10)$$

where τ_0 is the starting time lag, M is the number of time lag intervals, and τ_{\max} is the maximum time lag ($0 < \tau_0 < \tau_m$), all of which are defined on a RANDT1 Bulk Data card. Note that if, in Equation 9, $\tau_m = 0$, then

$$R_j(\tau_m) = \frac{1}{q_j^2}. \quad (11)$$

If more points for this data block remain to be done, the file is rewound and another pass is made. If additional file types are requested, steps 1 through 8 outlined above are repeated. This completes the uncoupled case processing.

4.64.7.4 The Coupled Case

The following 6 steps are accomplished in subroutine RAND8.

1. A list of unique subcase id's is extracted from the RANDPS cards.
2. The XYCDB is read for a list of requested points. This list is stored in core (subroutine RAND6).
3. An array of core is reserved for each point as follows:
Let NFREQ = the number of frequencies used and NUNØ be the number of unique subcase id's mentioned on the RANDPS cards. Each point requires $2 \text{ NFREQ} * \text{NUNØ}$ words of storage.
As many points as possible are done at once. The data file is read and the data are stored (real/imaginary) for each point until all subcases for all points in core have been processed.
4. For each RANDPS card $S_{ab}(f_i)$ is looked up for all f (subroutine TAB).
For each point in core $S_j^1(f)$ is computed:

$$S_j^1(f) = H_{ja}(f) S_{ab}(f) \bar{H}_{jb}(f), \quad (12)$$

where $H_{ja}(f)$ denotes the value of point j for subcase a . The bar over the third factor in Equation 12 denotes the complex conjugate. These $S_j^1(f)$ are summed over all RANDPS cards to form $S_j(f)$:

$$S_j(f) = \left| \sum_{ab} H_{ja}(f) S_{ab}(f) \overline{H}_{jb}(f) \right|.$$

(13)

Note that $S_{ba} = \overline{S_{ab}}$, the complex conjugate.

5. The mean response and autocorrelation functions are computed as in Equations 9, 10, and 11.

6. If more points for this file remain to be done, the file is rewound and another pass is made.

If additional file types are requested, steps 1 thru 6 are repeated. If not, the coupled case processing is complete.

4.64.8 Subroutines

4.64.8.1 Subroutine Name: RAND7

1. Entry Point: RAND7
2. Purpose: To initialize for both the coupled and uncoupled cases.
3. Calling Sequence: CALL RAND7(IFILE,NFILE,PSDL,DIT,ICØUP,NFREQ,NPSDL,NTAU,LTAB,CASECC,XYCDB).

PSDL,DIT,CASECC,XYCDB are GINØ file numbers for their respective data blocks -

- integer - input.

IFILE - Array of GINØ file numbers of data files to RANDØM - integer - input.

NFILE - Number of files in IFILE - integer - input.

ICØUP - -1 No RANDØM analysis to be done.

- 0 uncoupled algorithm to be used - integer - output.

- 1 coupled algorithm to be used.

NFREQ - Number of frequencies - integer - output.

NPSDL - Number of RANDPS cards selected - integer - output.

NTAU - Number of τ 's on RANDT1 cards - integer - output.

LTAB - Amount of core taken up by table storage - integer - output.

CØMMØN/RANDMX/

RAND7 stores most of its output data in /RANDMX/. See core storage layout of /RANDMX/ (section 4.64.9).

MODULE FUNCTIONAL DESCRIPTIONS

4.64.8.2 Subroutine Name: RAND5

1. Entry Point: RAND5
2. Purpose: To compute uncoupled PSDF and AUTØ numbers.
3. Calling Sequence: CALL RAND5(NFREQ,NPSDL,NTAU,XYCDB,LTAB,IFILE,PSDF,AUTØ,NFILE)
PSDF,AUTØ - GINØ file numbers of respective files - integer -input.
Other variables are as in RAND7 (Section 4.64.8.1).

4.64.8.3 Subroutine Name: RAND8

1. Entry Point: RAND8
2. Purpose: To compute coupled PSDF and AUTØ numbers.
3. Calling Sequence: CALL RAND8 (Same as RAND5).

4.64.8.4 Subroutine Name: RAND1

1. Entry Point: RAND1
2. Purpose: To put one ØFP type ID on PSDF and AUTØ.
3. Calling sequence: CALL RAND1 (FILE,MID,TYPE,ID,CØMP,Q).
FILE - GINØ file number of output file - integer - input.
MID - File type (PSDF = 4001,AUTØ = 4002) - integer - input.
TYPE - Curve type - DISP,VELØ,ACCE,LØAD,SPLF,ELFØ, or STRE - BCD, input.
ID - Point id - integer - input.
CØMP - Point component - integer - input.
Q - Mean deviation - real - input.

4.64.8.5 Subroutine Name: RAND2

1. Entry Points: RAND2, RAND2A
2. Purpose: To read a SØRT2 type output file until it finds a point id selected by the user in a list.
3. Calling Sequence: CALL RAND2 (FILE,ILIST,LØAD,IF,LEN,LLIST, DATA)
CALL RAND2A (DATA)
FILE - GINØ file number of the SØRT2 data file - integer - input.
ILIST - List of user desired points - input and output.

FUNCTIONAL MODULE RANDØM (RANDOM ANALYSIS MODULE)

LØAD - Subcase id of first data record in ILIST - integer - output.
IF - Format of data - real/imaginary or magnitude/phase - integer - output.
LEN - Length of the data line for this record - integer - output.
LLIST - Length of the ILIST array.
DATA - Data array - input.

4.64.8.6 Subroutine Name: RAND3

1. Entry Point: RAND3
2. Purpose: To compute the mean response \bar{q} .
3. Calling Sequence: CALL RAND3 (F,S,Q,N)
F - Array of frequencies - real - input.
S - Array of power spectral density functions - real - input.
Q(1) - Mean response - real - output.
Q(2) - Number of zero crossings - real - output.
N - Length of the F and S arrays - integer - input.

4.64.8.7 Subroutine Name: RAND4

1. Entry Point: RAND4
2. Purpose: To compute the autocorrelation function $R(\tau)$.
3. Calling Sequence: CALL RAND4 (F,S,TAU,R,N)
F,S,N are as described in RAND3.
TAU - τ point at which R is to computed - real - input.
R - Autocorrelation function at TAU - real - output.

4.64.8.8 Subroutine Name: RAND6

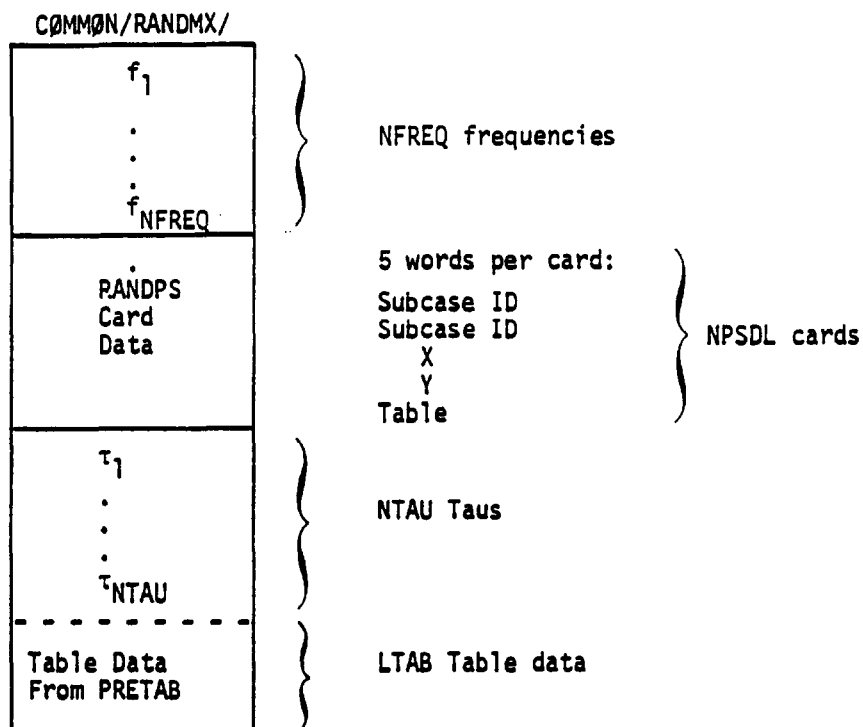
1. Entry Point: RAND6
2. Purpose: To extract from the XYCDB a list of user requested points for RANDØM output.
3. Calling Sequence: CALL RAND6 (XYCDB,BUFFER,NPØINT,IZ,INPUT)
XYCDB - GINØ file number of the XYCDB data block - integer - input.
BUFFER - GINØ buffer - array - input.
NPØINT - Number of points requested by the user for this file - integer - output.

MODULE FUNCTIONAL DESCRIPTIONS

IZ - Array in which RAND6 stores the list of requests - integer - output.
 INPUT - GINØ file number of data file for which list of request is desired.

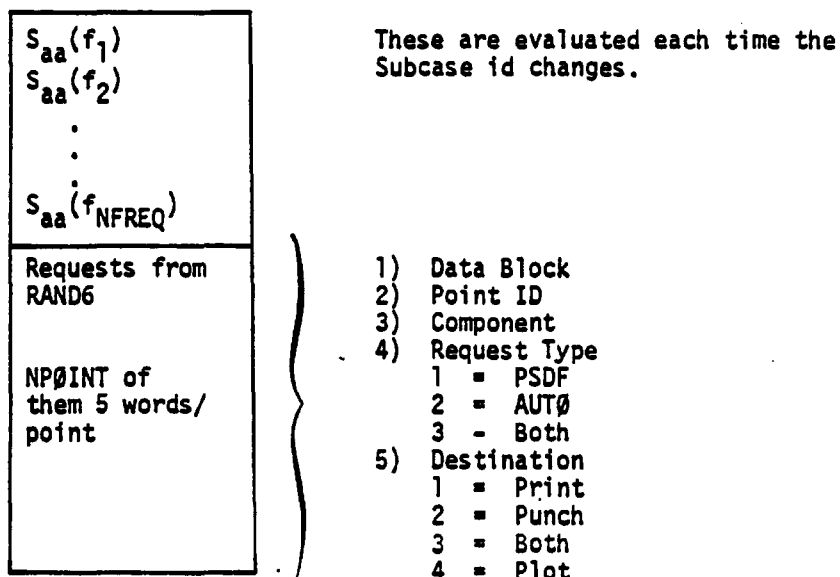
4.64.9 Design Requirements

Open Core at /RANDMX/ is arranged as follows:

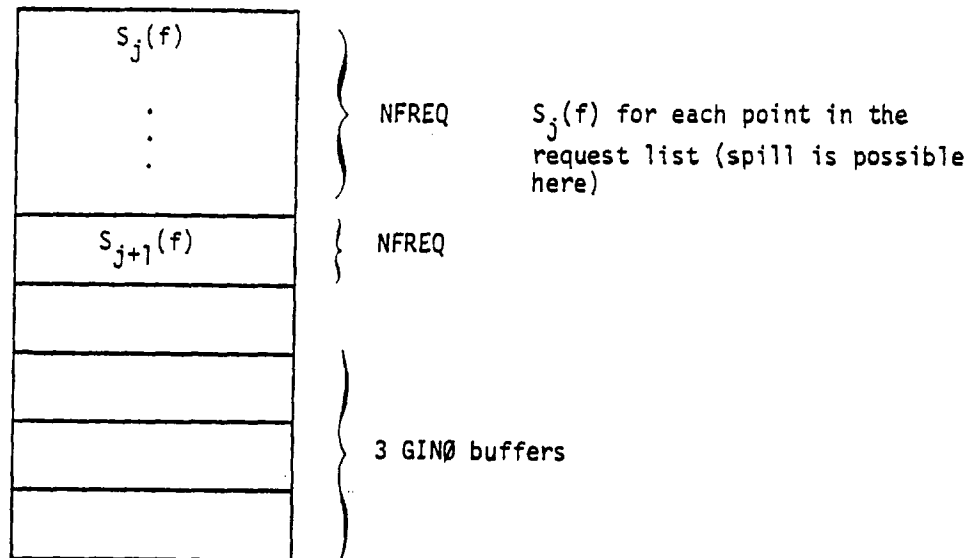


The above data are placed in core by RAND7 and are the same for both the coupled and uncoupled cases. The remaining data are core dependent.

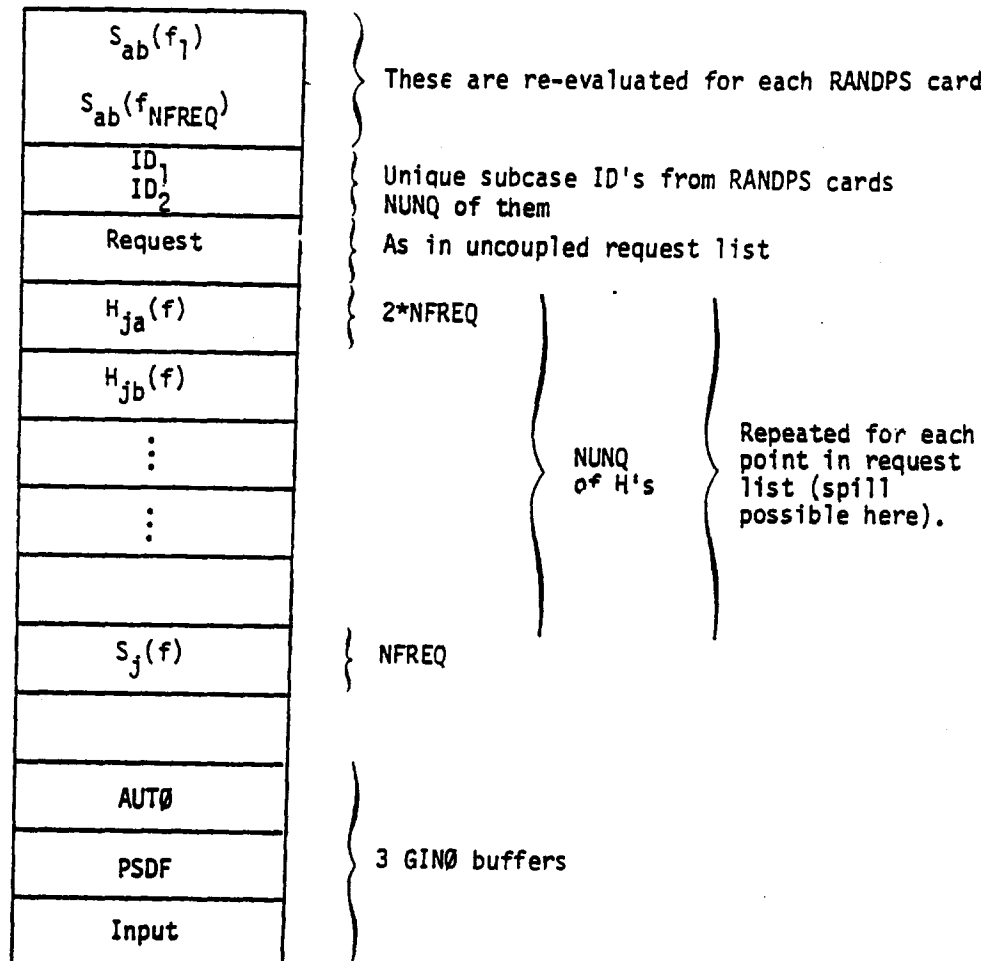
Uncoupled case data:



FUNCTIONAL MODULE RANDØM (RANDØM ANALYSIS MODULE)



Coupled case data:



MODULE FUNCTIONAL DESCRIPTIONS

4.64.10 Diagnostic Messages

RANDØM is defined as an output processor and thus must not stop due to user input error. Hence all messages are of a warning nature.

Random may issue message 3048.

FUNCTIONAL MODULE TRD (TRANSIENT ANALYSIS - DISPLACEMENT)

4.65 FUNCTIONAL MODULE TRD (TRANSIENT ANALYSIS - DISPLACEMENT)

4.65.1 Entry Point: TRD

4.65.2 Purpose

To solve the transient problem.

4.65.3 DMAP Calling Sequence

TRD CASECC,TRL,NLFT,DIT,{KDD},{BDD},{MDD},{PD},{UDVT},{PNLD}
{KHH},{BHH},{MHH},{PH},{UHV}, {PNLH} / V,N,FORM / V,N,NØUE /

V,N,NØNCUP/V,N,NCØL/C,Y,ISTART \$

4.65.4 Input Data Blocks

CASECC - Case Control Data Table.
TRL - Transient Response List.
NLFT - Non-linear Forcing Table.
DIT - Direct Input Tables.
KHH - Modal stiffness matrix - h set.
KDD - Dynamic stiffness matrix - d set.
BHH - Modal damping matrix - h set.
BDD - Dynamic damping matrix - d set.
MHH - Modal mass matrix - h set.
MDD - Dynamic mass matrix - d set.
PH - Transient Load Matrix.
PD - Linear dynamic load matrix for transient analysis - d set.

Notes:

1. CASECC cannot be purged.
2. TRL cannot be purged.
3. NLFT cannot be purged if nonlinear loads are selected in CASECC.
4. AT least one of the matrices KHH, BHH, or MHH must exist.

MODULE FUNCTIONAL DESCRIPTIONS

4.65.4 Output Data Blocks

- UHVT - Modal transient solution vectors - h set.
- PNLH - Nonlinear loads in modal transient problem - h set.
- UDVT - Displacement, velocity, and acceleration vector matrix in a transient analysis problem - d set.
- PNLD - Non-linear loads in a transient problem - d set.

Notes:

1. UHVT cannot be purged.
2. PNLH cannot be purged if nonlinear loads are selected.
3. UHVT will be read if it is not empty and the continue mode will be entered.
4. KHH, MHH, BHH, PH, UHVT, and PNLH fields will contain KDD, MDD, etc., for the direct solution.

4.65.6 Parameters

- FØRM - Input-BCD-no default. If FØRM = MØDAL a modal formulation will be used, otherwise a direct formulation will occur.
- NØUE - Input-integer-no default. NØUE indicates the number of extra points used in non-linear load formulation.
- NØNCUP - Input-integer-no default. If NØNCUP = -1 an uncoupled solution will be done.
- NCØL - Input/Output-integer-no default. If NCØL = 0, the initial time for the solution is 0.0. If NCØL > 0, the solution is continued from the specified output time of the previously checkpointed run. (See Section 11.3.2 of the Theoretical Manual for details.)
- ISTART - Input-integer-default = -1. If ISTART < 0, the first starting method is used. If ISTART \geq 0, the second (or alternate) starting method is used. (See Section 11.3.1 of the Theoretical Manual for details.)

4.65.7 Method

4.65.7.1 Overview of the Method

The Transient Analysis module integrates, over specified time periods, equations of motion of a structure having time dependent loads. A general structure may be used with real stiffness, mass and damping matrices. Non-linear effects may be calculated by specifying certain loading functions on the free, physical displacements of the system. This analysis is particularly useful when shock loads are applied to a structure. It is also more efficient than frequency analysis or

FUNCTIONAL MODULE TRD (TRANSIENT ANALYSIS - DISPLACEMENT)

complex eigenvalue analysis when the applied loads are well defined and the frequency characteristics are secondary to damping and peak load characteristics. This analysis is also the only dynamic general system analysis which allows non-linearities.

TRD will also continue the solution from some previous run. This can be used to recover from time-to-go failures or to extend the analysis. (See Section 11.3.2 of the Theoretical Manual for details.)

4.65.7.2 Logical Phases of Solution

1. The time increment from the TRL data block is used to identify the times at which the solution is obtained. The initial conditions are assembled.
2. The left hand matrix of the general integration equation, Equation 15 below, and the two right hand matrices are assembled. The triangular decomposition of the left hand matrix is performed.
3. The solution loop of the program may now proceed until the time increment is changed.
 - a. Compute the non-linear load for this time step. Add this load to the load vectors.
 - b. Multiply the displacement vectors into the right hand matrices and add the resultant vectors to the applied load vector.
 - c. Solve for the left hand displacement vector by performing a back substitution into the triangular decomposition of the left hand matrix. If this is an output time step, the velocity and acceleration are computed using differences of the displacement vectors.
 - d. If the time increment changes for the next time step, the program returns to Step 2. If the increment is the same steps 3a thru 3d are repeated.
4. If the equations are in the uncoupled modal formulation form (i.e., no transfer functions, direct input matrices, or non-linear functions), the solution logic is much faster. For each coordinate, the displacement, velocity and acceleration may be computed independently versus time. Steps 2 and 3 are omitted.

MODULE FUNCTIONAL DESCRIPTIONS

4.65.7.3 Algorithms for Each Logical Phase

1. Solution of the coupled equations: The matrix

$$[D] = \left(\frac{1}{\Delta t^2} [M] + \frac{1}{2\Delta t} [B] + \frac{1}{3} [K] \right), \quad (1)$$

is formed and decomposed. The matrix

$$[C] = \left(\frac{2}{\Delta t^2} [M] - \frac{1}{3} [K] \right), \quad (2)$$

is formed and saved. The matrix

$$[E] = \left[\frac{-1}{\Delta t^2} [M] + \frac{1}{2\Delta t} [B] - \frac{1}{3} [K] \right], \quad (3)$$

is formed and saved.

The solution loops then proceed until a time step change occurs.

The initial conditions presented to the integration are $\{u_0\}$, $\{\dot{u}_0\}$, $\{u_{-1}\}$, $\{P_0\}$ and $\{P_{-1}\}$, where $\{u_0\}$ and $\{\dot{u}_0\}$ are the starting displacement and velocity vectors, respectively, specified by the user. Two alternative starting methods have been provided, each having its own advantages. (See Section 11.3.1 of the Theoretical Manual for details.) In both the methods, $\{u_{-1}\}$ and $\{P_{-1}\}$ are calculated by the equations

$$\{u_{-1}\} = \{u_0\} - \{\dot{u}_0\} \Delta t, \quad (4)$$

and

$$\{P_{-1}\} = [K]\{u_{-1}\} + [B]\{\dot{u}_0\}. \quad (5)$$

The difference between the two starting methods lies in the different manner in which $\{P_0\}$ is computed. In the first method ($ISTART < 0$), the load specified by the user at $t = 0$ is never used but is replaced by

$$\{P_0\} = [K]\{u_0\} + [B]\{\dot{u}_0\}. \quad (6)$$

FUNCTIONAL MODULE TRD (TRANSIENT ANALYSIS - DISPLACEMENT)

The starting equation for this method is

$$[D]\{u_1\} = \frac{1}{3}\{P_{-1} + P_0 + P_1\} + \{N_0\} + [C]\{u_0\} + [E]\{\dot{u}_0\} \quad , \quad (7)$$

where $\{N_0\}$ is the non-linear load calculated from $\{u_0\}$.

In the second (or alternative) starting method ($ISTART \geq 0$), the user input load at $r = 0$ is included. Thus

$$\{P_0\} = \frac{1}{2} \{P_{ou}\} + [K]\{u_0\} + [B] \{\dot{u}_0\} \quad , \quad (8)$$

where $\{P_{ou}\}$ is the user-defined load at $r = 0$.

The starting equation for this method is

$$[D]\{u_1\} = \frac{1}{6}\{P_{ou} + 2P_1\} + \{N_0\} - \frac{1}{6} [K]\{u_0\} + \frac{1}{2\Delta t} [B]\{u_0\} + \frac{1}{\Delta t^2} [M]\{u_0 + \dot{u}_0\Delta t\} \quad (9)$$

$\{u_2\}$ through $\{u_n\}$ are now computed from the general equation:

$$\begin{aligned} [D] \{u_{i+2}\} &= \frac{1}{3} \{P_i + P_{i+1} + P_{i+2}\} + \\ &\{N_{i+1}\} + [C] \{u_{i+1}\} + [E] \{\dot{u}_i\}. \end{aligned} \quad (10)$$

If non-linear loads are selected, they are evaluated directly at the solution points for time step by the following process. NØLIN1 loads are computed as,

$$P_i(t) = S T (u_j(t)), \quad (11)$$

where T is a user selected table, i is the loaded solution point, j is the deflecting point, u_j is the previously computed displacement at point j . NØLIN2 loads are computed as,

$$P_i(t) = S u_j(t) u_k(t), \quad (12)$$

where i , j , and k are as in NØLIN1 loads.

NØLIN3 loads are computed as,

MODULE FUNCTIONAL DESCRIPTIONS

$$P_i(t) = \begin{cases} S \{u_j(t)\}A, & u_j(t) > 0 \\ 0 & , u_j(t) \leq 0 \end{cases} \quad (13)$$

NØLIN4 loads are computed as,

$$P_i(t) = \begin{cases} -S \{-u_i(t)\}A, & u_i(t) < 0 \\ 0 & , u_i(t) \geq 0. \end{cases} \quad (14)$$

The user specifies the set of times at which data is to be saved. If the current time is an output time, the displacement vector for time $t = t_i$ is output.

The velocity vector given by:

$$\{\dot{u}_i\} = \frac{1}{2\Delta t} [\{u_{i+1}\} - \{u_{i-1}\}], \quad (15)$$

is output.

The acceleration vector given by

$$\{\ddot{u}_i\} = \frac{1}{\Delta t^2} [\{u_{i+1}\} + \{u_{i-1}\} - 2\{u_i\}]. \quad (16)$$

is output.

If the time step is scheduled to change at t_{i+1} from Δt_1 to Δt_2 , the displacement for time $i+1$ has been calculated. $\{\dot{u}_{i-1}\}$, $\{u_i\}$, and $\{u_{i+1}\}$ are saved along with $\{P_{i+1}\}$. The matrices are formed and decomposed as in Equations 9,10, and 11 for $\Delta t = \Delta t_2$.

The following equation is used for computing $\{u_{i+2}\}$,

$$[D] \{u_{i+2}\} = \frac{1}{3} \{P_i^1 + P_{i+1} + P_{i+2}\} + \{N_{i+1}\} + [C] \{u_{i+1}\} + [E] \{\dot{u}_i^1\}. \quad (17)$$

The vectors $\{P_i^1\}$ and $\{\dot{u}_i^1\}$ in the above equation are calculated as follows. Define

$$\{\dot{u}_{i+1}\} = \frac{1}{\Delta t_1} (\{u_{i+1}\} - \{u_i\}) , \quad (18)$$

FUNCTIONAL MODULE TRD (TRANSIENT ANALYSIS - DISPLACEMENT)

$$\{\ddot{u}_{i+1}\} = \frac{1}{\Delta t^2} (\{u_{i+1}\} - 2\{u_i\} + \{u_{i-1}\}) , \quad (17)$$

$$\{\dot{u}_i\} = \{u_{i+1}\} - \{\dot{u}_{i+1}\} \Delta t_2 , \quad (20)$$

then:

$$\{u_i^1\} = \{u_{i+1}\} - \Delta t_2 \{\dot{u}_{i+1}\} + \frac{\Delta t_2^2}{2} \{\ddot{u}_{i+1}\} , \quad (21)$$

$$\{P_i^1\} = [M] \{\ddot{u}_{i+1}\} + [B] \{\dot{u}_i^1\} + [K] \{u_i^1\} . \quad (22)$$

If the CONTINUE mode is set (NCOL > 0) (see Section 11.3.2 of the Theoretical Manual for details), TRD1C/TRD1C2 will extract the displacement (u_n), velocity (\dot{u}_n) and acceleration (\ddot{u}_n) from the specified time step of the previous run. u_1 (the first displacement of the continued run) is given by

$$[D] \{u_1\} = \frac{1}{3} \{P_{-1} + P_0 + P_1\} + \{N_0\} + [C] \{u_n\} + [E] \{u_{-1}\} , \quad (23)$$

where

$$\{P_0\} = [K] \{u_n\} + [B] \{\dot{u}_n\} + [M] \{\ddot{u}_n\} , \quad (24)$$

$$\{u_{-1}\} = \{u_n\} - \Delta t \{\dot{u}_n\} + \frac{\Delta t^2}{2} \{\ddot{u}_n\} , \quad (25)$$

$$\{\dot{u}_{-1}\} = \{\dot{u}_n\} - \{\ddot{u}_n\} \Delta t , \quad (26)$$

and

$$\{P_{-1}\} = [M] \{\ddot{u}_n\} + [B] \{\dot{u}_{-1}\} + [K] \{u_{-1}\} . \quad (27)$$

2. Solution of Uncoupled Modal Equation: If the method of matrix formulation is modal and no transfer functions or direct input matrices are used, the equations may be solved in a more accurate, more direct manner. The diagonal terms of MHH, BHH, and KHH are stored in core. The following data are necessary to solve the transient behavior of a modal coordinate (1).

MODULE FUNCTIONAL DESCRIPTIONS

m_i = Modal mass of mode (MHH)

b_i = Modal damping coefficient (BHH)

K_i = Modal stiffness (KHH)

$$\omega_{oi} = (K_i/m_i)^{1/2} , \quad (28)$$

$$\beta_i = \frac{b_i}{2m_i} , \quad (29)$$

$$\omega_i^2 = |\omega_{oi}^2 - \beta^2| , \quad (30)$$

t_j = time of the j^{th} time step,

h_j = time increment after the j^{th} time,

f_{ij} = applied load on coordinate i at the j^{th} time.

The following coefficients are generated for each distinct time increment and stored in core.

There are four cases. ($\epsilon = 10^{-5}$ and the subscript i is implied).

a. If $\omega_o^2 > \beta^2 + \epsilon$ (underdamped):

$$F = e^{-\beta h} (\cos \omega h + \frac{\beta}{\omega} \sin \omega h) , \quad (31)$$

$$G = \frac{1}{\omega} e^{-\beta h} \sin \omega h , \quad (32)$$

$$A = \frac{1}{hkw} \{ e^{-\beta h} [(\frac{\omega^2 - \beta^2}{\omega_o^2} - \beta h) \sin \omega h - (\frac{2\omega\beta}{\omega_o^2} + h\omega) \cos \omega h] + \frac{2\beta\omega}{\omega_o^2} \} , \quad (33)$$

$$B = \frac{1}{hkw} \{ e^{-\beta h} [(-\frac{\omega^2 - \beta^2}{\omega_o^2}) \sin \omega h + \frac{2\omega\beta}{\omega_o^2} \cos \omega h] + \omega h - \frac{2\beta\omega}{\omega_o^2} \} , \quad (34)$$

$$F' = -\frac{\omega_o^2}{\omega} e^{-\beta h} \sin \omega h , \quad (35)$$

$$G' = e^{-\beta h} (\cos \omega h - \frac{\beta}{\omega} \sin \omega h) , \quad (36)$$

FUNCTIONAL MODULE TRD (TRANSIENT ANALYSIS - DISPLACEMENT)

$$A' = \frac{1}{hkw} [e^{-\beta h} \{(\beta + h\omega_0^2) \sin \omega h + \omega \cos \omega h\} - \omega] , \quad (37)$$

$$B' = \frac{1}{hkw} [-e^{-\beta h} (\beta \sin \omega h + \omega \cos \omega h) + \omega] , \quad (38)$$

b. If $|\omega_0^2 - \beta^2| < \epsilon$ (critically damped):

$$F = e^{-\beta h} (1 + \beta h) , \quad (39)$$

$$G = h e^{-\beta h} , \quad (40)$$

$$A = \frac{1}{hk} \left[\frac{2}{\beta} - \frac{1}{\beta} e^{-\beta h} (2 + 2h\beta + h^2 \beta^2) \right] , \quad (41)$$

$$B = \frac{1}{hk\beta} [-2 + \beta h + e^{-\beta h} (2 + \beta h)] , \quad (42)$$

$$F' = -\beta^2 h e^{-\beta h} , \quad (43)$$

$$G' = e^{-\beta h} (1 - \beta h) , \quad (44)$$

$$A' = \frac{1}{hk} [e^{-\beta h} (1 + h\beta + h^2 \beta^2) - 1] , \quad (45)$$

$$B' = \frac{1}{hk} [1 - e^{-\beta h} (\beta h + 1)] . \quad (46)$$

c. If $\omega_0^2 < \beta^2 - \epsilon$ (over damped):

$$F = e^{-\beta h} \left(\cosh \omega h + \frac{\beta}{\omega} \sinh \omega h \right) , \quad (47)$$

$$G = \frac{1}{\omega} e^{-\omega h} \sinh \omega h , \quad (48)$$

$$A = \frac{1}{hkw} \left\{ e^{-\beta h} \left[\left(\frac{\omega^2 + \beta^2}{\omega_0^2} - h\beta \right) \sinh \omega h - \left(\frac{2\omega\beta}{\omega_0^2} + h\omega \right) \cosh \omega h \right] + \frac{2\omega\beta}{\omega_0^2} \right\} , \quad (49)$$

$$B = \frac{1}{hkw} \left\{ e^{-\beta h} \left[\frac{\omega^2 + \beta^2}{\omega_0^2} \sinh \omega h + \frac{2\omega\beta}{\omega_0^2} \cosh \omega h \right] + \omega h - \frac{2\beta\omega}{\omega_0^2} \right\} , \quad (50)$$

MODULE FUNCTIONAL DESCRIPTIONS

$$F' = -\frac{\omega_0^2}{\omega} e^{-\beta h} \sinh \omega h, \quad (51)$$

$$G' = e^{-\beta h} \left(\cosh \omega h - \frac{\beta}{\omega} \sinh \omega h \right), \quad (52)$$

$$A' = \frac{1}{h k \omega} \left[e^{-\beta h} \{ (\beta + h \omega_0^2) \sinh \omega h + \omega \cosh \omega h \} - \omega \right], \quad (53)$$

$$B' = \frac{1}{h k \omega} \left[-e^{-\beta h} (\beta \sinh \omega h + \omega \cosh \omega h) + \omega \right]. \quad (54)$$

d. If $|\omega_0| = |\beta| \leq \xi$ (undamped):

$$F = 1, \quad (55)$$

$$G = h, \quad (56)$$

$$A = h^2/3m, \quad (57)$$

$$B = h^2/6m, \quad (58)$$

$$F' = 0, \quad (59)$$

$$G' = 1, \quad (60)$$

$$A' = h/2m, \quad (61)$$

$$B' = h/2m. \quad (62)$$

The equations for each displacement, velocity, and acceleration in terms of the applied loads and previous displacement and velocity are:

$$\xi_{1,j+1} = F_1 \xi_{1,j} + G_1 \dot{\xi}_{1,j} + A_1 f_{1,j} + B_1 f_{1,j+1}, \quad (63)$$

$$\dot{\xi}_{1,j+1} = F'_1 \xi_{1,j} + G'_1 \dot{\xi}_{1,j} + A'_1 f_{1,j} + B'_1 f_{1,j+1}, \quad (64)$$

$$\ddot{\xi}_{1,j+1} = \frac{P_{1,j+1}}{m_1} + \frac{b_1 \dot{\xi}_{1,j+1}}{m_1} - \frac{K_1 \xi_{1,j+1}}{m_1}. \quad (65)$$

FUNCTIONAL MODULE TRD (TRANSIENT ANALYSIS - DISPLACEMENT)

4.65.8 Subroutines

Utility routines PRETAB, TAB, SSG2A, CALCV, SSG2B, ADD, SDCOMP, and DECOMP are used. See subroutine descriptions, Section 3 for details.

4.65.8.1 Subroutine Name: TRD1A Single Precision TRD1A2 Double Precision

1. Entry Point: TRD1A, TRD1A2
2. Purpose: To assemble the loads at all time steps.
3. Calling Sequence: CALL TRD1A (CASECC, TRL, IC, NLFTP, NGRDUP, MDDAL)
CALL TRD1A2 (CASECC, TRL, IC, NLFTP, NGRDUP, MDDAL)

TRL, CASECC - GINØ file numbers of their respective data blocks - integer - input.
IC - GINØ file number of initial condition matrix - integer - input.
NLFTP - Non-linear load set id selected in CASECC - integer - input - output.
NGRDUP - Number of time step changes - integer - output.
MDDAL - If MDDAL = 1, a modal formulation is being used - integer - input.

4.65.8.2 Subroutine Name: INITL Single Precision INITL2 Double Precision

1. Entry Point: INITL, INITL2
2. Purpose: To form [C] and [E] matrices and to form and decompose the [D] matrix.
3. Calling Sequence: CALL INITL (OFFSET, DELTA)
CALL INITL2 (OFFSET, DELTA)

COMMON/TRDXX/ See Section 4.65.8.3.

OFFSET - Length of reserved area of core - integer - input.
DELTA - Current time increment - real - input.

4. Method: INITL/INITL2 will choose between symmetric and unsymmetric decomposition based on the trailers of the input matrices [K], [B], and [M]. It will also set TSYM in /TRDXX/ to inform the remaining routines.

MODULE FUNCTIONAL DESCRIPTIONS

4.65.8.3 Subroutine Name: TRD1C Single Precision
TRD1C2 Double Precision

1. Entry Point: TRD1C, TRD1C2
2. Purpose: To solve the coupled equations.
3. Calling Sequence: CALL TRD1C (IC,PAPPLD,NGRØUP,NLFTP,UDV,I,SCR1,DIT,NLFT,NØUE,MØDA1,PNL)
CALL TRD1C2 (IC,PAPPLD,NGRØUP,NLFTP,UDV,I,SCR1,DIT,NLFT,NØUE,MØDA1,PNL)

IC,NGRØUP } - Are as described in TRD1A - integer - input.
NLFTP

UDV,DIT } - Are GINØ file numbers of their respective data blocks - integer - input.
NLFT,PNL

SCR1 - GINØ file number of a scratch file.

PAPPLD - GINØ file number of applied loads - integer - input.

NØUE - Module parameter.

MØDA1 - -1 if FØRM ≠ MØDAL. 1 if FØRM = MØDAL - integer - input.

I - Current loop count. Runs from 1 to number of time step changes - integer - input.

COMMON/TRDXX/IK(7),IM(7),IB(7),C,LLL,ULL,E,SCR1,SCR2,IØPEN,ISYM,TØ,NØPD,ISP NL

IK(7) - Matrix control block for K matrix.

IM(7) - Matrix control block for M matrix.

IB(7) - Matrix control block for B matrix.

C - GINØ file number for C matrix.

LLL,ULL - GINØ file numbers for decomposition products of D matrix.

E - GINØ file number for E matrix.

SCR1,SCR2 - GINØ file numbers for 2 scratch files.

IØPEN - 1 implies C,ULL,LLL, and E are open.
0 implies C,ULL,LLL, and E are closed.

ISYM - 1 implies unsymmetric decomposition used.
0 implies symmetric decomposition used.

TØ - Initial time (usually 0.0).

NØPD - True if PD does not exist.
False if PD exists.

ISP NL - 0 if PNLD is not to be formed.
1 if PNLD is to be formed.

4.65.8.4 Subroutine Name: FØRM1 Single Precision
FØRM12 Double Precision

1. Entry Point: FØRM1, FØRM12
2. Purpose: To compute $\{u_{-1}\}$, $\{P_0^1\}$, and $\{P_{-1}\}$ for starting the integration procedure.

FUNCTIONAL MODULE TRD (TRANSIENT ANALYSIS - DISPLACEMENT)

3. Calling Sequence: CALL FØRM1 (UØ,UDØTØ,UI,PØ,PI,DELTAT,IBUF)
CALL FØRM12 (UØ,UDØTØ,UI,PØ,PI,DELTAT,IBUF)

UØ - Array of core containing $\{u_0\}$ - real - input.
UDØTØ - Array of core containing $\{u_0\}$ - real - input.
UI - Array of core for storage of $\{u_{-1}\}$ - real - output.
PØ - Array of core for storage of $\{P_0^1\}$ - real - output.
PI - Array of core for storage of $\{P_{-1}\}$ - real - output.
DELTAT - Current time step size - real - input.
IBUF - GINØ buffer.

4.65.8.5 Subroutine Name: MATVEC Single Precision
MATVC2 Double Precision

1. Entry Point: MATVEC, MATVC2

2. Purpose: To form the product $\{X\} = \{X\} + [A] \{Y\}$ where $[A]$ is a matrix and $\{Y\}$ is a vector.

3. Calling Sequence: CALL MATVEC (Y,X,FILEA,IBUF)
CALL MATVC2 (Y,X,FILEA,IBUF)

Y - Array of core containing Y array real - input.
X - Array of core containing X array real - input/output.
FILEA - Matrix control block for A. If $FILEA(1) \leq 0$, MATVEC will return.
IBUF - GINØ buffer. If $IBUF \leq 0$, MATVEC/MATVC2 will assume the file is already in core.
COMMON/TRDXX/ (see Section 4.65.8.3).

4.65.8.6 Subroutine Name: STEP Single Precision
STEP2 Double Precision

1. Entry Point: STEP, STEP2

2. Purpose: To integrate forward 1 time step.

3. Calling Sequence: CALL STEP (U2,U1,UØ,P,IBUF)
CALL STEP2 (U2,U1,UØ,P,IBUF)

U2 - Array which will contain $\{u_{i+2}\}$ - real - output.
U1 - Array containing $\{u_{i+1}\}$ - real - input.
UØ - Array containing $\{u_i\}$ - real - input.
P - Array containing combined load - real - input.
IBUF - GINØ buffer - input.
COMMON/TRDXX/ (see Section 4.65.8.3).

4.65.8.7 Subroutine Name: INTFBS

1. Entry Point: INTFBS
 2. Purpose: To perform the forward-backward substitution necessary to solve the system of equations: $[A] \{Y\} = \{X\}$ for $\{Y\}$ if $[A]$ was unsymmetric.
 3. Calling Sequence: CALL INTFBS (X,Y,IBUF)
- X - Load vector (i.e., right hand side) - real - input.
- Y - Solution vector - real - output.
- IBUF - GINØ buffer.
- COMMON/TRDXX/ (see Section 4.64.8.3).
- COMMON/INTFBS/FILEL(7),FILEU(7)
- FILEL - Matrix control block of the lower triangular factor from the decomposition of A.
- FILEU - Matrix control block of the upper triangular factor from the decomposition of B.

4.65.8.8 Subroutine Name: TRD1D Single Precision
TRD1D2 Double Precision

1. Entry Point: TRD1D, TRD1D2
 2. Purpose: To compute the non-linear loads at each time step.
 3. Calling Sequence: CALL TRD1D
CALL TRD1D2
- COMMON/TRDD1/NLFT,DIT,NLFTP,NØUT,ICØUNT,ILØØP,MØDA1,NZ,ICØRE,IU2,IP4,IPNL(7),NMØDES,
NSTEP,PNL
- The variables DIT,NLFT,NLFTP,MØDA1 and PNL are defined as in TRD1C (see section 4.65.8.4).
- NØUT - Output interval - integer - input.
- ICØUNT - Current time step counter - integer - input.
- ILØØP - Current time change counter - integer - input.
- NZ - Length of open core - integer - input.
- ICØRE - Pointer to first unused cell of open core - integer - input.
- IU2 - Pointer to displacement vector - 1 - integer - input.
- IP4 - Pointer to load area -1 - integer - input.
- IPNL - Matrix control block for PNL - integer - input/output.
- NMØDES - Number of modes if modal formulation is being used - integer - input.
- NSTEP - Number of times steps for this time increment - integer - input.

FUNCTIONAL MODULE TRD (TRANSIENT ANALYSIS - DISPLACEMENT)

4.65.8.9 Subroutine Name: TRD1E

1. Entry Point: TRD1E
2. Purpose: To solve the uncoupled modal equations.
3. Calling Sequence: CALL TRD1E (MHH,BHH,KHH,PH,UHV,NGRØUP)
MHH,BHH,KHH, - GINØ file numbers of their respective data blocks - integer- input.
PH,UHV
NGRØUP - Number of time step changes - integer - input.

4.65.8.10 Subroutine Name: FØRM2 Single Precision
FØRM22 Double Precision

1. Entry: FØRM2, FØRM22
2. Purpose: To compute $\{u_i^{\cdot}\}$ and $\{P_i^{\cdot}\}$ when changing time steps. (See Equations 15 through 25).
3. Calling Sequence: CALL FØRM2 (UDDIP1,UDIP1,UIP,PIP,IBUF)
CALL FØRM22 (UDDIP1,UDIP1,UIP,PIP,IBUF)
UDDIP1 - Array of core containing $\{\dot{u}_{i+1}\}$ - real - input.
UDIP1 - Array of core containing $\{\dot{u}_{i+1}\}$ - real - input.
UIP - Array of core containing $\{u_i^{\cdot}\}$ - real - output.
PIP - Array of core containing $\{P_i^{\cdot}\}$ - real - output.
IBUF - GINØ buffer
COMMON/TRDXX/ (see Section 4.65.8.3)

4.65.8.11 Subroutine Name: FBSINT

1. Entry Point: FBSINT
2. Purpose: To perform the same functions as INTFBS (Section 4.65.8.7) if [A] is symmetric. Subroutine FBS21 is called to process data in mixed precisions.
3. Calling Sequence: Identical to INTFBS.

MODULE FUNCTIONAL DESCRIPTIONS

4.65.9 Design Requirements

1. Open core at /TRDIX/ is illustrated as follows:

COMMON/TRDIX/

Open core for DECOMP or SDCOMP and ADD
Number of Steps
Δt
Output Interval

} Repeated for each time step change

FUNCTIONAL MODULE TRD (TRANSIENT ANALYSIS DISPLACEMENT)

This table is at the bottom of open core through the module.

2. Open Core at /TRDC1/ is illustrated as follows:

COMMON/TRDC1/	
u ₁	} NRØW
u ₂	} NRØW
u ₃	} NRØW
P1	} NRØW
P2	} NRØW
P3	} NRØW
P4	} NRØW
Type	} 5 words for each non-linear load card selected.
Table ID's	} Table ID's selected on NØLIN cards.
Table's for TAB	
Tab Buffer	} Used only if non-linear loads are selected.
C Buffer	} GINØ Buffers
D Buffer	
ULL Buffer	
LLL Buffer	
Solution Buffer	
Load Buffer	
Utility Buffer	

MODULE FUNCTIONAL DESCRIPTIONS

3. Open Core at /TRDE1/ is illustrated as follows:

COMMON/TRDE1/	
MHH	
BHH	
KHH	
F	
G	
A	
B	
F'	
G'	
A'	
B'	
ϵ_j	
ϵ_{j+1}	
ϵ'_j	
ϵ'_{j+1}	
r_j	
r_{j+1}	
PH Buffer	}
UHV Buffer	

Each section is of length H

2 GINØ buffers

FUNCTIONAL MODULE TRD (TRANSIENT ANALYSIS - DISPLACEMENT)

4.65.10 Diagnostic Messages

TRD may issue the following messages

3001, 3002, 3003, 3005, 3007, 3008, 3031, 3044, 3045, 3046.

FUNCTIONAL MODULE GKAM (GENERAL K ASSEMBLER MODAL)

4.66 FUNCTIONAL MODULE GKAM (GENERAL K ASSEMBLER MODAL)

4.66.1 Entry Point: GKAM

4.66.2 Purpose

To assemble the modal mass, damping and stiffness matrices.

4.66.3 DMAP Calling Sequence

GKAM USETD,PHIA,MI,LAMA,DIT,M2DD,B2DD,K2DD,CASECC / MHH,BHH,KHH,PHIDH / V,N,NØUE /
 C,Y,LMØDES / C,Y,LFREQ / C,Y,HFREQ / V,N,NØM2PP / V,N,NØB2PP / V,N,NØK2PP /
 V,N,NØNCUP / V,N,FMØDE / C,Y,KDAMP \$

4.66.4 Input Data Blocks

USETD - Displacement set definitions table dynamics.
PHIA - Eigenvectors matrix giving the eigenvectors (displacements) in the a set.
MI - Modal mass matrix.
LAMA - Real Eigenvalue Table.
DIT - Direct Input Table.
M2DD - Direct input mass matrix - d set.
B2DD - Direct input damping matrix - d set.
K2DD - Direct input stiffness matrix - d set.
CASECC - Case Control Data Table.

Notes:

1. USETD may be purged if $NØUE < 0$.
2. PHIA cannot be purged.
3. MI may be purged.
4. LAMA cannot be purged
5. DIT cannot be purged if $SDAMP \neq 0$ in CASECC.
6. CASECC cannot be purged
7. M2DD cannot be purged if $NØM2PP \geq 0$.
8. B2DD cannot be purged if $NØB2PP \geq 0$.
9. K2DD cannot be purged if $NØK2PP \geq 0$.

MODULE FUNCTIONAL DESCRIPTIONS

4.66.5 Output Data Blocks

- MHH - Modal mass matrix - h set.
- BHH - Modal damping matrix - h set.
- KHH - Modal stiffness matrix - h set.
- PHIDH - Transformation matrix from d set to modal coordinates.

Note: No output matrix can be purged.

4.66.6 Parameters

- NØUE - Input-integer-no default. NØUE indicates presence and number of extra points.
- LMØDES - Input-integer-no default. LMØDES selects the first LMØDES eigenvectors (or all if there are less than LMØDES) to use for the modal coordinates.
- LFREQ - Input-real-no default. If LMØDES = 0, eigenvectors with eigenvalues between LFREQ and HFREQ are used in the modal formulation.
- HFREQ - Input-real-no default. See LFREQ.
- NØM2PP - Input-integer-no default. If NØM2PP < 0, M2DD will not be used.
- NØB2PP - Input-integer-no default. If NØB2PP < 0, B2DD will not be used.
- NØK2PP - Input-integer-no default. If NØK2PP < 0, K2DD will not be used.
- NØNCUP - Output-integer-no default. If no direct input matrices exist the problem is considered uncoupled and NØNCUP is set to -1.
- FMØDE - Output-integer-default = 1. The mode number of the first selected eigenvector is stored in FMØDE.
- KDAMP - Input-integer-default = -1. KDAMP chooses the method of computing damping.

4.66.7 Method

The general system assembly module for the modal method is used when the real eigenvalues for the structure have been determined. With this method, it is possible to decrease the order of the problem without sacrificing accuracy. The module forms the conversion matrix between modal displacements and all free physical displacements of the system. It then forms the general matrices in terms of displacements of the modes and the extra points.

CASECC is read, and the selected structural damping table "id" is stored.

LAMA is read and the selected eigenvalues are stored in core. If an eigenvalue is selected, the corresponding column of PHIA is copied onto PHIDH1, a scratch file.

FUNCTIONAL MODULE GKAM (GENERAL K ASSEMBLER MODAL)

If extra points are not present ($N\emptysetUE < 0$), $PHIDH = PHIDH1$. If extra points are present:

$$[\Phi_{dh}] = \begin{bmatrix} \Phi_a & 0 \\ 0 & I \end{bmatrix} \quad (1)$$

This is accomplished in subroutine GKAM1B.

The "H" matrices are formed:

$$[M_{hh}] = \begin{bmatrix} m_i & 0 \\ 0 & 0 \end{bmatrix} + [\Phi_{dh}]^T [M_{dd}^2] [\Phi_{dh}] , \quad (2)$$

$$[B_{hh}] = \begin{bmatrix} b_i & 0 \\ 0 & 0 \end{bmatrix} + [\Phi_{dh}]^T [B_{dd}^2] [\Phi_{dh}] , \quad (3)$$

$$[K_{hh}] = \begin{bmatrix} k_i & 0 \\ 0 & 0 \end{bmatrix} + [\Phi_{dh}]^T [K_{dd}^2] [\Phi_{dh}] , \quad (4)$$

where m_i = diagonal terms of MI , and

$$b_i = m_i \omega_i g(\omega_i) , \quad (5)$$

$$k_i = m_i \omega_i^2 , \quad (6)$$

if $KDAMP = -1$ (the default).

If $KDAMP = 1$ (used for Aeroelastic),

$$b_i = 0 , \quad (7)$$

$$k_i = (1 + ig(\omega_i)) \omega_i^2 m_i . \quad (8)$$

ω_i is the frequency for the mode from LAMA and $g(\omega_i)$ is the tabular structural damping table selected in CASECC. If no selection is made, $g(\omega_i) = 0.0$. The "H" matrices are formed using subroutines GKAM1A, SSG2B, TAB, CALCV, MERGE.

4.66.8 Subroutines

4.66.8.1 Subroutine Name: GKAM1B.

1. Entry Point: GKAM1B.
2. Purpose: To construct $[\Phi_{dh}]$ if extra points are present.

MODULE FUNCTIONAL DESCRIPTIONS

3. Calling Sequence: CALL GKAM1B (USETD,SCR1,SCR2,PHIDH,PHIDH1,MØDES,CØRE,LHSET,NØUE)

USETD - GINØ file number of USETD - integer - input.
 SCR1 - GINØ file number of 1st scratch file - integer - input.
 SCR2 - GINØ file number of 2nd scratch file - integer - input.
 PHIDH - GINØ file number of PHIDH - integer - input.
 PHIDH1 - GINØ file number of PHIDH1 - integer - input.
 MØDES - Number of modes selected - integer - input.
 CØRE - Array of open core.
 LHSET - Length of h set - integer - output.
 NØUE - Extra point flag NØUE ≥ 0 indicates presence of extra points - integer - input.

4.66.8.2 Subroutine Name: GKAM1A.

1. Entry Point: GKAM1A.

2. Purpose: To form $[M_{hh}]$, $[B_{hh}]$, or $[K_{hh}]$.

3. Calling Sequence: CALL GKAM1A (MI,PHIDH,DIT,SCR1,SCR2,IØPT,IHH,NØI2DD,CØRE,MØDES,
 SDITD,LHSET,I2DD,IMSKIP,SCR3)

MI - GINØ file number of MI - integer - input.
 PHIDH - GINØ file number of PHIDH - integer - input.
 DIT - GINØ file number of DIT - integer - input.
 SCR1 - GINØ file number of scratch 1 - integer - input.
 SCR2 - GINØ file number of scratch 2 - integer - input.
 SCR3 - GINØ file number of scratch 3 - integer - input.
 IHH - GINØ file number of HH file (M, B, or K) being constructed - integer - input.
 I2DD - GINØ file number of 2DD file being used with IHH (K2DD, M2DD or B2DD) -
 integer - input.
 IØPT - Flag for equation to use

1 \Rightarrow MHH
 2 \Rightarrow BHH
 3 \Rightarrow KHH

FUNCTIONAL MODULE GKAM (GENERAL K ASSEMBLER MODAL)

- integer - input.

NØI2DD - NØI2DD < 0 implies I2DD purged - integer - input.

MØDES - Number of modes selected - integer - input.

SDTID - Id of structural damping table to be used for BHH - integer - input.

LHSET - Length of H set - integer - input.

IMSKIP - Number of records to skip in MI before extracting diagonal terms - integer - input.

CØRE - Array of modes selected.

4.66.9 Design Requirements

Three scratch files are necessary. Open core at /GKAMIX/ is used for mode storage. One packed eigenvector must be held in core.

4.66.10 Diagnostic Messages

Fatal error messages 3007 and 3008 may be issued by GKAM.

FUNCTIONAL MODULE DDR1 (DYNAMIC DATA RECOVERY - PART 1)

4.67 FUNCTIONAL MODULE DDR1 (DYNAMIC DATA RECOVERY - PART 1)

4.67.1 Entry Point: DDR1

4.67.2 Purpose: To transform modal solutions to physical solutions:

$$\{u_d\} = [\phi_{dh}] \{u_h\} .$$

4.67.3 DMAP Calling Sequence

DDR1 $\begin{Bmatrix} \text{PHIH} \\ \text{UHVF} \\ \text{UHVT} \end{Bmatrix}$, PHIDH/ $\begin{Bmatrix} \text{CPHID} \\ \text{UDVIF} \\ \text{UDVIT} \end{Bmatrix}$

4.67.4 Input Data Blocks

- PHIH - Complex eigenvectors (h set)
- UHVF - Modal frequency response solution vector (h set)
- UHVT - Modal transient solution vector (h set)
- PHIDH - Transformation matrix from d set to modal coordinates.

4.67.5 Output Data Blocks

- CPHID - Complex eigenvector matrix transformed from modal to physical coordinates
- UDVIF - Displacement vector matrix in frequency response problems (d set).
- UDVIT - Displacement vector matrix in transient response problems (d set).

4.67.6 Parameters

None

4.67.7 Method

Subroutine SSG2B is called to compute $\{u_d\}$ as in Equation 1.

4.67.8 Subroutines

DDR1 has no auxiliary subroutines. See section 3.5.13 for a descriptions of SSG2B.

4.67.9 Design Requirements

One scratch file is needed.

FUNCTIONAL MODULE DDR2 (DYNAMIC DATA RECOVERY - PART 2)

4.68 FUNCTIONAL MODULE DDR2 (DYNAMIC DATA RECOVERY - PART 2)

4.68.1 Entry Point: DDR2

4.68.2 Purpose

To compute mode acceleration displacements.

4.68.3 DMAP Calling Sequence

DDR2 USETD,UDVIT,{ $\begin{matrix} \text{PDP} \\ \text{PDT} \end{matrix}$ }, K2DD,B2DD,MDD,FRL,ULL,DM/UDV1,UEVF,PAF/V,N,TYPE/V,N,NØUE/
V,N,REACT/V,N,FRQSET \$

4.68.4 Input Data Blocks

USETD - Displacement set definitions table dynamics.
UDVIT - Displacement vectors - d set.
PDF - Dynamic load matrix for frequency analysis - d set.
PDT - Dynamic load matrix for transient analysis - d set.
K2DD - Direct input stiffness matrix - d set.
B2DD - Direct input damping matrix - d set.
MDD - Dynamic mass matrix - d set.
FRL - Frequency Response List.
ULL - Upper triangular factor of KLL - & set.
DM - Rigid body transformation matrix.

Notes:

1. USETD must not be purged.
2. UDVIT must not be purged.
3. PDF must not be purged.
4. FRL must not be purged if TYPE = FREQ.
5. MDD must not be purged.
6. ULL must not be purged.
7. DM must not be purged if REACT \geq 0.

4.68.5 Output Data Blocks

- UDV1 - Displacements after mode acceleration - d set.
- UEVF - Displacements at the extra points.
- PAF - Equivalent load vector for mode acceleration computations - a set.

4.68.6 Parameters

- TYPE - Input-BCD-no default. TYPE determines the type of mode acceleration which will be used, TRAN for transient or FREQ for frequency response.
- NØUE - Input-integer-no default. NØUE ≥ 0 indicates presence of extra points.
- REACT - Input-integer-no default. REACT ≥ 0 indicates presence of supports.
- FRQSET - Input-integer-no default. FRQSET chooses the frequency list if TYPE = FREQ.

4.68.7 Method

The equivalent load vector is computed:

$$\{P_d^a\} = \{P_d\} - [K_{dd}^2] \{u_d\} - [B_{dd}^2] \{\dot{u}_d\} - [M_{dd}] \{\ddot{u}_d\}. \quad (1)$$

For a transient analysis problem $\{u_d\}$, $\{\dot{u}_d\}$, and $\{\ddot{u}_d\}$ are given explicitly. For Frequency Response Analysis:

$$\{\dot{u}_d\} = i\omega \{u_d\}, \quad (2)$$

$$\{\ddot{u}_d\} = -\omega^2 \{u_d\}, \quad (3)$$

where ω is the forcing frequency and $\{u_d\}$ is the complex response vector. ω comes from FRQSET in FRL. The vector $\{P_d^a\}$ is the sum of applied loads and inertia loads due to the motion of the system approximated by its lower modes. The static solution using these loads will provide a better answer for displacements.

FUNCTIONAL MODULE DDR2 (DYNAMIC DATA RECOVERY - PART 2)

If extra points are present ($N\text{ØUE} \geq 0$), then

$$\{P_d^a\} \rightarrow \left\{ \frac{P_d^a}{P_e} \right\}, \quad (4)$$

$$\{u_d\} \rightarrow \left\{ \frac{u_d^a}{u_e} \right\}. \quad (5)$$

$\{u_e\}$ is placed in data block UEVF. Subroutines CALCV and SSG2A perform this calculation.

If supports are present ($\text{REACT} \geq 0$), then

$$\{P_\ell^a\} \rightarrow \left\{ \frac{P_\ell^a}{P_r} \right\}, \quad (6)$$

$$\{u_a\} \rightarrow \left\{ \frac{u_\ell^a}{u_r} \right\}. \quad (7)$$

Solve for $\{u_a^a\}$:

$$[L_{\ell\ell}] [U_{\ell\ell}] \{u_\ell^a\} = \{P_\ell^a\}. \quad (8)$$

This is accomplished in subroutine SSG3A.

If supports are present, then

$$\{u_a^a\} = \left\{ \frac{\{u_\ell^a\} + [D] \{u_r\}}{u_r} \right\}, \quad (9)$$

otherwise, $\{u_a^a\} = \{u_\ell^a\}$. Subroutine SDR1B performs this calculation.

MODULE FUNCTIONAL DESCRIPTIONS

If extra points are present, then

$$\{u_d^a\} \leftarrow \begin{Bmatrix} u_a^a \\ \vdots \\ u_e^a \end{Bmatrix}. \quad (10)$$

Note: If the problem type is transient, $\{u_d^a\}$ must be merged with $\{\dot{u}_d\}$ and $\{\ddot{u}_d\}$.

4.68.8 Subroutines Called

CALCV - See section 3.5.5.

SSG2A - See section 3.5.7.

SSG2B - See section 3.5.13.

SSG3A - See section 3.5.18.

SDR1B - See section 3.5.8.

4.68.8.1 Subroutine Name: DDR1A.

1. Entry Point: DDR1A.
2. Purpose: To construct the equivalent load vector $\{P_d^a\}$.
3. Calling Sequence: CALL DDR1A(PDF,K2DD,B2DD,MDD,UDV,PAF,FRL,FRQSET,SCR1,SCR2,SCR3,SCR4,TYPE,SCR5).

PDF K2DD B2DD MDD UDV PAF FRL SCR1-5	}	GINØ file number of appropriate data block - integer - input.
---	---	---

FRQSET - Frequency set list id - integer - input. FRQSET will be used only if TYPE = FREQ.
TYPE - Problem type - BCD - input.

FUNCTIONAL MODULE DDR2 (DYNAMIC DATA RECOVERY - PART 2)

4.68.8.2 Subroutine Name: DDR1B

1. Entry Point: DDR1B.
2. Purpose: To merge displacements with previously computed velocity and acceleration in a transient problem.
3. Calling Sequence: CALL DDR1B (UDV,UAD,UADV).
 - UDV - GINØ file number of displacement, velocity and acceleration file - integer - input.
 - UAD - GINØ file number of equivalent displacements - integer - input.
 - UADV - GINØ file number of new displacements, velocity and acceleration - integer - input.

4.68.9 Design Requirements

Open core for DDR2 begins at /DDR1X/. Open core for DDR1A begins /DDR1A/. Open core for DDR1B begins /DDR1B/. Six scratch files are needed.

4.68.10 Diagnostic Messages

None.

OUTPUT MODULE XYPLØT (X-Y DATA PLOTTER)

4.69 OUTPUT MODULE XYPLØT (X-Y DATA PLOTTER)

4.69.1 Entry Point: XYPLØT

4.69.2 Purpose

To process information supplied by module XYTRAN through a single data block and output to either PLT1 (BCD plot tape) or PLT2 (binary plot tape) for labeling and plotting X-Y data on an off-line plotter.

4.69.3 DMAP Calling Sequence

XYPLØT XYPLTT// \$

4.69.4 Input Data Blocks

XYPLTT - Plotting Control Values Table. Note if XYPLTT is purged, XYPLØT returns control without action.

4.69.5 Output Data Blocks

None. (All output consists of physical tapes produced for off-line plotters and possibly user warning messages to the installation output unit for printing).

4.69.6 Parameters

None.

4.69.7 Method

XYPLØT initially determines open core size and assigns buffers for its input file and output file. The remaining core is used to store data points read in for each plot. The input file is then opened and spaced forward over the header record containing the data block name. Should the system not be able to locate this file, a warning message is output and XYPLØT returns control to the calling program without further action. Otherwise XYPLØT reads in the first I.D. record from the input file. A check is made to determine if the word count of this record is correct. If not, the following records are checked until either the correct

MODULE FUNCTIONAL DESCRIPTIONS

word count is found or the error count reaches a specified limit. If the specified limit is reached, XYPLØT assumes the input file is invalid and returns control to the calling program after printing a warning message.

If the I.D. record had the proper word count, XYPLØT checks if new axes are necessary. If not, the next data record is read, and the data pairs are plotted on the previous axes. When new axes are necessary, a check is made to determine if they go on the lower half of a plot. If not, XYPLØT makes a number of I.D. data validity checks. Whenever possible, where I.D. data are questionable, default values are assigned and processing continues following a warning message that this particular plot may be invalid.

After the validity checks, XYPLØT terminates the previous plot and initializes the plotting parameters for the NASTRAN plotting software. This is done for each new plot so that it is possible to produce alternate plots on two different plotters. Normally, however, plots will be done for only one plotter on any single entry to XYPLØT. If required, a new plot is initiated, and curve and axes titles are prepared from the I.D. data and generated. If not a new plot, only the axes titles are done.

At this time XYPLØT computes the constants which will be used to transform the curve data into actual plotter counts. These constants are saved and used until new axes are drawn.

Following this, XYPLØT determines if any tick marks are to be placed along the X axis and at the X maximum and X minimum lines. If there are to be tick marks, the number and spacing (linear or logarithmic) is computed for them and plotted. As the X direction tick marks are prepared, a check is made to determine if Y grid lines are requested. If so, a grid line is prepared at each tick mark and plotted. Tick mark labels are prepared and plotted at the same time as the tick marks and grid lines, if any.

After the tick marks are completed, the X and Y axes are plotted if requested.

Once the curve titles, tick marks, and labeling have been accomplished, XYPLØT reads in the next record from the input data file. Normally all the data pairs for any I.D. record can be brought into core memory with a single read. However, provision is made for additional reads if the open core space is not sufficient to contain all the data on the initial read. A check is made to determine if there are an even number of data values (i.e., an X and Y value for each data

OUTPUT MODULE XYPLØT (X-Y DATA PLOTTER)

point). If not, a warning message is printed and the last value ignored. The data are then checked against the previously defined X and Y frame minimums and maximums (integer one for the X value means skip the point). Any data outside these limits are ignored and not plotted. The remaining data points are then converted to plotter counts and plotted in one of three modes. The three modes are: point plot with choice of symbol; line plot; combination of the first two.

After finishing the data, XYPLØT reads in the next I.D. record and continues as before until and end-of-file is reached. At this point it closes the input file, terminates the current plot and returns control to the calling program.

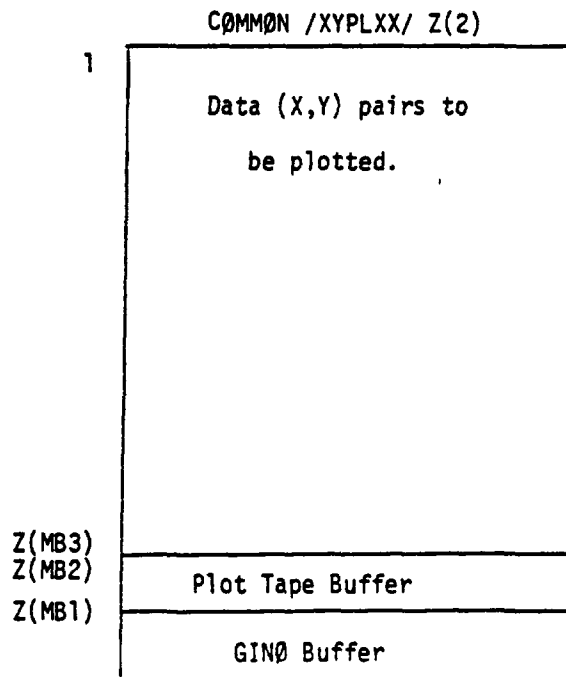
4.69.8 Subroutines

XYPLØT calls the following plotter utility subroutines: AXIS, LINE, PRINT, SØPEN, PLTSET, STPLØT, SYMBØL, TIPE, TYPFLT, and TYPINT. The descriptions of these subroutines may be found in Section 3.4,

4.69.9 Design Requirements

4.69.9.1 Allocation of Core Storage

XYPLØT uses open core for two GINØ buffers and the remainder as one large buffer for data points. It appears as follows:



MODULE FUNCTIONAL DESCRIPTIONS

Normally the data pairs buffer will be sufficiently large to hold all the data pairs for a single curve at one time. However, this is not necessary and XYPLØT could operate if the data pairs buffer were only two words long, although not efficiently. As an output module, XYPLØT has been programmed to avoid any system fatal errors. The worst condition that should occur is that no plots are produced. In all cases XYPLØT returns to the calling program so that other system functions may be continued.

4.69.9.2 Environment

The beginning of open core for XYPLØT is defined by /XYPLXX/. XYPLØT uses no scratch files. Common storage requirements consist of /XXPARM/ and /PLTDAT/ which are defined in the block data deck PLØTBD which must be loaded with XYPLØT. /CHAR94/ and SYMBLS/ are also defined in PLØTBD and are necessary for the subroutines called by XYPLØT. See Section 2.5 for a description of these common blocks.

When XYPLØT is called, there must be at least one physical tape set up to receive the plotted output, otherwise XYPLØT returns to the calling program without further action.

4.69.10 Diagnostic Messages

Diagnostic messages 991 through 997 may be output on the installation printer device as a result of XYPLØT operation. Generally they are self-explanatory and usually point out particular plots which are questionable rather than giving the user a precise method of solving the problem. This is not possible since XYPLØT receives all its information through a series of other modules rather than from the user directly. See Section 6 of the User's Manual for details.

OUTPUT MODULE ØFP (OUTPUT FILE PROCESSOR)

4.70 OUTPUT MODULE ØFP (OUTPUT FILE PROCESSOR)

4.70.1 Entry Point: ØFP

4.70.2 Purpose

ØFP outputs to the system output file, in user-oriented, self-explanatory formats, data blocks prepared for output by other functional modules.

4.70.3 DMAP Calling Sequence

ØFP DB1,DB2,DB3,DB4,DB5,DB6//V,N,CARDNØ \$

4.70.4 Input Data Blocks

One to six input data blocks in the output order desired. Any or all input data blocks may be purged.

4.70.5 Output Data Blocks

None

4.70.6 Parameters

CARDNØ - Input and output - integer - default = 0. CARDNØ is incremented by one and punched in columns 73-80 for each card punched by ØFP.

4.70.7 Method

4.70.7.1 Overall Logic Flow

The ØFP logic consists of defining one GINØ buffer and then entering one overall loop of six passes (one pass for each data block). All input data blocks are then handled identically one at a time.

Within each data block, each odd numbered (Identification) record and its respective immediately following even numbered (Data) record are considered as a pair, and is a completely separate entity. There is, and need be, no correspondence between these two records and the previous two records, or between these two records and the following two records.

MODULE FUNCTIONAL DESCRIPTIONS

Thus, within the loop for a given data block, after the file on which the data block resides is opened and its header record is skipped, ØFP reads an Identification record, defines various pointers and descriptors, and then, if any data are present in the Data record, processes this data line by line until the end-of-record is reached. This process continues for all Identification-Data record pairs.

4.70.7.2 Defining Descriptors and Pointers

Because ØFP was confronted with outputting a vast array of data classes having many data format and heading format configurations, it was decided that in order to keep ØFP from becoming a mammoth module of format statements, a system of pointers would be used in conjunction with all the different micro-format elements required.

Information in the Identification record is sufficient to select an initial class pointer. This class pointer, with the addition of a subclass pointer, points to an array of six pointers, five of which define five micro-line formats (from the master set of micro-line formats), and one of which points to a string of micro-data format pointers. These micro-data format pointers then each point to a micro-data format capable of outputting a single variable.

This design is such as to make possible the definition of macro-formats and to allow for easy modification and addition of more output data classes.

4.70.8 Subroutines

4.70.8.1 Subroutine Name: ØFPPUN

1. Entry Point: ØFPPUN
2. Purpose: To write output on the system punch unit.
3. Calling Sequence: CALL ØFPPUN (BUF,NWDS,IØPT,IDD,PNCHED)
BUF - Array to be output.
NWDS - Number of words in BUF to output.
IØPT - { 1 = Vector output.
 2 = General output.
IDD - { 0 = SØRT1 (1st word Integer).
 1 = SØRT2 (1st word Real).

OUTPUT MODULE ØFP (OUTPUT FILE PROCESSOR)

PNCHED - $\begin{cases} \text{.FALSE.} = \text{Punch heading cards.} \\ \text{.TRUE.} = \text{Do not punch heading cards.} \end{cases}$

4.70.8.2 Subroutine Name: ØFP1

1. Entry Point: ØFP1
2. Purpose: To call PAGE and write five micro-line formats.
3. Calling Sequence: CALL ØFP1

4.70.8.3 Subroutine Name: ØFP1A

1. Entry Point: ØFP1A
2. Purpose: An auxiliary routine to ØFP1. Called by ØFP1 only.
3. Calling Sequence: CALL ØFP1A(LINE)
LINE - Integer - Branch to format pointer.

4.70.8.4 Block Data Subprogram Name: ØFP1BD

ØFP1BD defines common block /ØFPBD1/.

4.70.8.5 Block Data Subprogram Name: ØFP5BD

ØFP5BD defines common block /ØFPBD5/.

4.70.8.6 Block Data Subprogram Name: ØF1PBD

ØF1PBD defines common block /ØFPB1/.

4.70.8.7 Block Data Subprogram Name: ØF2PBD

ØF2PBD defines common block /ØFPB2/.

4.70.8.8 Block Data Subprogram Name: ØF3PBD

ØF3PBD defines common block /ØFPB3/.

MODULE FUNCTIONAL DESCRIPTIONS

4.70.8.9 Block Data Subprogram Name: ØF4PBD

ØF4PBD defines common block /ØFPB4/.

4.70.8.10 Block Data Subprogram Name: ØF5PBD

ØF5PBD defines common block /ØFPB5/.

4.70.8.11 Block Data Subprogram: ØF6PBD

ØF6PBD defines common block /ØFPB6/.

4.70.8.12 Block Data Subprogram: ØF7PBD

ØF7PBD defines common block /ØFPB7/.

4.70.8.13 Block Data Subprogram Name: ØF8PBD

ØF8PBD defines common block /ØFPB8/.

4.70.8.14 Block Data Subprogram: ØF9PBD

ØF9PBD defines common block /ØFPB9/.

4.70.9 Design Requirements

The common blocks listed above interface between the main subroutines ØFP and ØFP1A. In addition COMMON/ØFPXXX/ is used to define open-core which contains the following.

L1	}	Five words which indicate the five format numbers defining the heading for the current data being output.
L2		
L3		
L4		
L5		

ID - A fifty word buffer for storage of the first fifty words of an identification record from the data block to be output.

BUFF - A GINØ buffer.

OUTPUT MODULE ØFP (OUTPUT FILE PROCESSOR)

The pointer system required by the design operates as described below. The arrays B,C,D,E, and ESINGL, referenced in the discussion below appear in subroutine ØFP.

1. The variable I is set equal to the Data type specified in the data block. The variable J is set equal to the class of data: 1 = Real - SØRT1, 2 = Complex - SØRT1, etc. Then the base pointer, CPØINT = B(I,J), is found.

B array

	SØRT 1		SØRT 2	
	Real	Complex	Real	Complex
Data type 1	0	130	120	132
Data type 2	2	134	122	136
.	4	138	124	140
.
.
.
.
Data type N
For Future Expansion				

Example:
For I = 2, J = 4.

CPØINT = B(I,J)
= 136

2. CPØINT is a index into the C array. Define: DPØINT = C(CPØINT). DPØINT is an index into the D array. Also

L1 = C(CPØINT + 1),

L2 = C(CPØINT + 2),

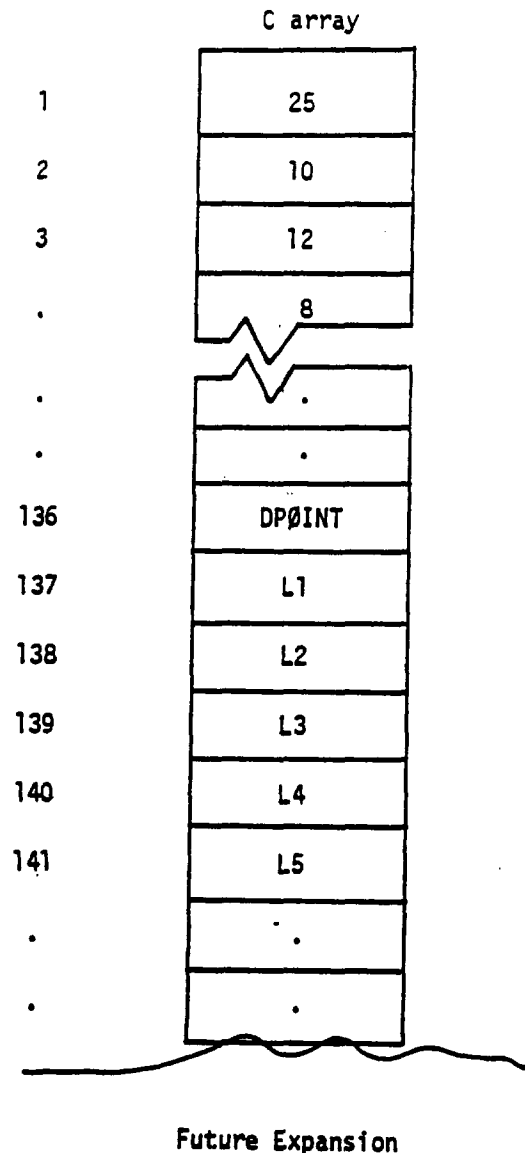
L3 = C(CPØINT + 3),

L4 = C(CPØINT + 4),

and L5 = C(CPØINT + 5).

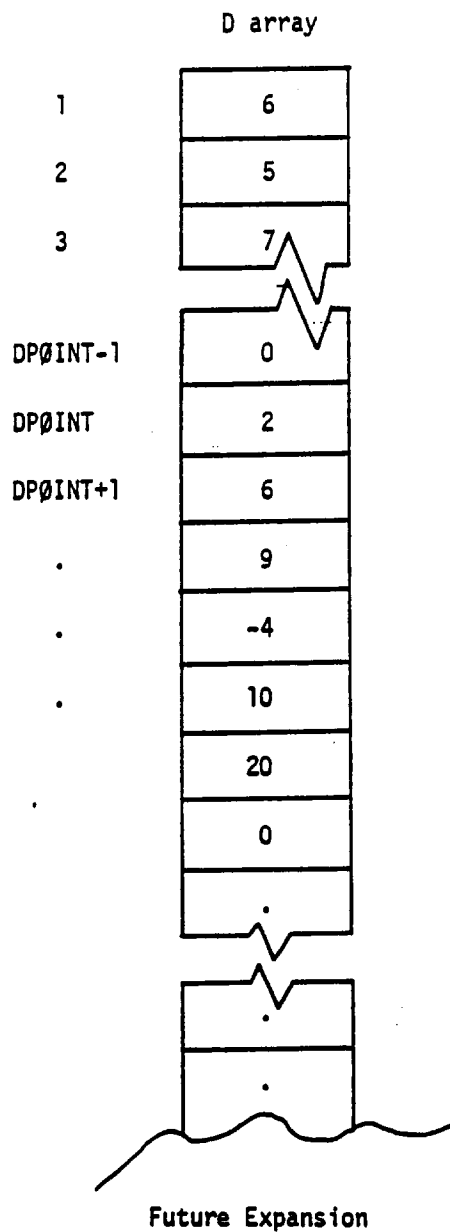
These are the 5 line format numbers which make up the heading format for the type of data currently being processed.

MODULE FUNCTIONAL DESCRIPTIONS



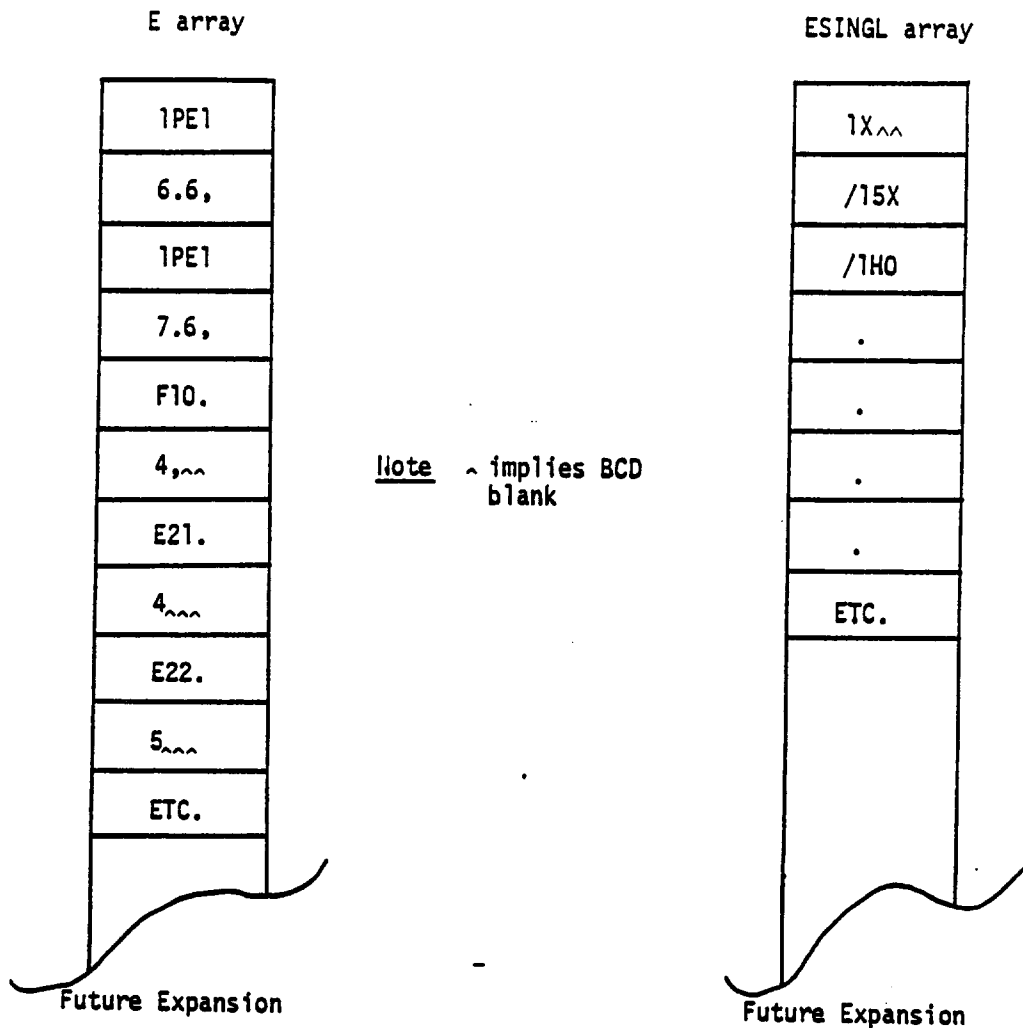
3. Word DPØINT of the D array defines the beginning of a string of pointers into the E array. This string is terminated by the first word containing a zero. Each word of this string thus defines a string of words in the E array which contains Hollerith data for construction of a format. Should a word in the D array be negative, the absolute value is used to point into the ESINGL array which contains Hollerith data also.

OUTPUT MODULE ØFP (OUTPUT FILE PROCESSOR)



4. The E array contains Hollerith data pertaining to the output of a variable; the ESINGL array contains Hollerith data pertaining to spacing and carriage control only.

MODULE FUNCTIONAL DESCRIPTIONS



4.70.10 Diagnostic Messages

If, during some phase of outputting a data block, ØFP encounters an error condition, work on that data block will cease, a warning message will be printed, and a call to the NASTRAN table-printer for table printing of this data block will be made. ØFP will then continue processing the remaining input data blocks.

OUTPUT MODULE MATPRN (GENERAL MATRIX PRINTER)

4.71 OUTPUT MODULE MATPRN (GENERAL MATRIX PRINTER)

4.71.1 Entry Point: MATPRN.

4.71.2 Purpose

To print general matrix data blocks.

4.71.3 DMAP Calling Sequence

MATPRN KGG,PL,PG,B2PP,UPV// \$

4.71.4 Input Data Blocks

KGG - Any matrix data block.

PL - Any matrix data block.

PG - Any matrix data block.

B2PP - Any matrix data block.

UPV - Any matrix data block.

Notes:

1. Any or all input data blocks can be purged.
2. If any data block is not a matrix, the TABPT routine will be called.

4.71.5 Output

The non-zero band of each column of the input matrix is unpacked and is printed in single precision format on the system output file.

4.71.6 Parameters

None.

4.71.7 Method

Subroutine MATDUM is called for each non-purged input file.

4.71.8 Subroutines

MATDUM - See subroutine description, section 3.4.28.

OUTPUT MODULE MATGPR (DISPLACEMENT METHOD MATRIX PRINTER)

4.72 OUTPUT MODULE MATGPR (DISPLACEMENT METHOD MATRIX PRINTER)

4.72.1 Entry Point: MATGPR

4.72.2 Purpose

To print displacement method matrices, identifying values with external grid point numbers.

4.72.3 DMAP Calling Sequence

MATGPR GPL,USET,SIL,ANYMAT // C,N,CØLSET / C,N,RØWSET / V,N,PRTØPT=ALL \$

4.72.4 Input Data Blocks

GPL - Grid Point List (This may also be GPLD if extra points are present.)

USET - Displacement set definitions table (This may also be USETD if extra points are present.)

SIL - Scalar Index List (This may also be SILD if extra points are present.)

ANYMAT- Any displacement method matrix.

Notes:

1. Unless CØLSET = RØWSET = 'H', GPL, USET and SIL must be present.
2. If ANYMAT is purged, MATGPR will return.

4.72.5 Output Data Blocks

The non-zero terms of ANYMAT are given external identification and printed on the system output file.

4.72.6 Parameters

CØLSET - Input-BCD-no default. CØLSET indicates the set to which the columns of ANYMAT belong. If CØLSET is not one of the following: M,Ø,R,SG,SB,L,A,F,S,N,G,E,P,NE,FE,D,H then MATGPR will return.

RØWSET - Input-BCD-default = X. RØWSET indicates the set to which the rows of ANYMAT belong. If RØWSET is not a legal set name, RØWSET = CØLSET.

PRTØPT - Input-BCD-default=ALL. If PRTØPT = null, only the null columns will be printed and identified. Otherwise, all columns will be so printed.

MODULE FUNCTIONAL DESCRIPTIONS

4.72.7 Method

The BCD parameters CØLSET and RØWSET are converted to bit positions in USET. CØLSET must be one of the following symbols: M,Ø,R,SG,SB,L,A,F,S,N,G,E,P,NE,FE,D,H,PS,SA,K,PA or else MATGPR will return. If RØWSET is not a legitimate symbol RØWSET = CØLSET.

GPL, USET, and SIL are placed in core. Each column and non-zero row element is identified according to the following scheme:

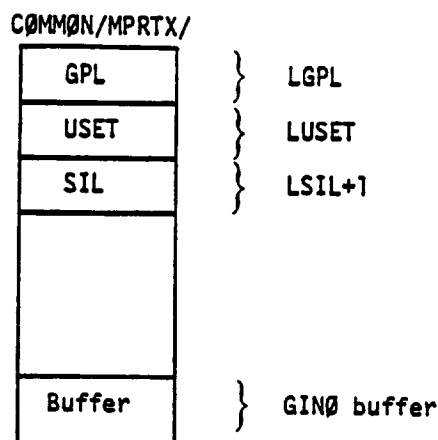
1. USET is searched for the number of members belonging to the g set (p set if USETD is used) before the current member of the matrix set.
2. This number is looked up in SIL to obtain the internal grid point number and type of point (scalar, grid, or extra).
3. The internal grid point number points into GPL for the external ID.

4.72.8 Subroutines

MATGPR has no auxiliary subroutines.

4.72.9 Design Requirements

1. Open core is defined at /MPRTX/.
2. Layout of open core is as follows:



4.72.10 Diagnostic Messages

MATGPR may issue the following diagnostic messages:

3007 and 3008.

OUTPUT MODULE MATPRT (MATRIX PRINTER)

4.73 OUTPUT MODULE MATPRT (MATRIX PRINTER)

4.73.1 Entry Point: PRTINT

4.73.2 Purpose

To print a matrix data block.

4.73.3 DMAP Calling Sequence

MATPRT X//C,N,RC/C,N,Y \$

4.73.4 Input Data Block

X - Matrix data block to be printed. If X is purged, then nothing is done.

4.73.5 Output Data Blocks

None.

4.73.6 Parameters

RC - Indicates whether X is stored by rows (RC = 1) or columns (RC = 0) - integer - input.

Y - Indicates whether X is to be printed (Y < 0, do not print X; Y > 0, print X)

- integer - input - default value = -1.

4.73.7 Method

Each column (or row) of the matrix is broken into groups of 6 terms (3 terms if complex) per printed line. If all the terms in a group = 0, the line is not printed. If the entire column (or row) = 0, it is not printed. If the entire matrix = 0, it is not printed.

4.73.8 Subroutines

4.73.8.1 Subroutine Name: INTPRT

1. Entry Point: INTPRT

2. Purpose: To print a matrix data block using subroutine MATPRT.

3. Calling Sequence: CALL INTPRT (A,CR,Ø,NAME)

MODULE FUNCTIONAL DESCRIPTIONS

where:

- A - Storage for 1 column (row) of the matrix + 1 GINØ buffer.
- CR - { 0 if the matrix is stored by columns.
1 if the matrix is stored by rows.
- Ø - { 0 if the matrix is not to be printed.
1 if the matrix is to be printed.
- NAME - 8 character name of the matrix (2 words, 4 characters per word).

3. Method: Subroutine MATPRT is called to print the matrix. Whenever MATPRT returns for a matrix name or column/row id to be printed, the name of the matrix (NAME₁, NAME₂) or the column or row id (as indicated by 'CR'), is printed.

4.73.8.2 Subroutine Name: MATPRT

1. Entry Points: MATPRT, PRTMAT
2. Purpose: To print a matrix data block.
3. Calling Sequence: CALL MATPRT (\$N₁, \$N₂, A, ØPT, CØLNUM)

CALL PRTMAT (\$N₁, \$N₂)

CØMMØN/XXMPRT/MCB(7)

where:

- N₁ - FØRTRAN statement number defining the return executed whenever a new page has been started (the calling program is expected to print the matrix and column id. CØLNUM = current column number).
- N₂ - FØRTRAN statement number defining the return executed whenever the column id must be printed in the middle of a page (CØLNUM = current column number).
- A - Storage for 1 column of the matrix + one GINØ buffer.
- ØPT - See subroutine description for VECPT, below, for the explanation of this argument.
- CØLNUM - Current column number being printed (output).
- MCB - Matrix control block.

OUTPUT MODULE MATPRT (MATRIX PRINTER)

3. Method: The matrix is unpacked and printed one column at a time. Whenever either of the nonstandard returns ($\$N_1, \N_2) is executed, the calling program must call PRTMAT to continue the printing of the matrix.

4. Additional Subroutines Called: VECprt.

4.73.8.3 Subroutine Name: VECprt

1. Entry Points: VECprt, PRTVEC

2. Purpose: To print a vector.

3. Calling Sequence: CALL VECprt ($\$N_1, \$N_2, P, N, A, \emptyset PT$)

CALL PRTVEC ($\$N_1, \N_2)

where:

N_1 - FORTRAN statement number defining the return executed whenever a new page has been started (the calling program is expected to print the vector id and any other subtitles desired).

N_2 - FORTRAN statement number defining the return executed whenever the vector id is to be printed in the middle of a page.

P - Vector type and precision.

N - Number of components in the vector.

A - Location of the vector.

$\emptyset PT$ - $\left\{ \begin{array}{l} = 0 \text{ if all the vector components are to be printed, regardless of its sparsity, and if it is to be printed starting on a new page if it will not fit on the current page.} \\ = +1 \text{ if only the printed lines which would have at least one non-zero component are to be printed, and if the vector is to be printed starting on a new page if it will not fit on the current page.} \\ = -1 \text{ if only the printed lines which would have at least one non-zero component are to be printed, and if as much of the vector as possible is to be printed on the current page.} \end{array} \right.$

3. Method: The vector will be printed as a single precision real or complex vector. The components will be printed 6 per line if real, 3 per line if complex. In addition, the first and last components of each line will be identified on each side of the line by their respective component members. In addition whenever either of the nonstandard

MODULE FUNCTIONAL DESCRIPTIONS

returns ($\$N_1, \N_2) is executed the calling program must call PRTVEC to continue the printing of the vector.

4. Additional Subroutines Called: FØRMAT.

4.73.8.4 Subroutine Name: FØRMAT

1. Entry Point: FØRMAT

2. Purpose: To print a line of 1 to 6 real numbers (optionally centered) preceded and followed by integer id's of the first and last number printed.

3. Calling Sequence: CALL FØRMAT (A,N1,N2,N3,L1,L2)

where:

A - Array from which the 1 to 6 real numbers are to be printed.

N1 - Index of the 1st number in the array to be printed.

N2 - Index of the last number in the array to be printed.

N3 - Increment to be used in extracting the 2nd, 3rd, etc., numbers in the array to be printed.

L1 - Integer id of the 1st number to be printed.

L2 - Integer id of the last number to be printed.

3. Method: If L1 and L2 are both positive, the numbers will be centered on the page. If either L1 or L2 is not positive, the numbers will be printed based upon the centering of 6 numbers.

4.73.9 Design Requirements

Open core is defined at /XXPRTI/. Open core contains one GINØ buffer followed by one unpacked real or complex single precision column of the matrix.

4.74 OUTPUT MODULE SEEMAT (PICTORIAL MATRIX PRINTER)

4.74.1 Entry Point: SEEMAT

4.74.2 Purpose

To show nonzero matrix elements on printer or plotter output positioned pictorially by row and column within the outlines of the matrix.

4.74.3 DMAP Calling Sequence

```
SEEMAT  M1,M2,M3,M4,M5//C,N,{PRINT  
      M0DELB1/C,N,M0DELN2/C,N,M0DELB2 $
```

Note that parameters PL0TTER, M0DELN1, M0DELB1, M0DELN2 and M0DELB2 are all described in paragraph 4 in section 4.74.6.

4.74.4 Input Data Blocks

M1	}	Matrix Data Blocks, any of which may be purged.
M2		
M3		
M4		
M5		

4.74.5 Output Data Blocks

None. The formatted matrix picture is output on the system output file or on a plot tape depending on the value of the first parameter.

4.74.6 Parameters

1. PRINT implies use of the system output file. (Any value other than PL0T implies PRINT). PL0T implies use of one of the plotters. Either of the plotter tapes PLT1 or PLT2 will be used, depending on the type of plotter requested.
2. PFILE is the plot number (Used only if first parameter is PL0T).

Input/output variable integer parameter. Frame or sheet number. The value of this parameter will be incremented by one (1) for each frame (sheet) created by SEEMAT.

The default value for this parameter is 0.

3. PACK is reserved for a future modification that will allow the representation of a nonzero block of the matrix with a single character. This parameter may be specified as C,N only. (see example in paragraph 4 below)

MODULE FUNCTIONAL DESCRIPTIONS

4. Plotter Name and Model Identification (Used only if parameter 1 = PLØT.)

Each plotter name has associated with it two model identifiers. Each of these model identifiers may either be an integer (MØDELNi) value or BCD (MØDELBi) value. If model identifier "i" (i = 1, 2) is an integer, insert its value for MØDELNi; if model identifier "i" (i = 1, 2) is BCD, insert its value for MØDELBi. In either case, specify the other value for model identifier "i" (MØDELBi and MØDELNi, respectively) as C,N only.

Below is a list of model identifiers allowable for each plotter name. A detailed explanation of this list can be found in section 4 of the User's Manual. Each plotter has associated with it a default model and several optional models. The model underlined is the default model. To access this default model, do not specify any of the last four DMAP parameters. For example to specify the CALCØMP 765, 205 (see section 4 of the User's Manual) the following DMAP statement may be used:

SEEMAT M1,M2,M3,M4,M5//C,N,PLØT/V,N,PFILE/C,N/C,N,CALCØMP \$

<u>Plotter Name</u>	<u>Model Identifiers</u>
BL	{ <u>LTE,30</u> } {STE,30}
EAI	{ <u>3500,300</u> } {3500,45}
SC	<u>4020,0</u>
CALCØMP	(<u>765,205</u>) 765,210 765,105 765,110 763,205 763,210 763,105 763,110 565,205 565,210 565,105 565,110 565,305 565,310 563,205 563,210

<u>Plotter Name</u>	<u>Model Identifiers</u>
	$\begin{pmatrix} 563,105 \\ 563,110 \\ 563,305 \\ 563,310 \end{pmatrix}$
DD	<u>80,B</u>
NASTRAN	$\begin{pmatrix} M,0 \\ T,0 \\ D,0 \\ M,1 \\ T,1 \\ D,1 \end{pmatrix}$

where:

BL is a Benson-Lehner plotter
 EAI is an Electronic Associates plotter
 SC is a Stromberg Carlson plotter
 CALCOMP is a California Computer plotter
 DD is a Data Display plotter
 NASTRAN is the NASTRAN general purpose plotter

4.74.7 Method

The matrix is partitioned into blocks which can be printed on a single sheet of output paper or plotted on a single frame or sheet of plotter output media. Only blocks containing nonzero elements will be printed. Row and column indices are indicated. The user of this module is cautioned to make sure that his line count limit is large enough. A default of 20,000 lines is provided by NASTRAN. This may be changed via the statement "MAXLINES = value" in the NASTRAN Case Control Deck. The transpose of the matrix is always printed.

The columns of the matrix are examined for nonzero terms. Let the matrix be partitioned into blocks, where a block consists of NL columns and 100 rows, where NL is the number of data lines per page obtained from /SYSTEM/. For each block containing nonzero terms, a BCD block image is stored in open core in packed bit format. Only blocks containing nonzero terms are stored. When NL columns have been passed, the blocks containing nonzero terms are printed on

MODULE FUNCTIONAL DESCRIPTIONS

the system output file or plotted. Note that since NASTRAN matrices are stored by column, the transpose of the matrix is what actually appears on the printed or plotted output. Blocks used for the first NL columns may now be re-used for subsequent groups of NL columns. This process is continued until all columns of the matrix have been processed. As many as five matrices may be handled during a single call to SEEMAT.

4.74.8 Subroutines

The plotter environment subroutines are utilized by SEEMAT. See section 3.4 for descriptions of the plotter utility routines.

4.74.8.1 Subroutine MAPSET

1. Entry Points: MAPSET, MAP
2. Purpose: Converts physical units to plotter units for module SEEMAT.
3. Calling Sequence: CALL MAPSET (X1,Y1,X2,Y2,KI1,KJ1,KI2,KJ2,L)
CALL MAP (X,Y,KI,KJ)

X,Y,Xi,Yi = Physical coordinates
KI,KJ,KI1,KJ1 = Plotter coordinates
L = Output Format Flag, input
 1 = KI,KJ are integer
 2 = KI,KJ are real

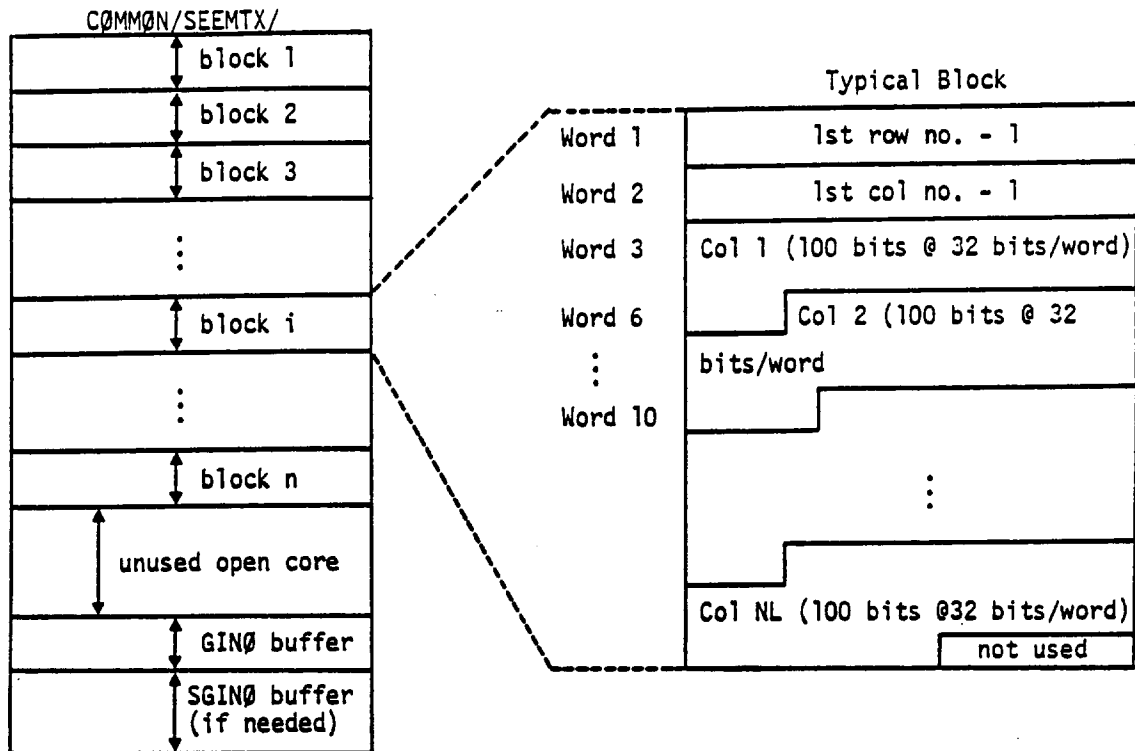
The meaning of i follows:

i = 1 point is lower left corner of frame
i = 2 point is upper right corner of frame
i = blank point is an arbitrary point on frame.

MODULE FUNCTIONAL DESCRIPTIONS

4.74.9 Design Requirements

4.74.9.1 Open Core Design



OUTPUT MODULE SEEMAT (PICTORIAL MATRIX PRINTER)

4.74.9.2 Data Requirements and Restrictions

1. All nonpurged input data blocks must be matrices. Error diagnostics will occur in the unpacking routines if an attempt is made to input a table data block to SEEMAT.
2. If the number of blocks needed overflows the available open core (e.g., a large full matrix can do this), a nonfatal diagnostic message will be output on the System Output File and processing for that matrix will be terminated. The user may decrease NL by adding a Case Control Card LINE = NL as a means of overcoming this restriction (printer only). Since the type of matrix for which one is interested in seeing the topology is usually sparse and at least partially banded, this restriction should not prove serious.

4.74.10 Diagnostic Messages

Diagnostic conditions detected by SEEMAT are nonfatal and result in appropriate error messages and termination of the processing of the current input matrix data block. The one exception is the condition of no open core for GINØ buffers, which should not occur in practice.

OUTPUT MODULE TABPT (TABLE PRINTER)

4.75 OUTPUT MODULE TABPT (TABLE PRINTER)

4.75.1 Entry Point: TABPT.

4.75.2 Purpose

To print table data blocks.

4.75.3 DMAP Calling Sequence

TABPT TAB1,TAB2,TAB3,TAB4,TAB5// \$

4.75.4 Input Data Blocks

TAB1 - Any NASTRAN data block.

TAB2 - Any NASTRAN data block.

TAB3 - Any NASTRAN data block.

TAB4 - Any NASTRAN data block.

TAB5 - Any NASTRAN data block.

Note: Any or all input data blocks can be purged.

4.75.5 Output

Each word in a data block is identified as real, BCD, or integer and printed on the system output file. The trailer data is also printed.

4.75.6 Parameters

None.

4.75.7 Method

Subroutine TABPRT is called for each non-purged input file.

4.75.8 Subroutines

TABPRT - See subroutine description, section 3.4.29.

OUTPUT MODULE PRTMSG (MESSAGE WRITER)

4.76 OUTPUT MODULE PRTMSG (MESSAGE WRITER)

4.76.1 Entry Point: PRTMSG

4.76.2 Purpose

To process a data block of user-oriented messages.

4.76.3 DMAP Calling Sequence

PRTMSG MSG// \$

4.76.4 Input Data Blocks

MSG - Messages to be printed (if purged, nothing is done).

4.76.5 Output Data Blocks

None

4.76.6 Parameters

None

4.76.7 Method

In addition to messages, the MSG data block may contain titles and subtitles. Before the first message is printed, a new page is started. From then on, a message count is maintained so as to start another new page when the maximum number of lines per page is exceeded. All messages are assumed to be only one line long. However, there is logic included to provide for messages of more than one line, forcing a new page at any time, and the alteration of titles and subtitles at any time. The description below of subroutine WRTMSG details all the included logic capability.

4.76.8 Subroutines

PRTMSG uses the utility routines EJECT and FREAD (see sections 3.4.62 and 3.4.15).

4.76.8.1 Subroutine Name: PRTMSG

MODULE FUNCTIONAL DESCRIPTIONS

1. Entry Point: PRTMSG

2. Purpose: To print the user messages in the MSG data block.

3. Calling Sequence: CALL PRTMSG

COMMON/OUTPUT/TITLE(32,6) - See OUTPUT miscellaneous table description in section 2.5.

Where:

TITLE = NASTRAN title, subtitle, label, and three extra subtitles.

4. Method: Open the MSG data block, and skip record 0. If the MSG data block does not exist, nothing else is attempted. Otherwise, the three extra subtitles are set to all blanks, and WRTMSG is called.

4.76.8.2 Subroutine Name: WRTMSG (General purpose subroutine)

1. Entry Point: WRTMSG

2. Purpose: To process a data block of user-oriented messages.

3. Calling Sequence: CALL WRTMSG (MSG)

COMMON/SYSTEM/ - See SYSTEM table description in section 2.4.1.8.

Where:

MSG = GINØ file name of the MSG data block.

and in /SYSTEM/

MAXLIN = Maximum number of lines permitted per page.

CØUNT = Number of lines thus far printed on the current page.

4. Method:

a. Save the current NASTRAN title, subtitle, and label. Force the first message to start on a new page (CØUNT=MAXLIN).

b. Read one word (N) from MSG. If an end-of-record condition occurs, force the first message in the next record to start on a new page (CØUNT=MAXLIN). Then repeat this step.

c. If $N < 0$, the next 32 words are assumed to be a replacement for TITLE (1-32,N). Force the next message to start on a new page (CØUNT=MAXLIN). Repeat step b.

OUTPUT MODULE PRTMSG (MESSAGE WRITER)

- d. If $N > 0$, a list and format follow. The next N items are assumed to be the list items. If $N = 0$, only a format follows.
- e. Read one word (NF) from MSG. If $NF < 0$, NF = number of lines to be generated by this message (repeat this step). If $NF = 0$, this message will be printed starting on a new page. (CØUNT=MAXLIN, repeat this step). Unless otherwise instructed, this subroutine assumes that each message will generate only 1 line of output. In either case, integer function EJECT is called to maintain the page line count. If this message will not fit on this page, any extra title(s) explicitly specified are printed below the NASTRAN title, subtitle, and label.
- f. If $NF > 0$, NF = size of the format (the format must be a continuous string of characters, contrary to the usual NASTRAN method of specifying at most 4 characters per word). The next NF words are assumed to be the format to be used with the list items read in step d, in one the following FØRTRAN statements:

WRITE (MØ,FØR) [if no list is specified].
or
WRITE (MØ,FØR) (LIST(I), I = 1,N)

Repeat step b.

- g. When the end of the MSG data block is encountered in step b, the NASTRAN title, subtitle, and label are restored, and the MSG data block is closed with a rewind.

5. Design Requirements:

- a. The message data block (MSG) must be opened before calling WRTMSG.
- b. In general, a set of messages is one record of the data block. Each set of messages will start on a new page.

4.76.9 Design Requirements

Open core is defined at /XXPMSG/, and is used only for one GINØ buffer which is defined at the beginning of open core.

OUTPUT MODULE PRTPARM (PARAMETER AND DMAP MESSAGE PRINTER)

4.77 OUTPUT MODULE PRTPARM (PARAMETER AND DMAP MESSAGE PRINTER)

4.77.1 Entry Point: PRTPRM

4.77.2 Purpose

To print parameter values and DMAP messages.

4.77.3 DMAP Calling Sequence

PRTPARM //C,N, α /C,N, β \$

4.77.4 Input Data Blocks

None

4.77.5 Output Data Blocks

None

4.77.6 Parameters

α - Integer value (no default value)

β - BCD value (default value = XXXXXXXX)

4.77.7 Method

As a parameter printer, use $\alpha = 0$. There are two options:

1. β = parameter name will cause the printout of the value of that parameter.

Example: PRTPARM //C,N,0/C,N,LUSET \$

2. β = XXXXXXXX will cause the printout of the values of all parameters in the current XVPS. Since this is the default value, it need not be specified.

Example: PRTPARM //C,N,0 \$

As a DMAP message printer, use $\alpha \neq 0$. There are two options:

1. $\alpha > 0$ causes the printout of the j^{th} message of category β where $j = |\alpha|$ and β is one of the values shown below. (The number of messages available in each category is also shown).

Example: PRTPARM //C,N,1/C,N,DMAP \$

MODULE FUNCTIONAL DESCRIPTIONS

2. $\alpha < 0$ causes the same action as $\alpha > 0$ with the additional action of program termination. Thus, PRTPARM may be used as a fatal message printer.

Example: PRTPARM //C,N,-2/C,N,PLA \$

4.77.8 Remarks

1. β is always a value.

2. TABLE OF β CATEGORY VALUES

<u>Rigid Format</u>	<u>Value of Beta</u>	<u>Number of Messages</u>
Static Analysis	STATICS	4
Static Analysis with Inertia Relief	INERTIA	4
Normal Mode Analysis	MØDES	4
Static Analysis with Differential Stiffness	DIFFSTIF	6
Buckling Analysis	BUCKLING	7
Piecewise Linear Analysis	PLA	6
Direct Complex Eigenvalue Analysis	DIRCEAD	5
Direct Frequency and Random Response	DIRFRRD	6
Direct Transient Response	DIRTRD	5
Modal Complex Eigenvalue Analysis	MDLCEAD	6
Modal Frequency and Random Response	MDLFRRD	7
Modal Transient Response	MDLTRD	7
DMAP	DMAP	1

3. For details on error messages for the i^{th} Rigid Format see section 3.(1 + 1).2 in the User's Manual.

4.77.9 Subroutines

PRTPRM has no auxiliary subroutines.

OUTPUT MODULE PRTPARM (PARAMETER AND DMAP MESSAGE PRINTER)

4.77.10 Diagnostic Messages

Values of α and β inconsistent with the above under "Method" will result in diagnostic messages from PRTPARM.

MATRIX MODULE ADD (ADD TWO MATRICES)

4.78 MATRIX MODULE ADD (ADD TWO MATRICES)

4.78.1 Entry Point: DADD

4.78.2 Purpose

To compute $[C] = \alpha[A] + \beta[B]$.

4.78.3 DMAP Calling Sequence

ADD A,B/C,C,Y,ALPHA=(1.0,2.0)/C,Y,BETA=(3.0,4.0) \$

4.78.4 Input Data Blocks

A - Any matrix \neq B

B - Any matrix \neq A

Note: A and/or B can be purged.

4.78.5 Output Data Blocks

C - Matrix.

The type of C is maximum of the types of A, B, α , β . The shape of C is the shape of A if A is present. Otherwise it is that of B.

Note: C cannot be purged.

4.78.6 Parameters

ALPHA - Input-complex default value = 1.0. This is the scalar multiplier for A.

BETA - Input-complex default value = 1.0. This is the scalar multiplier for B.

Note: If $\text{Im}(\alpha)$ or $\text{Im}(\beta) = 0.0$, the parameter will be considered real.

4.78.7 Method

If [A] is not purged, the number of columns, rows, and form of [C] = number of columns, rows, and form of [A]. Otherwise the [B] descriptors are used. The type of [C] is the maximum compatible type of [A], [B], ALPHA and BETA. ALPHA and BETA are assumed to be real if their imaginary parts are zero.

MODULE FUNCTIONAL DESCRIPTIONS

4.78.8 Subroutines

ADD - See subroutine description, Section 3.5.10.

4.78.9 Design Requirements

Open core is defined at /DADDA/.

4.78.10 Diagnostic Messages

None.

MATRIX MODULE MPYAD (MULTIPLY ADD)

4.79 MATRIX MODULE MPYAD (MULTIPLY ADD)

4.79.1 Entry Point: DMPYAD

4.79.2 Purpose

MPYAD performs the multiplication of two matrices and, optionally, addition of a third matrix to the product: $[D] = [A][B] \pm [C]$

4.79.3 DMAP Calling Sequence

MPYAD A,B,C/D/C,N,T/C,N,SIGNAB/C,N,SIGNC/C,N,TYPE \$

4.79.4 Input Data Blocks

- A - Left hand matrix in the matrix product $[A][B]$
- B - Right hand matrix in the matrix product $[A][B]$
- C - Matrix to be added to $[A][B]$

Notes:

1. If no matrix is to added, C must be purged.
2. A, B, C must be physically different data blocks.
3. A and B must not be purged.
4. Either A or B (but not both) may be a NASTRAN diagonal matrix. In this case, C must be purged.

4.79.5 Output Data Block

- D - Matrix resulting from the MPYAD operation.

Note: D may not be purged.

4.79.6 Parameters

T - Integer-input-no default.

T = $\begin{cases} 1, & \text{perform } [A]^T[B] \\ 0, & \text{perform } [A][B] \end{cases}$

MODULE FUNCTIONAL DESCRIPTIONS

SIGNAB - Integer-input-no default.	SIGNAB = $\begin{cases} +1, \text{ perform } [A][B] \\ -1, \text{ perform } -[A][B] \end{cases}$
SIGNC - Integer-input-no default.	SIGNC = $\begin{cases} +1, \text{ add } [C] \\ -1, \text{ subtract } [C] \end{cases}$
TYPE - Integer-input-default = 0.	TYPE = $\begin{cases} 0, \text{ logical choice based on input matrices} \\ 1, \text{ elements of } [D] \text{ will be output in real single precision} \\ 2, \text{ elements of } [D] \text{ will be output in real double precision.} \\ 3, \text{ elements of } [D] \text{ will be output in complex single precision.} \\ 4, \text{ elements of } [D] \text{ will be output in complex double precision} \end{cases}$

4.79.7 Examples

1. $[D] = [A][B] + [C]$ (D double precision)
 MPYAD A,B,C / D / C,N,0 / C,N,1 / C,N,2 \$
2. $[D] = [A]^T[B] - [C]$ (D single precision)
 MPYAD A,B,C / D / C,N,1 / C,N,1 / C,N,-1 / C,N,1 \$
3. $[D] = -[A][B]$ (D double precision)
 MPYAD A,B, / D / C,N,0 / C,N,-1 / C,N,0 / C,N,2 \$

4.79.8 Method

DMPYAD reads the trailers for the data blocks A, B and C. /MPYADX/ is initialized. If neither [A] nor [B] is diagonal, MPYAD is called, the trailer for D is written, and the module exits. Otherwise, /DMPYX/ is initialized, and DMPY is called to perform the diagonal multiplication. If the matrix [C] is present, /ADDX/ is initialized, and ADD is called to perform the matrix addition. The trailer for D is written and the module exits.

4.79.9 Subroutines

DMPYAD calls the following matrix operations:

MPYAD (see Section 3.5.12 for details)

DMPY (see Section 3.5.21 for details)

ADD (see Section 3.6.10 for details).

MATRIX MODULE MPYAD (MULTIPLY ADD)

4.79.10 Design Requirements

4.79.10.1 Allocation of Core Storage

See descriptions for MPYAD, DMPY and ADD.

4.79.10.2 Environment

The module MPYAD consists of one subroutine, DMPYAD. One scratch file is used. Three common blocks define open core, one for each of the three overlay segments containing the matrix operations:

/MPYA1D/ included at end of segment containing MPYAD.

/MPYA2D/ included at end of segment containing DMPY.

/MPYA3D/ included at end of segment containing ADD.

MATRIX MODULE SØLVE (SOLVES THE MATRIX EQUATION $[A][X] = [B]$)

4.80 MATRIX MODULE SØLVE (SOLVES THE MATRIX EQUATION $[A][X] = [B]$)

4.80.1 Entry Point: SØLVE

4.80.2 Purpose

To solve the matrix equation,

$$[A][X] = \pm [B] \quad (1)$$

4.80.3 DMAP Calling Sequence

SØLVE A,B / X / V,Y,SYM / V,Y,SIGN / V,Y,PREC / V,Y,TYPE \$

4.80.4 Input Data Blocks

A - A square real or complex matrix.

B - A rectangular matrix.

Note: If B is purged, the identity matrix is assumed.

4.80.5 Output Data Blocks

X - A rectangular matrix.

4.80.6 Parameters

SYM - Input-integer-default = 0	$\left\{ \begin{array}{l} -1 - \text{use unsymmetric decomposition} \\ 1 - \text{use symmetric decomposition} \\ 0 - \text{logical choice based on input matrix [A].} \end{array} \right.$
- Output	value actually used for SYM
SIGN - Input-integer-default = 1	$\left\{ \begin{array}{l} 1 - \text{solve } [A][X] = [B] \\ -1 - \text{solve } [A][X] = -[B] \end{array} \right.$
PREC - Input-integer-default = 0	$\left\{ \begin{array}{l} 0 - \text{logical choice based on input matrices} \\ 1 - \text{use single precision arithmetic} \\ 2 - \text{use double precision arithmetic} \end{array} \right.$
- Output	value actually used for PREC
TYPE - Input-integer-default = 0	$\left\{ \begin{array}{l} 0 - \text{logical choice based on input matrices} \\ 1 - \text{output type of matrix [X] is real single precision} \\ 2 - \text{output type of matrix [X] is real double precision} \end{array} \right.$
- Output	value actually used for TYPE

MODULE FUNCTIONAL DESCRIPTIONS

- 3 - output type of matrix [X] is complex
single precision
- 4 - output type of matrix [X] is complex
double precision

4.80.7 Method

Depending upon the SYM flag and the type of [A], either SDCOMP CDCOMP, or DECOMP is called to form

$$[A] = [L][U] . \quad (2)$$

FBS or GFBS is called to solve

$$[L][Y] = \pm [B] , \quad (3)$$

and

$$[U][X] = [Y] . \quad (4)$$

4.80.8 Subroutines

The above mentioned subroutines are the only ones called by SOLVE and are documented in section 3.5.

4.80.9 Design Requirements

The appropriate subroutines should be referenced for the design requirements peculiar to each routine.

4.80.10 Diagnostic Messages

The individual routines should be referred to for diagnostic messages.

MATRIX MODULE DECØMP (MATRIX DECOMPOSITION)

4.81 MATRIX MODULE DECØMP (MATRIX DECOMPOSITION)

4.81.1 Entry Point: DDCØMP

4.81.2 Purpose

To decompose a square matrix [A] into lower [L] and upper [U] triangular factors.

4.81.3 DMAP Calling Sequence

DECØMP A/L,U/V,Y,KSVM/V,Y,CHØLSKY/V,N,MINDIAG/V,N,DET/V,N,PØWER/V,N,SING \$

4.81.4 Input Data Blocks

A - A square matrix.

4.81.5 Output Data Blocks

L - Lower triangular factor of [A].

U - Non-standard upper triangular factor of [A].

4.81.6 Parameters

KSVM - Input-integer-no default. 1, use symmetric decomposition. 0, use unsymmetric decomposition.

CHØLSKY - Input-integer-default = 0. 1, use Cholesky decomposition. 0, do not use Cholesky decomposition.

MINDIAG - Output-real-no default. The minimum diagonal term of [U].

DET - Output-complex single precision-no default. The scaled value of the determinant of [A].

PØWER - Output-integer-no default. Integer PØWER of 10 by which DET should be multiplied to obtain the determinant of [A].

SING - Output-integer-no default. SING is set to -1 if [A] is singular.

4.81.7 Method

Depending upon the type of [A] and the KSVM flag, a calling sequence is set up, and either

MODULE FUNCTIONAL DESCRIPTIONS

CDCOMP, DECOMP, or SDCOMP is called.

4.81.8 Subroutines

The major subroutines used are DECOMP, CDCOMP and SDCOMP. Descriptions of these subroutines can be found in sections 3.5.15, 3.5.16, and 3.5.14 respectively.

4.81.9 Design Requirements

The individual subroutine writeups should be consulted for the particular restrictions of each routine.

4.81.10 Diagnostic Messages

See the appropriate subroutine descriptions.

MATRIX MODULE FBS (FORWARD-BACKWARD SUBSTITUTION)

4.82 MATRIX MODULE FBS (FORWARD-BACKWARD SUBSTITUTION)

4.82.1 Entry Point: DFBS

4.82.2 Purpose

To solve the equation,

$$[L][U][X] = \pm [B] \quad (1)$$

where [L] and [U] are the upper and lower triangular factors obtained via matrix module DECØMP.

4.82.3 DMAP Calling Sequence

FBS L,U,B / X / V,Y,SYM / V,Y,SIGN / V,Y,PREC / V,Y,TYPE \$

4.82.4 Input Data Blocks

L,U - Matrices output from module DECØMP.

B - Rectangular matrix.

4.82.5 Output Data Blocks

X - Rectangular matrix.

4.82.6 Parameters

SYM - Input-integer-no default.

- Output

SIGN - Input-integer-no default.

PREC - Input-integer-no default

- Output

{ 0 - symmetric/unsymmetric if matrix [U]
 purged/not purged.
 1 - matrix [L][U] is symmetric
-1 - matrix [L][U] is unsymmetric

value actually used for SYM

{ 1 - solve [L][U][X] = [B]
-1 - solve [L][U][X] = -[B]

{ 0 - logical choice based on input matrices
 1 - use single precision arithmetic
 2 - use double precision arithmetic

value actually used for PREC

MODULE FUNCTIONAL DESCRIPTIONS

TYPE - Input-integer-no default.

- 0 - logical choice based on input matrices
- 1 - output [X] in single precision
- 2 - output [X] in double precision
- 3 - output [X] in complex single precision
- 4 - output [X] in complex double precision

- Output

value actually used for TYPE

4.82.7 Method

Depending upon the value of the parameter SYM, either FBS or GFBS is called.

4.82.8 Subroutines

The above routines are the only ones called by DFBS. Their descriptions are given in Section 3.5.17 for FBS and 3.5.19 for GFBS.

4.82.9 Design Requirements

The appropriate routines should be referenced for their individual requirements.

4.82.10 Diagnostic Messages

The individual subroutines should be referred to for the messages.

MATRIX MODULE PARTN (PARTITION A MATRIX)

4.83 MATRIX MODULE PARTN (PARTITION A MATRIX)

4.83.1 Entry Point: PARTN1

4.83.2 Purpose

To partition [A] into [A11], [A12], [A21] and [A22]:

$$[A] \Rightarrow \begin{bmatrix} A11 & | & A21 \\ \hline A12 & | & A22 \end{bmatrix}. \quad (1)$$

4.83.3 DMAP Calling Sequence

PARTN A,RP,CP/A11,A12,A21,A22/V,Y,SYM/V,Y,TYPE/V,Y,FØRM1/V,Y,FØRM2/V,Y,FØRM3/V,Y,FØRM4 \$

4.83.4 Input Data Blocks

A - Matrix to be partitioned.

RP - Row partitioning vector - single precision column vector.

CP - Column partitioning vector - single precision column vector.

Notes:

1. If A is purged, PARTN returns.
2. If RP is purged, A is partitioned as follows:

$$[A] \Rightarrow \begin{bmatrix} A11 \\ \hline A12 \end{bmatrix}. \quad (2)$$

3. If CP is purged and $SYM > 0$, A is partitioned as follows:

$$[A] \Rightarrow [A11; A21]. \quad (3)$$

4. If CP is purged and $SYM \leq 0$, A is partitioned as follows:

$$[A] \Rightarrow \begin{bmatrix} A11 & | & A21 \\ \hline A12 & | & A22 \end{bmatrix}, \quad (4)$$

where RP is used as both the row and column partitioner.

5. RP and CP cannot both be purged.

MODULE FUNCTIONAL DESCRIPTIONS

4.83.5 Output Data Blocks

A11 - Partition of A.
A12 - Partition of A.
A21 - Partition of A.
A22 - Partition of A.

Notes:

1. Any or all output data blocks can be purged.
2. For the shape of outputs (number of rows and columns) see Section 4.83.7 below.

4.83.6 Parameters

SYM - Input-integer-default=-1. SYM chooses between a symmetric partition and an unsymmetric partition. If $SYM \leq 0$, CP is used as RP. If $SYM > 0$, CP and RP are distinct.

TYPE - $\left\{ \begin{array}{l} \text{Input-integer-default} = 0. \text{ Type of output matrices } 0 \leq TYPE \leq 4; \text{ default is logical choice based on matrix A and requested partitions.} \\ \text{Output-integer} \quad \quad \quad \text{Type of output matrices produced.} \end{array} \right.$

FØRM1 $\left\{ \begin{array}{l} \text{Input-integer-default} = 0. \text{ Form of A11, } 0 < FØRM1 \leq 8; \text{ default uses logical choice based on matrix A.} \\ \text{Output-integer} \quad \quad \quad \text{Form of A11 produced.} \end{array} \right.$

FØRM2 - Same as FØRM1 but applied to A12.

FØRM3 - Same as FØRM1 but applied to A21.

FØRM4 - Same as FØRM1 but applied to A22.

4.83.7 Method

Let N1 = number of non-zero terms in RP.

Let N2 = number of non-zero terms in CP.

Let NRØWA = number of rows in A.

Let NCØLA = number of columns in A.

CASE:1 RP purged.

A11 is a $(NRØWA-N2) \times NCØLA$ matrix.

A12 is a $N2 \times NCØLA$ matrix.

A21 is not written.

A22 is not written.

MATRIX MODULE PARTN (PARTITION A MATRIX)

CASE 2: CP purged and $\text{SYM} > 0$.

A11 is a $\text{NRØWA} \times (\text{NCØLA} - \text{N1})$ matrix.

A12 is not written.

A21 is a $\text{NRØWA} \times \text{N1}$ matrix.

A22 is not written.

CASE 3: CP purged and $\text{SYM} \leq 0$.

A11 is a $(\text{NRØWA} - \text{N1}) \times (\text{NCØLA} - \text{N1})$ matrix.

A12 is a $\text{N1} \times (\text{NCØLA} - \text{N1})$ matrix.

A21 is a $(\text{NRØWA} - \text{N1}) \times \text{N1}$ matrix.

A22 is a $\text{N1} \times \text{N1}$ matrix.

CASE 4: Neither RP nor CP purged.

A11 is a $(\text{NRØWA} - \text{N2}) \times (\text{NCØLA} - \text{N1})$ matrix.

A12 is a $\text{N2} \times (\text{NCØLA} - \text{N1})$ matrix.

A21 is a $(\text{NRØWA} - \text{N2}) \times \text{N1}$ matrix.

A22 is a $\text{N2} \times \text{N1}$ matrix.

In general if $a_{ij} \in [A]$, then:

$a_{ij} \in [A11]$ if $\text{RP}(J) = \text{CP}(I) = 0$

$a_{ij} \in [A12]$ if $\text{CP}(I) \neq 0, \text{RP}(J) = 0$

$a_{ij} \in [A21]$ if $\text{CP}(I) = 0, \text{RP}(J) \neq 0$

$a_{ij} \in [A22]$ if $\text{CP}(I) \neq 0, \text{RP}(J) \neq 0$

4.83.8 Subroutines

4.83.8.1 Subroutine Name: PARTN2

1. Entry Point: PARTN2

2. Purpose: Initialization routine for PARTN1 and MERGE1. It calls PARTN3 to build the bit strings from the partitioning vectors CP and RP and sets default options based on SYM.

MODULE FUNCTIONAL DESCRIPTIONS

3. Calling Sequence: CALL PARTN2 (CP,RP,CØRE,BUF)

CP = GINØ file name of column partitioning vector
RP = GINØ file name of row partitioning vector
CØRE = Location of first word open core
BUF = Location of GINØ buffer.

4.83.8.2 Subroutine Name: PARTN3

1. Entry Point: PARTN3
2. Purpose: Builds bit strings as directed by PARTN2.
3. Calling Sequence: CALL PARTN3 (FILE,SIZE,ØNES,IZ,NZ,HERE,BUF,CØRE)

FILE = GINØ file name
SIZE = Length of partitioning vector
ØNES = Number of non-zero terms in vector
IZ = Pointer to working core
NZ = Length of working core
HERE = Logical Flag
BUF = Location of GINØ buffer
CØRE = Location of open core

4.83.9 Design Requirements

Open core is defined at /PARTN1/.

4.83.10 Diagnostic Messages

Messages 2166, 2167, 2168, 2169, 2170, 2171, 2172, 2173, 2174, 2175, 2176, 3002, and 3008 may be issued.

MATRIX MODULE MERGE (MERGE MATRICES TOGETHER)

4.84 MATRIX MODULE MERGE (MERGE MATRICES TOGETHER)

4.84.1 Entry Point: MERGE1

4.84.2 Purpose

To form

$$[A] \Leftarrow \left[\begin{array}{c|c} A11 & A21 \\ \hline A12 & A22 \end{array} \right]. \quad (1)$$

4.84.3 DMAP Calling Sequence

MERGE A11,A12,A21,A22,RP,CP/A/V,Y,SYM/V,Y,TYPE/V,Y,FORM \$

4.84.4 Input Data Blocks

A11 - Matrix \neq A12, A21, A22.

A12 - Matrix \neq A11, A21, A22.

A21 - Matrix \neq A11, A12, A22.

A22 - Matrix \neq A11, A12, A21.

RP - Row partitioning vector - single precision vector.

CP - Column partitioning vector - single precision vector.

Notes:

1. Any or all of A11, A12, A21, A22 can be purged which implies $[A_{IJ}] = [0]$.
2. RP and CP cannot both be purged.
3. See method section for meaning when RP or CP is purged.

4.84.5 Output Data Blocks

A - Merged matrix from A11, A12, A21, A22.

Notes: A cannot be purged.

MODULE FUNCTIONAL DESCRIPTIONS

4.84.6 Parameters

SYM	-	Input-integer-no default.	SYM \leq 0, CP is used as RP. SYM $>$ 0, CP and RP are distinct.
TYPE	-	$\left\{ \begin{array}{l} \text{Input-integer-default} = 0. \\ \text{Output-integer} \end{array} \right.$	Type of A, $0 \leq \text{TYPE} \leq 4$; default is logical choice based on types of input partitions and system precision flag. Type chosen for A.
FØRM	-	$\left\{ \begin{array}{l} \text{Input-integer-default} = 0. \\ \text{Output-integer.} \end{array} \right.$	Form of A, $0 \leq \text{FORM} \leq 8$; default is logical choice based on forms of input partitions. Form chosen for A.

4.84.7 Method

MERGE is the inverse of PARTN in the sense that if A11, A12, A21, A22 were produced by PARTN using RP, CP, FØRM, SYM, and TYPE from A, MERGE will reproduce A. See PARTN (Section 4.83) for options on RP, CP and SYM.

4.84.8 Subroutines

Subroutines PARTN2 and PARTN3 are used. These routines are described in Section 4.83.

4.84.9 Design Requirements

Open core is defined at /MERGE1/ .

4.84.10 Diagnostic Messages

Messages 2161, 2162, 2163, 2164, 2170, 2171, 2172, 2173, 2174, 2175, 2176, 3002, and 3008 may be issued.

MATRIX MODULE TRNSP (TRANPOSE A MATRIX)

4.85 MATRIX MODULE TRNSP (TRANPOSE A MATRIX).

4.85.1 Entry Point: DTRANP

4.85.2 Purpose

To form $[A]^T$ given $[A]$.

4.85.3 DMAP Calling Sequence

TRNSP A/AT \$

4.85.4 Input Data Blocks

A - Any matrix data block.

Note: If $[A]$ is purged, TRNSP returns.

4.85.5 Output Data Blocks

AT - The matrix transpose of $[A]$.

Note: AT cannot be purged.

4.85.6 Parameters

None.

4.85.7 Method

Subroutine TRNSP is called.

4.85.8 Subroutines

TRNSP - See subroutine description, section 3.5.25.

4.85.9 Design Requirements

Open core is defined at /DTRANX/. Eight scratch files are used.

MATRIX MODULE SMPYAD (STRING MULTIPLY ADD)

4.86 MATRIX MODULE SMPYAD (STRING MULTIPLY ADD)

4.86.1 Entry Point: SMPYAD

4.86.2 Purpose

To multiply a series of matrices together.

4.86.3 DMAP Calling Sequence

SMPYAD A,B,C,D,E,F/G/C,N,N/C,N,SIGNX/C,N,SIGNF/C,N,PG/C,N,TA/C,N,TB/C,N,TC/C,N,TD \$

4.86.4 Input Data Blocks

$\left. \begin{matrix} A \\ B \\ C \\ D \\ E \end{matrix} \right\}$ - Up to 5 matrices to be multiplied together, from left to right.

F - Matrix to be added to the above product.

Notes:

1. If one of the five multiplication matrices is required in the product (see parameter "N" below) and is purged, the multiplication will not be done.
2. If the F matrix is purged, no matrix will be added to the product.

4.86.5 Output Data Blocks

G - Resultant matrix (may not be pre-purged).

4.86.6 Parameters

- N - Number of matrices involved in the product - integer - input.
- SIGNX - Sign of the product matrix (e.g., [A][B][C][D][E]) - integer - input.
1 for plus, -1 for minus.
- SIGNF - Sign of the matrix to be added to the product matrix - integer - input.
1 for plus, -1 for minus.
- PG - Output precision of the final result - integer - input.
1 for single precision, 2 for double precision, 0 for logical choice based on input matrices.

MODULE FUNCTIONAL DESCRIPTIONS

TA }
TB } - Transpose indicators for the [A][B][C] and [D] matrices (1 if transposed
TC } matrix to be used in the product; 0 if untransposed) - integer - input.
TD }

Note: All the parameters except "N" have default values. They are these:

1. SIGNX = 1 (sign of product is plus)
2. SIGNF = 1 (sign of added matrix is plus)
3. PG = 1 (single precision result)
4. TA }
TB } = 0 (use untransposed [A], [B], [C], and [D] matrices in the product)
TC }
TD }

4.86.7 Method

The method is the same as for the MPYAD module with one exception and one addition:

1. None of the matrices may be diagonal.
2. Except for the final product, all intermediate matrix products are generated in double precision.

The matrices are multiplied together from right-to-left, i.e., the first product calculated is the product of matrix n-1 and matrix n.

4.86.8 Subroutines

MPYAD is called (see section 3.5.12 for details).

4.86.9 Design Requirements

1. Two scratch files are required.
2. Open core is the /MPYADX/ common block, the same one as used by the MPYAD module, (see section 4.79).

FOR MATERIAL PREVIOUSLY COVERED
IN SECTION 4.87,
SEE NEW SECTION 8

FOR MATERIAL PREVIOUSLY COVERED
IN SECTION 4.88,
SEE NEW SECTION 9.1

EXECUTIVE PREFACE MODULE IFP4 (INPUT FILE PROCESSOR - PHASE 4)

4.89 EXECUTIVE PREFACE MODULE IFP4 (INPUT FILE PROCESSOR - PHASE 4)

4.89.1 Entry Point: IFP4

4.89.2 Purpose

1. To convert the data card images related to fluid and hydroelastic analysis into conventional data card images as output from the IFP module.
2. To calculate data related to the boundaries and the harmonic degrees of freedom of the axisymmetric fluid and append this data to the MATPØØL data block.

4.89.3 Calling Sequence

CALL IFP4. IFP4, an Executive Preface Module, is called only by the Preface driver SEMINT.

4.89.4 Input Data Blocks

AXIC - Bulk Data Deck cards related to the hydroelastic problem as output from IFP.

GEØM1 - Grid Point and Coordinate System Data.

GEØM2 - Scalar Point and Element Connection Data.

GEØM4 - Constraint Data.

MATPØØL - Direct Input Matrix Data.

4.89.5 Output Data Blocks

Same as the Input Data Blocks.

4.89.6 Parameters

Not applicable to IFP4.

4.89.7 Method

IFP4 allocates the available core as it proceeds. Each type of data card image on the AXIC data block is read and used to form tables or new data card images. The new data and any existing data on the Input data blocks are merged and written on one of two scratch files. After the scratch file data are complete the data are then copied back on the Input/Output data files. (This is not normally allowed. The preface modules, however, have the privilege of writing on an input file.)

MODULE FUNCTIONAL DESCRIPTIONS

Data cards as referenced below refer to card images as found on the AXIC input data block. The actual data cards are read and first processed by the IFP. The fluid data card types found on the AXIC data block, and used by IFP4, are listed below along with a list of the output card images produced as a result of their presence, and the data blocks effected.

<u>IFP4 Input Card Image</u>	<u>IFP4 Output Card Image</u>	<u>Data Block Effected</u>
AXIF	none	all below
BDYLIST	data	MATPØØL
CFLUID2	CFLUID2	GEØM2
CFLUID3	CFLUID3	GEØM2
CFLUID4	CFLUID4	GEØM2
DMIAX	DMIG	MATPØØL
FLSYM	data (header)	MATPØØL
FREEPT	SPØINT MPC	GEØM2 GEØM4
FSLIST	CMFREE SPC	GEØM2 GEØM4
GRIDB	GRID	GEØM1
PRESPT	SPØINT MPC	GEØM2 GEØM4
RINGFL	GRID SEQGP	GEØM1 GEØM1

It should be noted that the output card images may be a function of several types of input card images as detailed in the following.

4.89.7.1 Data values found on the AXIF card are first stored in core for subsequent use. They are:

1. CS_f , the coordinate system number for the fluid system
2. g , the value of gravity
3. ρ_d , the default value of fluid density
4. B_d , the default value of the compression coefficient
5. $NØSYM$, an integer 0 or 1 indicating whether the unsymmetric (*series) terms are used in the computations.

EXECUTIVE PREFACE MODULE IFP4 (INPUT FILE PROCESSOR - PHASE 4)

6. A list of harmonic numbers n_j , $j = 1, 2, \dots, J$, indicating the harmonics to be formulated. If none are supplied by the user, it is implied that the fluid is not to be solved, however the processing of the boundary points (GRIDB) may be necessary as discussed later.

The list of harmonic numbers (n_j) are converted to an in core list of index numbers (I_j) as follows.

If $NOSYM=0$, implying only the symmetric series:

$$I_j = 2n_j + 2, \quad n_j \geq 0, \quad (1)$$

for $j = 1$ to J .

If $NOSYM=1$, implying the symmetric and unsymmetric (*) series:

$$I_j = 2n_j + 2, \quad n_j \geq 0, \quad (2)$$

for $j = 1$ to J , plus the additional indices:

$$I_j^* = 2n_j + 1, \quad n_j > 0, \quad (3)$$

for $j = 1$ to J .

The list of indices as formed above is sorted and held in core for subsequent use. The complete list of indices may thus be up to $2J$ in length. Henceforth the number of indices in the list is referred to as N .

4.89.7.2 GEOM1 Data Block Processing

1. The GEOM1 file is read and the coordinate system as specified by CS_f is located among the CØRD1C, CØRD2C, CØRD1S, or CØRD2S card images. Its type (cylindrical or spherical) is saved as a flag for use in later processing. If the coordinate system is not located among the above card types a fatal error is indicated to the user and a cylindrical type is assumed to permit further checking of data.
2. GRIDB card images are read and stored in core, 5 words per image. A GRIDB card image defines a normal grid point except that it's location is fixed to a fluid (RINGFL) point.
3. If any GRIDB card images are present IFP4 at this time forms a boundary list table in core.

MODULE FUNCTIONAL DESCRIPTIONS

4. For each fluid point, IDF_j , contained in a BDYLIST card image, a seven word entry is placed in the boundary list table. The contents of this entry are, using data from the BDYLIST card image:

- | | | |
|----------------|---|-------------------------------|
| 1. IDF_j | | |
| 2. 1 | } | Integer 1's (temporary flags) |
| 3. 1 | | |
| 4. 1 | | |
| 5. IDF_{j-1} | | |
| 6. IDF_{j+1} | | |
| 7. ρ_b | | |

where j indicates the respective IDF in the BDYLIST list of IDFs.

If IDF_{j-1} or IDF_{j+1} does not exist a zero (0) is entered.

If IDF_{j-1} or IDF_{j+1} is designated to be AXIS then a minus one (-1) is entered.

Should ρ_b as specified in the BDYLIST card image be missing, the default ρ_d , as specified on the AXIF card image, is used for position 7 of the entry. If both are missing, a user fatal error results. Missing is denoted by an integer -1.

After all BDYLIST card images have been processed and the entries added to the boundary list table, a sort is performed such that the entries are in sort by the primary IDF (found in the first position of each entry).

5. An initial "pass" of RINGFL card images is now made. The meridinal angles (x_2 for a cylindrical coordinate system or x_3 for a spherical system) must be zero and are checked. A binary search is performed to find one or more entries whose primary IDF matches the IDF of each RINGFL card image. When found the values $X1$, $X2$, and $X3$ of the respective images replace the three integer ones in position 2, 3, and 4 of that entry. If an entry is not found, a user fatal error is indicated.

If after all RINGFL card images have been passed, any of the entries in the boundary list table (residing in core) still contain the three integer ones in positions 2, 3, and 4, a user fatal error message is indicated for those particular BDYLIST identification numbers (IDF_j s).

EXECUTIVE PREFACE MODULE IFP4 (INPUT FILE PROCESSOR - PHASE 4)

6. At this time a normal GRID card image is created from each GRIDB image and merged along with existing GRID card images on GEØM1. Additional GRID card images will be added to GEØM1 in subsequent manipulations.

For each GRIDB card image now residing in core (note 4.89.7.2-2) a normal GRID card is created and consists of the following eight values.

<u>Field</u>	<u>Symbol</u>	<u>Description</u>
1	ID_g	ID given on GRIDB image.
2	CS_ℓ	CS_f from the AXIF image.
3-5	x_1, x_2, x_3	<p>These values are formulated by finding $X1, X2, X3$ in the boundary list table entry whose primary identification number matches the reference identification number (IDF) given on the GRIDB card image, and then:</p> <p>$x_1 = X1$</p> <p>$x_2 = \phi$ if CS_f is cylindrical, or $X2$ if CS_f is spherical.</p> <p>$x_3 = X3$ if CS_f is cylindrical, or ϕ if CS_f is spherical.</p> <p>Where ϕ is supplied by the GRIDB card image.</p>
6	CS_d	CD from the GRIDB image.
7	PS	PS from the GRIDB image.
8	0	Not used.

The resulting GRID data card images are merged with the existing GRID cards on data blocks GEØM1. If no harmonics exist for the fluid, the module processing is complete.

7. To generate the nonsymmetric connection tables for the boundary, the boundary list table is further altered at this time to result in a table listing the geometry and related grid points for each boundary fluid point.
 - a. For each fluid point entry now in the boundary list table the values $X1, X2$, and $X3$ are converted to r and z values which are stored respectively in place of $X1$ and $X2$. If CS_f is cylindrical then:

MODULE FUNCTIONAL DESCRIPTIONS

$$\left. \begin{aligned} r &= X1 \\ z &= X3 \end{aligned} \right\} \quad (4)$$

X2 must be zero.

If CS_f is spherical then:

$$\left. \begin{aligned} r &= X1 \sin\left(\frac{\pi}{180} X2\right) \\ z &= X1 \cos\left(\frac{\pi}{180} X2\right) \end{aligned} \right\} \quad (5)$$

X3 must be zero.

- b. For each set of three fluid point identification numbers (position 1, 5, and 6 of each entry), the three pairs of coordinates are extracted. The primary values r_j and z_j are given with each entry. The secondary values r_{j-1} , z_{j-1} , r_{j+1} , z_{j+1} must be found by finding the entries which have the same primary identification number as the secondary identification numbers (IDF_{j-1} or IDF_{j+1}) in question. If "axis" (-1) is a secondary identification number, then:

$$\left. \begin{aligned} r_{axis} &= 0 \\ z_{axis} &= z_j \end{aligned} \right\} \quad (6)$$

If "not available" (0) is a secondary identification number, then:

$$\left. \begin{aligned} r &= r_j \\ z &= z_j \end{aligned} \right\} \quad (7)$$

The values l , c , and s are calculated and replace the 4th, 5th and 6th word of each entry in the boundary point list at this time such that each entry will now contain:

EXECUTIVE PREFACE MODULE IFP4 (INPUT FILE PROCESSOR - PHASE 4)

<u>Field</u>	<u>Symbol</u>	<u>Description</u>
1	IDF _j	Fluid point identification
2	r _j	Radial location
3	z _j	Vertical location
4	l	Length and associated angle components of a conical section
5	c	
6	s	
7	ρ _b	Fluid density

where:

$$l = \sqrt{\Delta r^2 + \Delta z^2} \quad (8)$$

$$c = \frac{\Delta z}{l} \quad (9)$$

$$s = \frac{\Delta r}{l} \quad (10)$$

and:
$$\Delta r = \frac{1}{2} \left\{ r_{j+1} - r_{j-1} + \frac{1}{4r_j} [(r_{j+1} - r_j)^2 - (r_{j-1} - r_j)^2] \right\} \quad (11)$$

$$\Delta z = \frac{1}{2} \left\{ z_{j-1} - z_{j+1} + \frac{1}{4r_j} [(r_{j+1} - r_j)(z_j - z_{j+1}) - (r_j - r_{j-1})(z_{j-1} - z_j)] \right\} \quad (12)$$

This list, now referred to as the "boundary point geometry table", remains in core.

The values l, s, and c correspond to the cross section length, and the sine and cosine of the angle ψ as given in Equation 14, Section 16.1.5 of the Theoretical Manual.

- c. As any number of GRIDB points may be connected to a fluid point, the GRIDB card images are now sorted on the referenced fluid point identification (the fifth word of each GRIDB entry). For each set of GRIDB points with the same referenced fluid point the sort is further made on the angle (φ).
- d. The boundary point geometry table, generated above, is used at this time to form the boundary matrix part of the MATPØØL data block. For each entry in the table, all GRIDB points which reference it are appended to form a new entry of the following form.

MODULE FUNCTIONAL DESCRIPTIONS

<u>Field</u>	<u>Symbol</u>	<u>Description</u>
1	Id_f	Fluid point identification
2-7	r, z, ℓ, c, s, ρ_b	Fluid point properties
8	$Id_g(1)$	GRIDB identification
9	ϕ_1	Angular position of GRIDB point
.	.	.
.	.	.
.	.	.
6+2M	$Id_g(M)$	GRIDB identification
7+2M	ϕ_M	Angular position of GRIDB point
8+2M	-1	End of entry flags
9+2M	-1	

where M equals the number of GRIDB points that reference Id_f . Each new boundary fluid point entry is then placed in the MATPØØL data block. The list of harmonic indices n_j , the gravity G, the NØSYM flag, the fluid coordinate system CS_f , and the symmetric boundary information (from the FLSYM card image) are placed in the first record of the boundary list data in the MATPØØL data block. If DMIG data card images resulting from DMIAX data cards are present on the MATPØØL data block, they are merged to the existing DMIG card image data. Matrix names are checked for uniqueness.

- The RINGFL data cards define a ring (fluid point) with its axis coincidental with the axis of the fluid coordinate system. Its degrees of freedom are the harmonics of the pressure around the circle. Special GRID point card images corresponding to the RINGFL data cards are generated at this time and added to the GRID card images now on GEØM1. Each RINGFL card image is read and N GRID card images are created containing the following data.

EXECUTIVE PREFACE MODULE IFP4 (INPUT FILE PROCESSOR - PHASE 4)

<u>Field</u>	<u>Value</u>	<u>Description</u>
1	$Id_f + 5 \cdot 10^5 \cdot (I_i)$	Point identification
2	CS_f	Fluid coordinate system number
3-5	$X1, X2, X3$	Location coordinates
6	-1	Fluid point flag
7	0	Permanent single point constraints
8	0	Not used

where i goes from 1 to N for each RINGFL card image read.

SEQGP card images are created for each RINGFL card image and merged with SEQGP cards on GEOM1. The contents of the entry are:

<u>Field</u>	<u>Value</u>	<u>Description</u>
1	$Id_f + 5 \cdot 10^5 \cdot (I_i)$	Grid identification
2	$Id_f \cdot 10^3 + (I_i - 1)$	Sequence number

where i goes from 1 to N for each RINGFL data card image.

4.89.7.2 GEOM2 Data Block Processing

1. The fluid element connections as specified by the CFLUID2, CFLUID3, and CFLUID4 card images are now operated upon. Each input card image is used together with the harmonic indices to define N "structural elements". The data given by the input card image is:

<u>Field</u>	<u>Value</u>	<u>Description</u>
1	Id_e	Element identification number
2 thru $j+1$	Id_j	Where $j = 2, 3, \text{ or } 4$ fluid point connections
$j+2$	ρ	Fluid density
$j+3$	B	Fluid bulk modulus

For each input card image, connection card images are created for all harmonics in the problem. Their format is:

MODULE FUNCTIONAL DESCRIPTIONS

<u>Field</u>	<u>Value</u>	<u>Description</u>
1	$Id_e \cdot 10^3 + I_i$	Converted element identification
2 thru j+1	$Id_j + 5 \cdot 10^5 (I_i)$	Where j = 2, 3, or 4 connected fluid points
j+2	ρ	Fluid density
j+3	B	Fluid bulk modulus
j+4	n	Harmonic number

where $i = 1, 2, \dots, N$ and n is an integer such that,

$$\frac{I_1 - 3}{2} < n \leq \frac{I_1 - 1}{2} \quad (13)$$

The harmonic element connection card images are merged into the GEOM2 data block as they are generated.

2. FSLIST card images each define a sequential list of fluid (RINGFL) points on a free surface. The FREEPT card images input to IFP4 each define a point on the free surface where a displacement may be output. The following operations are a result of FSLIST and FREEPT card images data.

For each fluid point (IDF_j) defined in a FSLIST card image, a three word entry is placed in core containing IDF_j , IDF_{j+1} , and ρ . The subscript j indicates the respective IDF in the FSLIST card image list of IDFs. if IDF_j or IDF_{j+1} is represented by "AXIS" in the FSLIST card image, a minus one (-1) is used. If IDF_j is the last point in the list, IDF_{j+1} is set to -2. If the fluid density (ρ) is not present (an integer -1) in the FSLIST card image, the default fluid density (ρ_d) from the AXIF image is used. If both ρ and ρ_d are missing a user fatal error results.

A set of structural mass elements are generated for each of the entries just added to core. Each set represents all harmonics in the problem. Connection card images called CMFREE elements are created such that each element consists of the following:

EXECUTIVE PREFACE MODULE IFP4 (INPUT FILE PROCESSOR - PHASE 4)

<u>Field</u>	<u>Symbol</u>	<u>Description</u>
1	Id	Element Id = $10^6 k + I_i$
2	Idg ₁	$IDF_{k,1} + 5 \cdot 10^5 I_i$
3	Idg ₂	$IDF_{k,2} + 5 \cdot 10^5 I_i$
4	γ_j	(ρ times G) the weight density, where G = gravity from AXIF image
5	n	Integer harmonic number such that; $\frac{I_i - 3}{2} < n \leq \frac{I_i - 1}{2}$

where I_i represents the i^{th} entry in the harmonic index list, and k is the index of the entry in the FSLIST table of entries. If $IDF_{k,1} = -1$, $IDF_{k,1}$ is set to $IDF_{k,2}$. If $IDF_{k,2} = -1$, then $IDF_{k,2}$ is set to $IDF_{k,1}$. Both can not be -1 initially. Thus for each entry, $k = 1$ thru K (the total number of entries), CMFREE images are created for all harmonic indices (I_i), $i = 1$ thru N . These CMFREE element entries are merged into the GEOM2 data block as were the CFLUID2, CFLUID3, and CFLUID4 card images.

4.89.7.4 GEOM4 Data Block Processing

1. If FREEPT (free surface displacement point) card images are present, and gravity as specified in the AXIF card image is nonzero, a multipoint constraint (MPC) is generated at this time along with a scalar point (SPØINT) having the same identification number (Id_p) as specified by each FREEPT card image. As each FREEPT card is read an SPØINT card image is placed in core and the following MPC card image is merged into the MPC data of GEOM4:

MODULE FUNCTIONAL DESCRIPTIONS

	<u>Field</u>	<u>Symbol</u>	<u>Value</u>	<u>Description</u>
	1	SID	102	Set identification number
	2	GID	Id_p	FREET identification number
	3	Comp	0	Component
	4	A_1	$- \rho G $	Density times gravity
Repeats for $i=1$ to N	$2+3i$	GID_i	$Id_f + 5 \cdot 10^5 (I_i)$	Fluid point harmonic identification
	$3+3i$	Comp	0	Component
	$4+3i$	A_{i+1}	C_i	Harmonic coefficient
	\vdots	\vdots	\vdots	\vdots
	$5+3N$	Flag	-1	End of image flag
	$6+3N$	Flag	-1	
	$7+3N$	Flag	-1	

I_i is a harmonic index, and n equals an integer such that;

$$\frac{I_i - 3}{2} < n \leq \frac{I_i - 1}{2} \quad , \quad (14)$$

$$\begin{aligned} \text{then: } C_i &= \cosine \frac{n\pi\phi}{180} & \text{if } I_i \text{ is even,} \\ C_i &= \sin \frac{n\pi\phi}{180} & \text{if } I_i \text{ is odd,} \end{aligned} \quad \left. \vphantom{\begin{aligned} \text{then: } C_i &= \cosine \frac{n\pi\phi}{180} \\ C_i &= \sin \frac{n\pi\phi}{180} \end{aligned}} \right\} \quad (15)$$

where: ϕ is the angle given in the FREET card image.

- Additional MPC card images are created if PRESPT card images are present. For each PRESPT card image read, an SPØINT is added to the in-core list of SPØINTs, and the following MPC card image is merged onto GEØM4.

EXECUTIVE PREFACE MODULE IFP4 (INPUT FILE PROCESSOR - PHASE 4)

<u>Field</u>	<u>Symbol</u>	<u>Value</u>	<u>Description</u>
1	SID	102	Set identification
2	GID	Id_p	PRESPT identification
3	Comp	0	Component
4	A_1	-1.0	Coefficient
Repeat for $i=1$ to N	$2+3i$	GID_i	$Id_f + 5 \cdot 10^5 (I_i)$ Connected fluid harmonic identification
	$3+3i$	Comp	0 Component
	$4+3i$	A_{i+1}	C_i Harmonic coefficient
	\vdots	\vdots	\vdots
	\vdots	\vdots	\vdots
$5+3N$	Flag	-1	
$6+3N$	Flag	-1	End of image flag
$7+3N$	Flag	-1	

I_i is a harmonic index, and n equals an integer such that,

$$\frac{I_i - 3}{2} < n \leq \frac{I_i - 1}{2} \quad (16)$$

$$\left. \begin{aligned} C_i &= \cosine \frac{n\pi\phi}{180} && \text{if } I_i \text{ is even.} \\ C_i &= \sin \frac{n\pi\phi}{180} && \text{if } I_i \text{ is odd.} \end{aligned} \right\} \quad (17)$$

ϕ is the angle given in the PRESPT card image.

3. If any SPØINTs were placed in core as a result of the presence of FREEPT or PRESPT card data, they are merged with the existing scalar point data on a scratch file.
4. At this time, if any harmonics are specified, an MPCADD card image is generated for each unique set identification present in the MPC and MPCADD card images on GEØM4. This MPCADD card image will then contain the internal set identification and include the user set identification. Thus as the MPC and MPCADD card images are read from GEØM4, a list of the set identifications present is created in core. An MPCADD card is then generated for

MODULE FUNCTIONAL DESCRIPTIONS

each unique set identification (Id) present. Its format is:

<u>Field</u>	<u>Value</u>	<u>Description</u>
1	$2 \cdot 10^8 + Id$	Internal set identification
2	Id	User set identification
3*	102	Generated MPC set identification
3 or 4*	-1	End of image flag

If any MPCADD card images are present on GEOM4, as a result of the user's specifications, their set identification (Id) in field one is modified to an internal set identification ($2 \cdot 10^8 + Id$), and if any MPC card images have been internally generated for set 102, the 102 set identification is added to the list of included set identifications therein.

If any MPC card images have been created for set 102, the following MPCADD card is generated in any event so as to assure that the 102 set be included in the solution. The set identification used here ($2 \cdot 10^8$), will be referenced in later computations if the user has not referenced any MPC constraint set.

<u>Field</u>	<u>Value</u>	<u>Description</u>
1	$2 \cdot 10^8$	Set identification
2	102	Generated MPC set to be included
3	-1	End of image flag

5. Should the user specify gravity G to be zero (0) on the AXIF card, it is assumed that the effects of gravity on the free surface are to be removed. To accomplish this, a single point constraint (SPC set 102) set is created at this time by IFP4.

For each fluid point Id_f not equal to minus one (-1) or minus two (-2) in the free surface list, an SPC1 card image is merged onto GEOM4 containing the constraint information for all harmonics of this point. Its format is the following:

*Set identification 102 is inserted only if any MPC card images have been generated for set 102. If $Id = 102$ a user fatal error is indicated.

EXECUTIVE PREFACE MODULE IFP4 (INPUT FILE PROCESSOR - PHASE 4)

<u>Field</u>	<u>Value</u>	<u>Description</u>
1	102	Set identification
2	0	Component to be constrained
2+i	$Id_f + 5 \cdot 10^5 I_i$	Point to be constrained
:	:	:
:	:	:
:	:	:
2+i	$Id_f + 5 \cdot 10^5 I_i$	Point to be constrained
:	:	:
:	:	:
:	:	:
2+N	$Id_f + 5 \cdot 10^5 I_N$	Point to be constrained
3+N	-1	End of image flag

I_i is the i^{th} entry in the list of harmonic indices.

6. Analogous operations to those described in paragraph (4) of this section are performed at this time for existing SPC, SPC1 and SPCADD card images. The data is merged onto a scratch file and when complete, the scratch file is merged onto the GEOM4 data block.

4.89.8 Subroutines

4.89.8.1 Subroutine Name: IFP4A

1. Entry Point: IFP4A
2. Purpose: To write the first line of the user fatal error messages.
3. Calling Sequence: CALL IFP4A(NUM)

NUM - Message number minus 4030.

4.89.8.2 Subroutine Name: IFP4B

1. Entry Point: IFP4B
2. Purpose : To copy data from IFP data files up to and including a given record onto a scratch file. On option the data on the scratch file may be copied onto the original data block.

MODULE FUNCTIONAL DESCRIPTIONS

3. Calling Sequence:

CALL IFP4B(FILE,SCRT,ANY,SPACE,LSPACE,RECID,EØF)

where:

FILE - File Number
SCRT - Scratch File Number
ANY - Flag = .TRUE. if RECID is found on FILE,
= .FALSE. if record is missing
SPACE - Area of core for working space
LSPACE - Length of working space
RECID - Record Number of FILE where the copy process stops. If -1 is
used the copy process proceeds to the end of FILE and the data
on SCRT is copied back onto FILE.
EØF - Flag = .TRUE. if end of record is encountered on return.

4.89.8.3 Subroutine Name: IFP4C

1. Entry Point: IFP4C

2. Purpose: To open an IFP generated file and a scratch file and to copy the header record from the IFP file onto the scratch file.

3. Calling Sequence:

CALL IFP4C(FILE,SCRT,BUF1,BUF2,EØF)

where:

FILE - File number
SCRT - Scratch file number
BUF1 - Buffer area in core for reading FILE
BUF2 - Buffer area in core for writing SCRT
EØF - Flag = .TRUE. if FILE is null

4.89.8.4 Subroutine Name: IFP4E

1. Entry Point: IFP4E

2. Purpose: To check identification numbers of fluid points for possible difficulties with large numbers.

EXECUTIVE PREFACE MODULE IFP4 (INPUT FILE PROCESSOR - PHASE 4)

3. Calling Sequence: CALL IFP4E(ID)

where:

ID - Identification number

4.89.8.5 Subroutine Name: IFP4F

1. Entry Point: IFP4F

2. Purpose: To test if a bit in a trailer record word is on or off.

3. Calling Sequence: CALL IFP4F(IBIT,FILE,BIT)

where:

IBIT - Position of bit in trailer

FILE - File number

BIT - Flag = .TRUE. if bit is on
= .FALSE. if bit is off

4.89.8.6 Subroutine Name: IFP4G

1. Entry Point: IFP4G

2. Purpose: To turn on a bit in a trailer record.

3. Calling Sequence: CALL IFP4G(IBIT,FILE)

where:

IBIT - Position of bit in trailer

FILE - File number

FUNCTIONAL MODULE BMG (BOUNDARY MATRIX GENERATOR FOR HYDROELASTIC PROBLEMS)

4.90 FUNCTIONAL MODULE BMG (BOUNDARY MATRIX GENERATOR FOR HYDROELASTIC PROBLEMS)

4.90.1 Entry Point: BMG

4.90.2 Purpose

The MATP00L data block may contain data related to fluid boundaries which is generated by the IFP4 preface module. The purpose of this module is to combine these data with the geometry data (EQEXIN, BGPDT, and CSTM data blocks) to produce matrix terms which describe fluid-structure connection forces. These matrix terms are produced in the form of internal DMIG data card images. The module MTRXIN must always be used in conjunction with module BMG to produce NASTRAN matrices.

4.90.3 DMAP Calling Sequence

BMG MATP00L,BGPDT,EQEXIN,CSTM / BDP00L / V,N,N0KBFL / V,N,N0ABFL / V,N,MFACT \$

4.90.4 Input Data Blocks

MATP00L - Direct Input Matrices and Hydroelastic Boundary data.

BGPDT - Basic Grid Point Definition Table.

EQEXIN - Equivalence between External and Internal Grid Point numbers.

CSTM - Coordinate System Transformation Matrices.

4.90.5 Output Data Blocks

BDP00L - Boundary Matrices ABFL and KBFL in DMIG Format.

4.90.6 Parameters

N0KBFL - Existence of KBFL Matrix Data = 0,
No KBFL Data = -1, output parameter.

N0ABFL - Existence of ABFL Matrix Data = 0,
No ABFL Data = -1, output parameter.

MFACT - Complex Factor for Symmetric Structures, output parameter.

4.90.7 Method

The fluid boundary data, contained in the MATP00L data block, is grouped by the fluid points on the boundary. For each fluid point the geometric parameters of the surface and the positions of the associated grid points are listed. The input data read for each fluid point are operated

MODULE FUNCTIONAL DESCRIPTIONS

on to produce matrix terms. The output matrix data are written on two files. The ABFL matrix terms are written on the BDP00L file. The KBFL data are written on a scratch file. After the processing of the input file is complete, the KBFL data is appended to the BDP00L file.

During the processing the core is allocated for the geometry data blocks. For each fluid point, tables are also created which must fit in the remaining core. The following description lists the form of the input data and the various steps used in the process.

4.90.7.1 Form of the Boundary Data Record on the MATP00L Data Block

Three levels of data are contained within this record. The first level is a list of the overall definition parameters of the fluid and the boundary. The second level consists of fluid points and their associated data. Attached to each fluid point is a third level consisting of a list of the connected structural grid points and their angular position on the fluid circle. The actual data in the record are:

1. Header data:

<u>Symbol</u>	<u>Description</u>
CS_f	Fluid coordinate system identification
M	Number of symmetric sections
S1 S2	Symmetry definitions of first and second boundary
g	Value of gravity
N0SYM	Flag for n* series
k	Number of harmonic indices below
$n_1, n_2, \dots, n_j, \dots, n_k$	List of harmonic indices

FUNCTIONAL MODULE BNG (BOUNDARY MATRIX GENERATOR FOR HYDROELASTIC PROBLEMS)

2. Fluid point data

<u>Symbol</u>	<u>Description</u>
Id_{f1}	First fluid point (RINGFL) identification
r, z, l, C, S, ρ	Fluid point properties
N_g	Number of connected grid points below
Id_1	Grid point identification
ϕ_1	Angular position of First grid point
Id_2	\vdots
ϕ_2	\vdots
\vdots	\vdots
Id_i	\vdots
ϕ_i	(grid point data)
\vdots	\vdots
Id_g	\vdots
ϕ_g	\vdots
Id_{f2}	Second fluid point
\vdots	\vdots
etc.	etc.

4.90.7.2 Selection of Harmonics to Match Boundary Conditions

The Header Record for the boundary data is read and a list of harmonics (n_j and n_j^*) to be included in the matrices are precalculated. If NØSYM = YES, the indices for the sine series will be included. A test is made on each value of n and n^* using the values of S1 and S2 in the header data.

1. If $M = 0$ or 1 accept all values of n and n^* .
2. If $S1 = S2$, Calculate:

$$K = \frac{2n}{M} \quad (1)$$

- a. If K is not an integer reject n or n^*
- b. If K is an integer:
 - accept n if $S1 = S$
 - accept n^* if $S1 = A$

MODULE FUNCTIONAL DESCRIPTIONS

3. If $S1 \neq S2$, calculate:

$$K = \frac{1}{2} \left[\frac{4n}{M} - 1 \right] \quad (2)$$

a. If K is not an integer reject n or n^*

b. If K is an integer:

accept n if $S1 = S$

accept n^* if $S1 = A$

A list of allowable values of n and n^* is built in core. If the final list is null, only the KBFL matrix is generated. The parameter MFACT is the complex number $(M,0)$ if M is nonzero. The value $(1,0)$ is used if M is zero.

4.90.7.3 Formation of Geometry Table for Internal Use

The core is allocated for the BGPDT data block and an extra word for each of its entries. The data is read in groups of four words and stored in five word entries, reserving the first word for the external identification number. The EQEXIN data contains a paired list of external and internal numbers. The EQEXIN is read and the external numbers are placed in the corresponding BGPDT entries in core. The resulting BGPDT data are sorted on the external identification numbers. The referenced coordinate system number and the basic location vector for any grid point are now available by using a single binary search.

4.90.7.4 CSTM Processing

The CSTM data block is now read and stored in core. The fluid coordinate system identification number, CS_f , is found and the 3 by 3 transformation matrix, $[T_{of}]$, is extracted directly from the data.

4.90.7.5 File Initialization

The processing of the matrix data may now begin. The files for the ABFL and KBFL output data are opened and the matrix header data is written. The boundary data on the MATP00L data block are read and one fluid point at a time is processed.

4.90.7.6 Calculations of Areas Associated with Boundary Grid Points

The first set of the parameters Id , r , z , ℓ_k , c_k , S_k , and ρ_k are read for the fluid point where $k = 1$ if the fluid point has only one entry or $k = 1, 2, 3 \dots$ if the fluid point is connected with multiple boundaries. The connected grid point numbers (Id_i) and angles (ϕ_i) are read and placed in core. Twenty-six words are allocated for each grid point.

For each connected grid point the calculated data are:

(1)	Id_i	Identification number
(2)	ϕ_i	Azimuthal angle (radians)
(3)	ϕ_{0i}	Angle midway to previous point
(4)	ϕ_{1i}	Angle midway to next point
(5-22)	$[V_i]$	3x3 double precision transformation
(23-26)	$\{W_i\}$	3x1 vector

The midway angles are defined in general as:

$$\begin{aligned}\phi_{0i} &= \frac{\phi_i + \phi_{i-1}}{2} , \\ \phi_{1i} &= \frac{\phi_i + \phi_{i+1}}{2} .\end{aligned}\tag{3}$$

The angles for the first point are:

$$\begin{aligned}\phi_{01} &= \phi_1 , \quad M \geq 2 , \\ \phi_{01} &= \frac{\phi_1 + \phi_N - 2\pi}{2} , \quad M = 0.\end{aligned}\tag{4}$$

The angles for the last point are:

$$\begin{aligned}\phi_{1N} &= \phi_N , \quad M \geq 2 , \\ \phi_{1N} &= \frac{\phi_N + \phi_1 + 2\pi}{2} , \quad M = 0 .\end{aligned}\tag{5}$$

All of the grid point data are sorted on the grid point numbers before the transformations $[V_i]$ and $\{W_i\}$ are calculated.

MODULE FUNCTIONAL DESCRIPTIONS

The equations for the transformation matrices are:

$$\begin{aligned} [V_i] &= [T_i]^T [T_{of}] \quad , \\ \{W_i\} &= [T_i]^T [T_{of}] \begin{Bmatrix} 0 \\ 0 \\ 1 \end{Bmatrix} \quad , \end{aligned} \tag{6}$$

Where $[T_i]$ is the 3 by 3 global-to-basic transformation matrix for grid point i .

4.90.7.7 Calculation of Matrix Terms

The matrix terms corresponding to one fluid point are generated for the ABFL and KBFL data tables at this stage. The ABFL matrix terms are generated as follows:

1. For each allowable value of n or n^* a matrix column is generated in the ABFL table. The internal numbers, G_j , for the fluid point identification number, Id_f , are:

$$G_j = Id_f + (n + 1)10^6 \quad \text{cosine series}$$

$$G_j = Id_f + \frac{2n^* + 1}{2} 10^6 \quad \text{sine series}$$

These numbers label the column of the matrix.

2. Each row position in the matrix is labeled by a grid point, Id_i , and its three components: $C = 1, 2, 3$. The values corresponding to these positions are the values of the vector $\{A_{in}\}$ where:

$$\{A_{in}\} = \sum_k A_{ik}^n [V_i] \begin{Bmatrix} C_k \cos \phi_i \\ C_k \sin \phi_i \\ S \end{Bmatrix} \quad . \tag{7}$$

The vector $\{A_{in}^*\}$ is similar. The coefficients are:

$$\begin{aligned} A_{ik}^0 &= r_{lk} (\phi_{1i} - \phi_{0i}) \quad , \quad n = 0 \\ A_{ik}^n &= \frac{r_{lk}}{n} [\sin(n\phi_{1i}) - \sin(n\phi_{0i})] \quad , \quad n > 0 \\ A_{ik}^{n^*} &= \frac{r_{lk}}{n} [\cos(n^*\phi_{1i}) - \cos(n^*\phi_{0i})] \quad , \quad n^* > 0 \end{aligned} \tag{8}$$

FUNCTIONAL MODULE BMG (BOUNDARY MATRIX GENERATOR FOR HYDROELASTIC PROBLEMS)

where $k = 1, 2, 3, \dots$ is an index if the fluid point occurs more than once in the boundary list tables.

3. If gravity, g , is nonzero the KBFL matrix terms are calculated. Each grid point, Id_i , connected to the fluid point is used to produce three columns of the matrix corresponding to the three components, $C_j = 1, 2, 3$. The three rows for each column are the same three components, $C_i = 1, 2, 3$, of the grid point. The equation for the three terms in each column is:

$$\{K_i\}_j = \sum_k B_{ii}^k w_{ij} [V_i] \begin{Bmatrix} C_k \cos \phi_i \\ C_k \sin \phi_i \\ S_k \end{Bmatrix} \quad (9)$$

where:

$$w_{ij} \text{ is the } j^{\text{th}} \text{ component of } \{w_i\} \text{ and:} \quad (10)$$

$$B_{ii}^k = r_{Lk} \rho_k (\phi_{i1} - \phi_{i0}) g .$$

k is the index used if the points are included in more than one boundary list entry.

4.90.7.8 Wrapup Operation

The final operations involve rewinding the scratch file containing the KBFL data and appending that data to the BDP00L data block output file.

4.90.8 Additional Subroutines

BMGTNS - This routine is a slightly modified version of utility subroutine PRETRD so as to have only one entry point.

4.90.9 Design Requirements

The major core requirements are that the BGPDT data must fit and that twenty-six words for each boundary grid point converted to one fluid point must fit in core.

MODULE FUNCTIONAL DESCRIPTIONS

4.90.10 Diagnostic Messages

The following system fatal messages may be issued by BMG:

***SYSTEM FATAL MESSAGE 2148, COORDINATE SYSTEM = XXXXX CAN NOT BE FOUND IN CSTM DATA.

***SYSTEM FATAL MESSAGE 2149, CONNECTED FLUID POINT ID = XXXXX IS MISSING BGPDT DATA.

EXECUTIVE PREFACE MODULE IFP5 (INPUT FILE PROCESSOR - PHASE 5)

4.91 EXECUTIVE PREFACE MODULE IFP5 (INPUT FILE PROCESSOR - PHASE 5)

4.91.1 Entry Point: IFP5

4.91.2 Purpose

1. To convert the data card images related to acoustic analysis into conventional grid points and elements.
2. To calculate slot-cavity interface matrix terms and generate corresponding scalar elements.
3. To generate plot elements describing the sides of the acoustic elements.

4.91.3 Calling Sequence

CALL IFP5. IFP5, an Executive Preface Module, is called only by the Preface driver SEMINT.

4.91.4 Input Data Blocks

AXIC - Contains Bulk Data Cards related to the acoustic parameter, points, and boundaries.

GEØM1- Grid point and coordinate system data

GEØM2- Element Data, including acoustic elements.

4.91.5 Output Data Blocks

GEØM1 - Same format as input, acoustic points are merged in as GRID points.

GEØM2 - Same format as input, scalar elements and plot elements are added.

4.91.6 Parameters

Not applicable to IFP5

4.91.7 Method

IFP5 converts the data on the AXIC data block into conventional grid points and elements. The GEØM1 and GEØM2 data are read and merged with the new data on scratch files. The complete data sets are copied back onto the GEØM1 and GEØM2 files. (This is not normally allowed in a NASTRAN Module. The preface modules, however, have the privilege of writing on an input file.)

The data cards listed below are processed by IFP5. The corresponding output card image and its data block are given for each card.

MODULE FUNCTIONAL DESCRIPTIONS

<u>IFP5 Input Card Image</u>	<u>Data Block In</u>	<u>IFP5 Output Card Image</u>	<u>Data Block Out</u>
AXSLØT	AXIC	(all below)	(all below)
CAXIF2 } CAXIF3 } CAXIF4 } CSLØT3 } CSLØT4 }	GEØM2	{ CAXIF2 { CAXIF3 { CAXIF4 { CSLØT3 { CSLØT4 { PLØTEL	GEØM2
GRIDF } GRIDS }	AXIC	GRID	GEØM1
BDYLIST	AXIC	CELAS2	GEØM2

It should be noted that the formats of the CAXIFi data cards are exactly the same as the CFLUIDi data cards as generated by IFP4, Section 4.89. The following steps are followed to process the data:

1. The AXSLØT card is read from the AXIC file, its data are:

ρ_d - default density
 B_d - default bulk modulus
 N - harmonic number
 w_d - default slot width
 M - Number of slots

2. The GRIDS data card images are read and stored in core. The contents of each card are:

Id_s - identification number
 r - radius
 z - axial coordinate
 w - slot width
 Id_f - identification of assoicated GRIDF

3. The GRIDF data card images are read and stored in core. The contents of each card are

Id_f - identification number
 r - radius
 z - axial coordinate

EXECUTIVE PREFACE MODULE IFP5 (INPUT FILE PROCESSOR - PHASE 5)

4. If the value in field 5 of GRIDS card is nonzero an IDF card is generated with the values Id_f , r , z as given on the GRIDS card. These cards are added to the GRIDF images and the complete list of GRIDF cards is sorted.
5. Data block GEØM1 is copied onto the first scratch file up to the first GRID card. The GRIDF and GRIDS cards are merged with the GRID cards in the GRID card format as follows:

<u>GRID Field</u>	<u>Value GRIDF</u>	<u>Value GRIDS</u>
1	Id_f	Id_s
2	0	0
3	r	r
4	z	z
5	0	w
6	-1	-1
7	0	0

6. The remainder of GEØM1 is copied onto the scratch file. The scratch file is then copied back onto GEØM1, starting from the beginning.
7. The SLBDY data card images are read from the AXIC data block. For each entry, Id_i on a logical card, five words are allocated in core and the following is stored.

$$Id_i, Id_{i-1}, Id_{i+1}, RHØ, M$$

where Id_i is a point number in the list

Id_{i-1} is the previous point number in the list

Id_{i+1} is the next point number in the list

$RHØ, M$ are given on the logical card

If Id_i is the first entry on a logical card, $Id_{i-1} = -1$. If Id_i is the last entry on a logical card, $Id_{i+1} = -1$.

8. After all SLBDY cards are processed the above list is sorted on the first entry in each group of 5.
9. Plot elements (PLØTEL) are generated and placed on the first scratch file. The GEØM2 data block is read and for each (AXIFI) data card a series of PLØTEL cards are generated and written on the first scratch file (SCRT1).

MODULE FUNCTIONAL DESCRIPTIONS

CAXIF2 Data

Id
G1
G2
 ρ
B
N

PLØTEL Data

Id + 10^6
G1
G2

CAXIF3 Data

Id
G1
G2
G3
 ρ
B
N

PLØTEL Data

Id + $2 \cdot 10^6$
G1
G2

Id + $3 \cdot 10^6$
G2
G3

Id + $4 \cdot 10^6$
G3
G1

CAXIF4 Data

Id
G1
G2
G3
G4
 ρ
B
N

PLØTEL Data

Id + $5 \cdot 10^6$	Id + $6 \cdot 10^6$
G1	G2
G2	G3
Id + $7 \cdot 10^6$	Id + $8 \cdot 10^6$
G3	G4
G4	G1

10. A second scratch file (SCRT2) is opened and the GEØM2 data is copied onto SCRT2 to the CELAS2 data card position. The boundary list data is processed and CELAS2 data cards are generated and appended to SCRT2. For each five word entry in the Boundary Table search the GRIDS data card images for the following data

r_i, z_i, w_i, IDF from GRIDS card "IDS_i"

$r_{i-1}, z_{i-1}, w_{i-1}$ from GRIDS card "IDS_{i-1}"

$r_{i+1}, z_{i+1}, w_{i+1}$ from GRIDS card "IDS_{i+1}"

where $r = r_i, z = z_i$ if the corresponding identification number IDS is -1. If a GRIDS card can not be found a fatal error exists. The following data is calculated for each entry:

EXECUTIVE PREFACE MODULE IFP5 (INPUT FILE PROCESSOR - PHASE 5)

$$l_1 = \sqrt{(z_{i+1} - z_i)^2 + (r_{i+1} - r_i)^2} ,$$

$$l_2 = \sqrt{(z_{i-1} - z_i)^2 + (r_{i-1} - r_i)^2} ,$$

$$\bar{l} = \frac{1}{2} \sqrt{(z_{i+1} - z_{i-1})^2 + (r_{i+1} - r_{i-1})^2} ,$$

$$\bar{w} = \frac{1}{4(l_1 + l_2)} [l_1 w_{i+1} + l_2 w_{i-1}] + \frac{3}{4} w_i ,$$

$$\bar{r} = \frac{1}{4(l_1 + l_2)} [l_1 r_{i+1} + l_2 r_{i-1}] + \frac{3}{4} r_i .$$

The coefficients for slot interaction are:

$$\beta = \frac{2\pi\bar{r}}{M\bar{w}}$$

(If $\beta \geq 1$ a fatal error exists)

$$l_c = \frac{\bar{w}}{2\pi} \left[\left(\beta + \frac{1}{\beta} \right) \log_e \left(\frac{\beta+1}{\beta-1} \right) + 2 \log_e \frac{(\beta+1)(\beta-1)}{\beta r} \right]$$

and

$$l_e = \text{Max} (l_c, .01\bar{w})$$

$$K_f = \frac{\bar{w} \bar{l}}{\rho l_e} F_i$$

where $F_i = M$ if $N = 0$ or $N = \frac{M}{2}$, $G_i = \frac{M}{2}$ otherwise.

11. For each entry in the Boundary Table, corresponding GRIDF data card with ID = IDF is found in core. For the corresponding GRIDF point IDF_j calculate the following:

$$\alpha = \frac{\sin \frac{N\bar{w}}{2\bar{r}}}{\frac{N\bar{w}}{2\bar{r}}}$$

MODULE FUNCTIONAL DESCRIPTIONS

12. CELAS2 elements are now generated for the slot point IDS_i and the corresponding axisymmetric fluid point IDF_j . The format of this data is:

Id	K	G1	C1	G2	C2
$Id_e + 1$	$(1-\alpha)K_f$	IDS_i	"1"	blank	blank
$Id_e + 2$	αK_f	IDS_i	"1"	IDF_i	"1"
$Id_e + 3$	$\alpha(1-\alpha)K_f$	IDF_i	"1"	blank	blank

The element identification numbers Id_e are sequential starting with 10,000,001. With each new point on the boundary list, IDF_i , the value Id_e is incremented by 3.

13. When all points on the boundary list are processed, the remainder of GEOM2 is copied onto SCRT2. If CSLØT3 and/or CSLØT4 elements are encountered, they will produce PLØTEL data card images which are written on SCRT1 in the following format:

CSLØT3 Data

Id
G1
G2
G3
p
B
M
N

PLØTEL Data

$Id + 9 \cdot 10^6$
G1
G2

 $Id + 10 \cdot 10^6$
G2
G3

 $Id + 11 \cdot 10^6$
G3
G1

CSLØT4 Data

Id
G1
G2
G3
G4
p
B
M
N

PLØTEL Data

$Id + 12 \cdot 10^6$
G1
G2

 $Id + 13 \cdot 10^6$
G2
G3

 $Id + 14 \cdot 10^6$
G3
G4

 $Id + 15 \cdot 10^6$
G4
G1

EXECUTIVE PREFACE MODULE IFP5 (INPUT FILE PROCESSOR - PHASE 5)

14. When GEØM2 has been completely copied onto SCRT2, the files are rewound and the data from SCRT2 (containing the new CELAS2 elements) and the PLØTEL data from SCRT1 are merged and copied back onto GEØM2.

4.91.8 Subroutines

4.91.8.1 IFP5A

1. Entry Point: IFP5A
2. Purpose: Prints formal part of messages for IFP5.
3. Calling Sequence: CALL IFP5A (NUM)
NUM = IFP5 message number.

4.91.9 Design Requirements

Discussed under Method.

4.91.10 Diagnostic Messages

Many user messages relevant to the acoustic cavity modeling data may be issued.

FUNCTIONAL MODULE PLTTRAN

4.92 FUNCTIONAL MODULE PLTTRAN

4.92.1 Entry Point: PLTTRA

4.92.2 Purpose

To modify the SIL and BGPDT tables for the purpose of plotting special scalar grid points. Each grid point with one degree of freedom is given six degrees of freedom in the modified SIL data block. These points are identified in the BGPDP data block by the value (-2) in the first entry for each point.

4.92.3 DMAP Calling Sequence

PLTTRAN BGPDT, $\left\{ \begin{smallmatrix} \text{HSIL} \\ \text{SIL} \end{smallmatrix} \right\} / \text{BGPDP}, \left\{ \begin{smallmatrix} \text{HSIP} \\ \text{SIP} \end{smallmatrix} \right\} / \text{V,N,LUSET} / \text{V,N,LUSEP} \$$

4.92.4 Input Data Blocks

Data Block BGPDT - Four entries per grid or scalar point as follows:

1. Local coordinate system number or -1 if point is a scalar point.
- 2-4. X, Y, Z location in basic coordinate system.

Data Block SIL (or HSIL) - One entry per grid or scalar point. The value of the entry is the location of the first degree of freedom of the point in the vector containing all degrees of freedom.

4.92.5 Output Data Blocks

Data Block BGPDP - Same format as BGPDT. If a point is determined to have one degree of freedom and is not a scalar point, the value (-2) is placed in the first entry for that point.

Data Block SIP (or HSIP) - Same format as SIL. All points except time scalar points are given six (6) degrees of freedom. A true scalar point has the value (-1) in the first slot of its BGPDT entry.

4.92.6 Parameters

LUSET - Total number of degrees of freedom.

LUSEP - New value for LUSET taking into account the change in the number of degrees of freedom when the special scalar points are expanded to six degrees of freedom.

MODULE FUNCTIONAL DESCRIPTIONS

4.92.7 Method

The SIL is read 1 word at a time; the BGPDT is read 4 words at a time. If the difference between the new SIL number and the previous SIL value is 1 and the first entry in the BGPDT is zero a fluid scalar grid point exists. In this event the new SIP increment is 6 and the value -2 is placed in the first word of the BGPDP entry.

4.92.8 Subroutines

None.

4.92.9 Design Requirements

Open core is defined at /PLTRN1/ and must be sufficient to hold four (4) GINØ buffers.

4.92.10 Diagnostic Messages

Messages 3001, 3002, 3003, 3008, 5011 and 5012 may be issued.

MATRIX MODULE UPARTN

4.93 MATRIX MODULE UPARTN (PARTITIONS A MATRIX BASED ON USET)

4.93.1 Entry Point: DUPART

4.93.2 Purpose:

To compute a partitioning vector based on the displacement sets as defined by USET and create the symmetric partitions of the input matrix.

For example this module will perform

$$[K_{nn}] \Rightarrow \begin{bmatrix} K_{ff} & K_{fs} \\ K_{sf} & K_{ss} \end{bmatrix}$$

4.93.3 DMAP Calling Sequence

UPARTN USET,KNN / KFF,KSF,KFS,KSS / C,Y,MAJOR=N / C,Y,SUBO=F / C,Y,SUBI=S \$

4.93.4 Input Data Blocks

USET - Displacement set definitions table (This may also be USETD if extra points are present).

KNN - Any square displacement matrix. The associated set of KNN (N) must be given in the first parameter.

- Note: 1. USET may not be purged.
2. If KNN is purged, UPARTN will return.

4.93.5 Output Data Blocks

KFF - Matrix. It will have SUBO rows and columns.

KSF - Matrix. It will have SUBI rows and SUBO columns.

KFS - Matrix. It will have SUBO rows and SUBI columns.

KSS - Matrix. It will have SUBI rows and columns.

Note: Any purged or omitted output data blocks will not be written.

MODULE FUNCTIONAL DESCRIPTIONS

4.93.6 Parameters

MAJØR - Input - BCD - No default value. This is the set of KNN.

SUBØ - Input - BCD - No default value. This is the first subset of MAJØR.

SUB1 - Input - BCD - No default value. This is the second subset of MAJØR.

Note: 1. MAJOR, SUBØ, and SUB1 must be selected from the following list: M,Ø,R,SG,SB,L,
A,F,S,N,G,E,P,NE,FE,D,PS,SA,K and PA.

2. The set equation $MAJØR = SUBØ + SUB1$ should be satisfied.

4.93.7 Method

The module driver, DUPART, checks the compatibility of the parameter data and directly calls UPART and MPART (an entry point in UPART). All work is then accomplished in the UPART routine.

4.93.8 Subroutines

UPART - See subroutine description, section 3.5.9.

4.93.9 Design Requirements

One scratch file.

4.93.10 Diagnostic Messages

UPARTN may issue one of the following diagnostic messages:

3007 or 3059

MATRIX MODULE UMERGE

4.94 MATRIX MODULE UMERGE (MERGES TWO MATRICES BASED ON USET).

4.94.1 Entry Point: DUMERG

4.94.2 Purpose:

To merge two matrices into a third based on the displacement sets. For example, this module will perform:

$$\left\{ \begin{array}{c} \phi_a \\ \phi_o \end{array} \right\} \Rightarrow \left\{ \phi_f \right\}$$

4.94.3 DMAP Calling Sequence

UMERGE USET,PHIA,PHI0 / PHIF / C,Y,MAJOR=F / C,Y,SUB0=A / C,Y,SUB1=0 \$

4.94.4 Input Data Blocks

USET - Displacement set definitions table (this may also be USETD if extra points are present.)

PHIA - Any two matrices except that their rows must be associated with degrees-of-freedom specified by USET and the parameter list. PHIA's degrees-of-freedom are specified by SUB0 and PHI0's by SUB1.

Note: Either matrix may not be present and its respective degrees-of-freedom will be filled with zeros.

4.94.5 Output Data Blocks

PHIF - Matrix. Its terms will be associated with degrees-of-freedom in the set specified by MAJOR.

Note: PHIF must be present.

4.94.6 Parameters

MAJOR - Input - BCD - No default value. This is the set of PHIF.

SUB0 - Input - BCD - No default value. This is the set of PHIA.

SUB1 - Input - BCD - No default value. This is the set of PHI0.

MODULE FUNCTIONAL DESCRIPTIONS

- Note: 1. MAJØR, SUB0 and SUB1 must be selected from the following list: M, Ø, R, SG, SB, L, A, F, S, N, G, E, NE, FE, D, PS, SA, K and PA.
2. The set equation $MAJØR = SUB0 + SUB1$ should be satisfied.

4.94.7 Method

The module driver, DUMERG, checks the compatibility of the parameter data and directly calls SDR1B. All work is then accomplished in the SDR1B routine.

4.94.8 Subroutines

SDR1B - See its subroutine description, section 3.5.

4.94.9 Design Requirements

One scratch file.

4.94.10 Diagnostic Messages

UMERGE may issue one of the following diagnostic messages:

3007 or 3059

MATRIX MODULE VEC

4.95 MATRIX MODULE VEC (CREATES PARTITIONING VECTOR BASED ON USET)

4.95.1 Entry Point: VEC

4.95.2 Purpose:

VEC creates a partitioning vector based on USET that may be used in PARTN and MERGE.

4.95.3 DMAP Calling Sequence

VEC USET / V / C,N,SET / C,N,SET0 / C,N,SET1 \$

4.95.4 Input Data Blocks

USET - Displacement set definition table (this may be USETD if extra points are present).

Note: USET must be present.

4.95.5 Output Data Blocks

V - Partitioning vector.

Note: V may not be purged.

4.95.6 Parameters

SET - Input-BCD-no default. SET indicates the set to which the partitioning vector applies.

SET0 - Input-BCD-no default. SET0 indicates the upper partition of SET.

SET1 - Input-BCD-no default. SET1 indicates the lower partition of SET.

4.95.7 Method

The BCD parameters SET, SET0, and SET1 are converted to bit positions in USET. They must be one of the following 17 symbols: M,Ø,R,SG,SB,L,A,F,S,N,G,E,P,NE,FE,D,H or else a fatal error will result.

USET is read into core and the file closed. The output file is then opened and each entry is compared with the three converted parameters as follows:

1. USET is searched for members of SET. If the entry is not a member of SET, it is checked that it is not a member of SET0 or SET1 before going to the next entry.

MODULE FUNCTIONAL DESCRIPTIONS

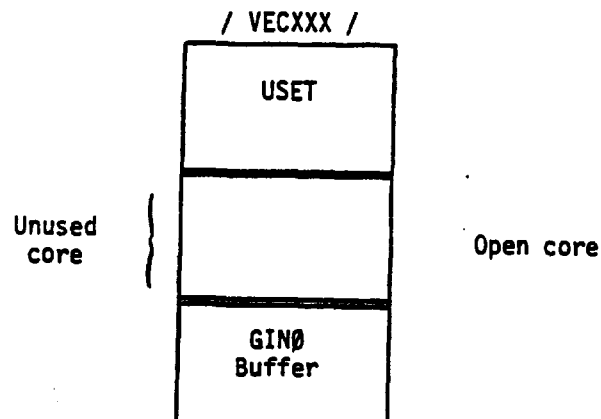
2. The entry that belongs to SET is then checked if it is also a member of SET1. If it is, the entry is also checked if it is a member of SET0, which is fatal, before replacing the entry with 1.0.
3. If the entry is a member of SET and not a member of SET1, the entry is checked to verify that it is a member of SET0 before replacing it with a 0.0.
4. After all entries have been successfully processed, a check is made to insure that a vector exists and that the entries are not all zeros or ones (fatal error).
5. The rewritten entries are then written onto the output data block as a matrix consisting of one (1) column.

4.95.8 Subroutines

VEC has no auxiliary subroutines.

4.95.9 Design Requirements

1. Open core is defined at / VECXXX /
2. Layout of open core is as follows:



MATRIX MODULE ADD5 (ADD MATRICES)

4.96 MATRIX MODULE ADD5 (ADD MATRICES)

4.96.1 Entry Point: DADD5

4.96.2 Purpose

To compute $[X] = \alpha[A] + \beta[B] + \gamma[C] + \delta[D] + \epsilon[E]$.

4.96.3 DMAP Calling Sequence

```
ADD  A,B,C,D,E/X/C,Y,ALPHA=(1.0,2.0)/C,Y,BETA=(3.0,4.0)/C,Y,GAMMA=(5.0,6.0)/  
      C,Y,DELTA=(7.0,8.0)/C,Y,EPSLN=(9.0,0.0) $
```

4.96.4 Input Data Blocks

A, B, C, D, and E must be distinct matrices.

Note: Any of the input matrices may be purged.

4.96.5 Output Data Blocks

X - Matrix.

The type of X is maximum of the types of A, B, C, D, E, α , β , γ , δ , ϵ . The form of X is the form of A if A is present. Otherwise it is that of the first non-purged input.

Note: X cannot be purged.

4.96.6 Parameters

ALPHA - Input-complex default value = 1,0. This is the scalar multiplier for A.

BETA - Input-complex default value = 1,0. This is the scalar multiplier for B.

GAMMA - Input-complex default value = 1,0. This is the scalar multiplier for C.

DELTA - Input-complex default value = 1,0. This is the scalar multiplier for D.

EPSLN - Input-complex default value = 1,0. This is the scalar multiplier for E.

Note: If $\text{Im}(\alpha)$, $\text{Im}(\beta)$, $\text{Im}(\gamma)$, $\text{Im}(\delta)$ or $\text{Im}(\epsilon) = 0.0$, the parameter will be considered real.

4.96.7 Method

If [A] is not purged, the number of columns, rows, and form of [X] = number of columns, rows, and form of [A]. Otherwise the descriptors of the first non-purged input are used. The type of

MODULE FUNCTIONAL DESCRIPTIONS

[X] is the maximum compatible type of [A], [B], [C], [D], [E], ALPHA, BETA, GAMMA, DELTA and EPSLN. ALPHA, BETA, GAMMA, DELTA and EPSLN are assumed to be real if their imaginary parts are zero.

4.96.8 Subroutines

SADD - See subroutine description, Section 3.5.26.

4.96.9 Design Requirements

Open core is defined at /DADDA/.

4.96.10 Diagnostic Messages

None.

FUNCTIONAL MODULE INPUT (INPUT GENERATOR)

4.97 FUNCTIONAL MODULE INPUT (INPUT GENERATOR)

4.97.1 Entry Point

INPUT (See also Section 2.6 of NASTRAN User's Manual)

4.97.2 Purpose

Generates part of the bulk data input for a large number of academic test problems. It is also intended as an example of a user module with FØRTRAN read statements.

4.97.3 DMAF Calling Sequence

INPUT GEØM1,GEØM2,GEØM3,GEØM4, / Ø1,Ø2,Ø3,Ø4,Ø5 / C,N,α / C,N,ß / C,N,δ \$

4.97.4 Input Data Blocks

GEØMi - Preface files to be merged with data to be generated by the module.

4.97.5 Output Data Blocks

Øi - Modified output file corresponding to GEØMi as required by the execution of the module.

4.97.6 Parameters

α - Problem type selector - Input, integer, default value = -1 (an illegal value for execution).

ß - Problem type option selector - Input, integer, default value = 0.

δ - Problem type option selector - Input, integer, default value = 0.

4.97.7 Method

Based on the values of the parameters, INPUT reads one or more data cards from the input stream using standard FØRTRAN read statements. Since the data deck has already been processed through the ENDDATA card at this point, these data cards always follow the ENDDA1A card. Since FØRTRAN I/Ø is used, integer data on these cards must be right-justified. Once the data cards are read and checked, INPUT generates the table data blocks that would have been generated if the equivalent actual cards had appeared in the bulk data deck. These generated records are added by INPUT to those coming from the corresponding input data file (generated by IFP) and are written onto the appropriate output data file.

MODULE FUNCTIONAL DESCRIPTIONS

Note that the data records to be created by INPUT must not include any of the data card types generated by the preface, IFP. That is, data cannot be appended to records already existing in the GEØMi files.

The following table defines the equivalence between input (GEØMi)/output (Øi) files and the data records generated under user selection by INPUT. The input files are optional and the data records generated are dependent on the FØRTRAN input.

Parameters			Files		Data Records Generated
α	β	γ	Input	Output	
1 - Laplace Network	1	-	GEØM2	Ø2	CELAS4,CNGRNT*
	2-3	-	GEØM2	Ø2	CELAS4,CMASS4,CNGRNT*
	1	-	GEØM4	Ø4	SPC
2 - Rectangular Network	1	0-3	GEØM1	Ø1	GRID
	2-4	0-3	GEØM1	Ø1	GRID,SEQGP
	1-4	0	GEØM2	Ø2	CBAR,CNGRNT*
	1-4	1-3	GEØM2	Ø2	CRØD,CNGRNT*
3 - Rectangular Plate	1	-	GEØM1	Ø1	GRID
	2-4	-	GEØM1	Ø1	GRID,SEQGF
	1-4	-	GEØM2	Ø2	CQUAD1,CNGRNT*
	1	-	GEØM4	Ø4	SPC
	2-4	-	GEØM4	Ø4	SPC,SEQGP
4 - Rectangular Plate	1	-	GEØM1	Ø1	GRID
	2-4	-	GEØM1	Ø1	GRID,SEQGP
	1-4	-	GEØM2	Ø2	CTRIA1,CNGRNT*
	1	-	GEØM4	Ø4	SPC
	2-4	-	GEØM4	Ø4	SPC,SEQGP
5 - N-Cell String	-	-	GEØM2	Ø2	CDAMP4,CELAS4,CMASS4,CNGRNT*
6 - N-Cell Bar	-	-	GEØM1	Ø1	GRID
	-	-	GEØM2	Ø2	CBAR,CNGRNT*
	-	-	GEØM4	Ø4	ØMIT
7 - Full Matrix	-	-	GEØM2	Ø2	CELAS4,CNGRNT*
	-	-	GEØM3	Ø3	SLØAD
	-	-	-	Ø5	SPØINT
8 - Spoked Wheel	-	-	GEØM1	Ø1	GRID
	-	-	GEØM2	Ø2	CBAR,CNGRNT*

*The CNGRNT feature may be suppressed only if SYSTEM(57) \neq 0 (the default value is 0).

FUNCTIONAL MODULE INPUT (INPUT GENERATOR)

4.97.8 Subroutines

IUNION - Integer function which computes the union of constraint codes.

INPABD - Initializes the common block /INPUTA/.

4.97.9 Design Requirements

Open core is defined at /INPUTX/ and must be sufficient to hold two GINØ buffers.

4.97.10 Diagnostic Messages

Many user messages are generated by INPUT. These are mostly related to improper or inconsistent data presented by the user and are usually self-explanatory. The messages generated internally within INPUT are 1738 through 1745. In addition, INPUT writes an echo of all data read from the input stream and certain informational output related to the processing that occurs while generating the user's problem data.

FUNCTIONAL MODULE INPUTT1

4.98 FUNCTIONAL MODULE INPUTT1

4.98.1 Entry Point

INPTT1

4.98.2 Purpose

Recovers GINØ-written data blocks (tables or matrices) from User Tapes designated for that purpose (NASTRAN permanent GINØ files INPT, INP1, INP2, ---, and/or INP9). Normally, these tapes would be written by the companion module ØUTPUT1 (see Section 4.100) in a previous run.

4.98.3 DMAP Calling Sequence

INPUTT1 / Ø1,Ø2,Ø3,Ø4,Ø5 / V,N,P1 / V,N,P2 / V,N,P3 \$

4.98.4 Input Data Blocks

None.

4.98.5 Output Data Blocks

Øi - Any data block which is to be recovered from the User Tape. Purged outputs (either implicit or explicit) are ignored.

4.98.6 Parameters

P1 - Tape positioning option (Input, integer, default value = 0)

P2 - User Tape code (Input, integer, default value = 0)

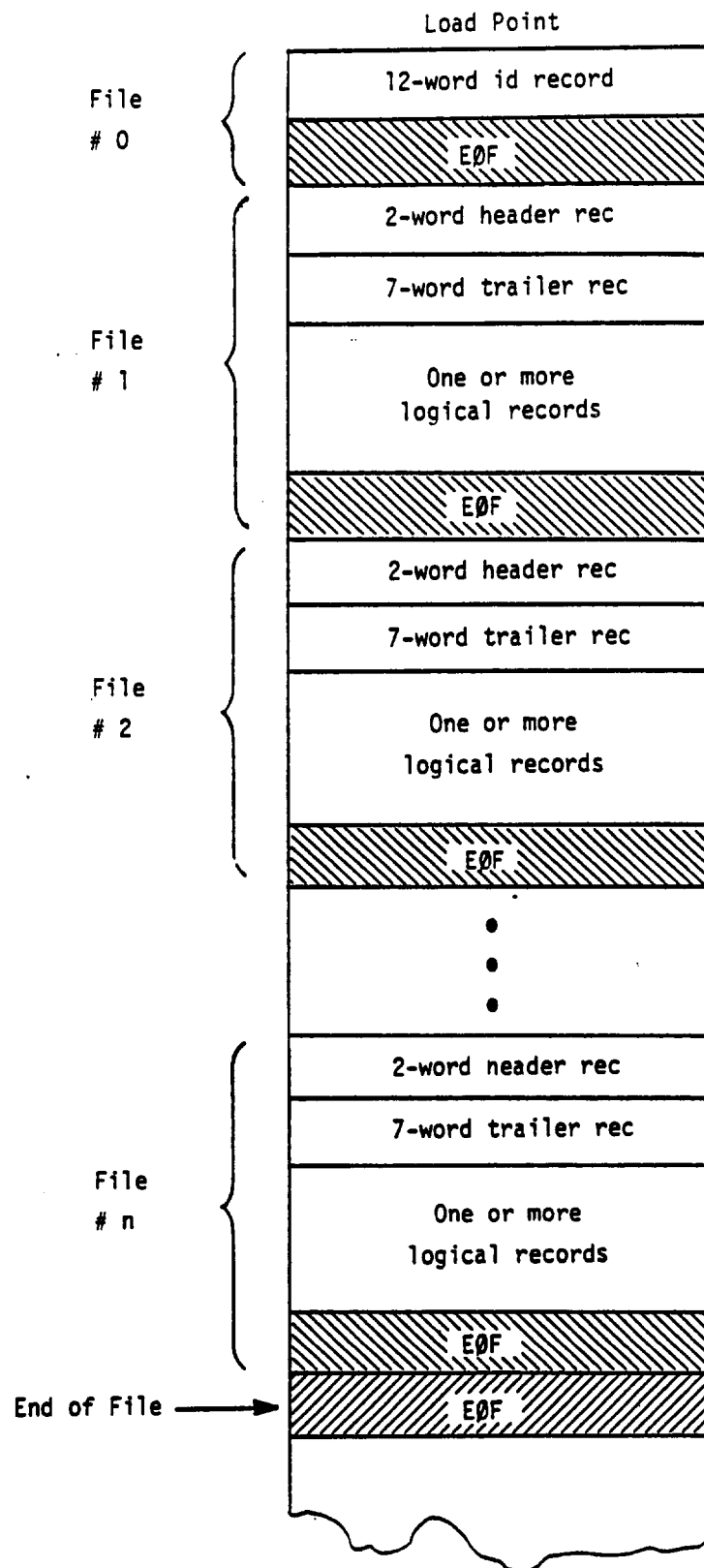
P3 - User Tape Label (Input, alphanumeric, default value = 'XXXXXXXX')

4.98.7 Method

INPTT1 examines the first parameter and positions the User Tape designated by the second parameter (checking the User Tape Label defined by the third parameter if appropriate). INPTT1 then copies the next data blocks from the User Tape and writes them on the non-purged output data blocks in the DMAP instruction. The User Tape is left positioned wherever it is when the DMAP instruction requirements are satisfied. In this way, multiple calls can be made.

MODULE FUNCTIONAL DESCRIPTIONS

The configuration of files and records on the User Tape is shown in the sketch below.



4.98.8 Subroutines

None.

4.98.9 Design Requirements

Open core is defined at /INPLXX/ and must be sufficient to hold two GINØ buffers plus one word of working core. Since blast I/Ø techniques are used, efficiency is enhanced by any additional core up to the longest logical record to be read in any one data block.

The User Tapes must be physical tapes.

4.98.10 Diagnostic Messages

Messages 3008, 4105, 4106, 4107, 4108, 4109, 4110, 4111, 4112, 4113, 4127, 4132, 4133, 4134, 4135, 4136, 4137, 4138, 4139, 4140, 4141, and 4142 may be issued.

In addition, when a file table of contents is requested, printout is generated giving the file number and the value of the first two words of the header record for each 'file' on the tape.

FUNCTIONAL MODULE INPUTT2

4.99 FUNCTIONAL MODULE INPUTT2

4.99.1 Entry Point:

INPTT2

4.99.2 Purpose

Recovers FORTRAN-written data blocks (tables or matrices) from User Tapes. Any legitimate FORTRAN unit number not already utilized by NASTRAN may be used for this purpose. On the CDC machines, these unit numbers must also be compiled into the system in deck NASTRAN. Normally, these files would be written by the companion module OUTPUT2 (see Section 4.101) in a previous run. It is intended that files also be easily generated by external FORTRAN programs, however.

4.99.3 DMAP Calling Sequence

INPUTT2 / 01,02,03,04,05 / V,N,P1 / V,N,P2 / V,N,P3 \$

4.99.4 Input Data Blocks

None.

4.99.5 Output Data Blocks

01 - Any data block which is to be recovered from the User Tape. Purged outputs (either implicit or explicit) are ignored.

4.99.6 Parameters

P1 - File Positioning Option (Input, integer, default value = 0)

P2 - User Tape Code (Input, integer, default value = 11. This is the FORTRAN unit number for the file.

P3 - User Tape Label (Input, Alphanumeric, default value = 'XXXXXXX'.)

4.99.7 Method

INPTT2 examines the first parameter and positions the User Tape designed by the second parameter (checking the User Tape Label defined by the third parameter if appropriate). INPTT2 then copies the next data blocks from the User Tape and writes them (via GIN0) on the non-purged output data blocks in the DMAP instruction.

MODULE FUNCTIONAL DESCRIPTIONS

The User Tape is left positioned wherever it is when the DMAP instruction requirements are satisfied. In this way, multiple calls may be made.

A description of the file configuration is given below for those FØRTRAN programmers who may wish to generate User Tapes with their own external programs for input to NASTRAN.

Format of INPUT2/ØUTPUT2 File

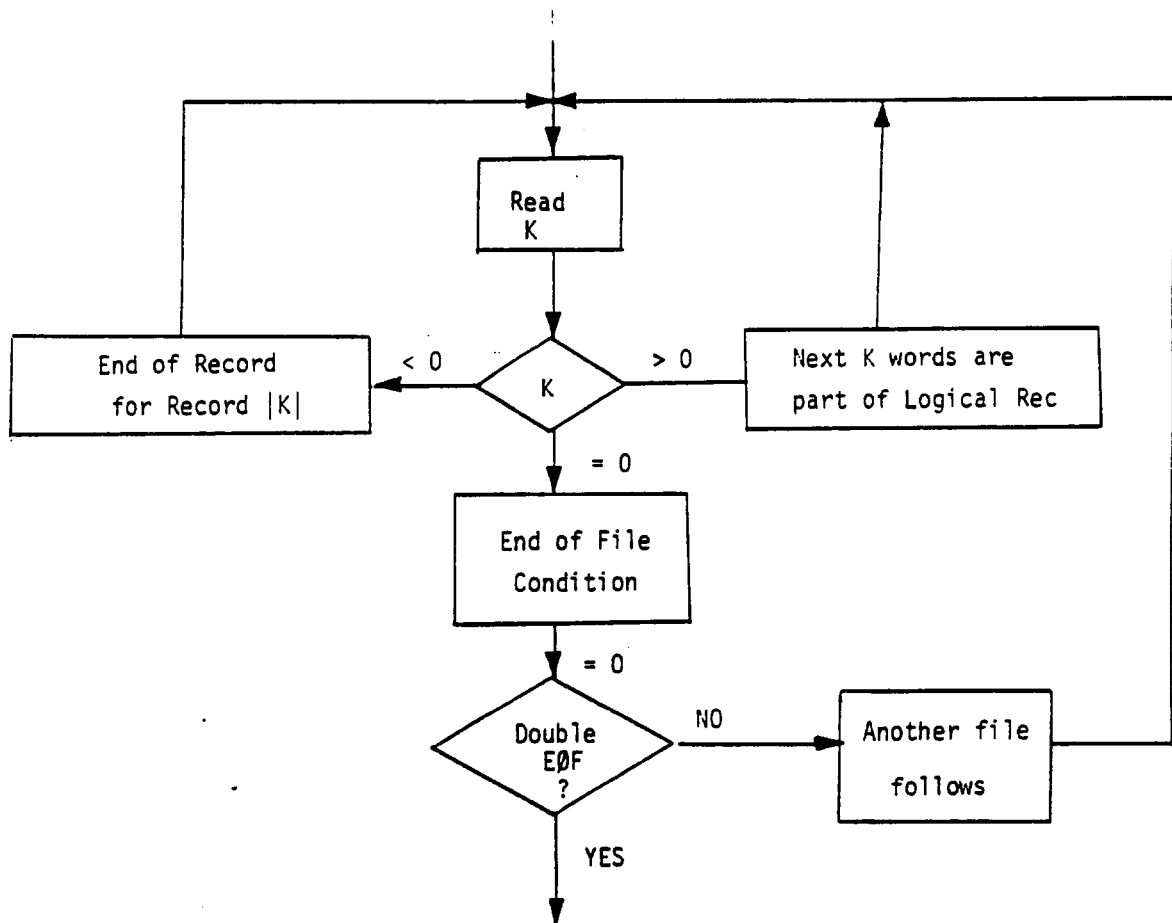
<u>NASTRAN File</u>	<u>Rec</u>		<u>FØRTRAN Rec</u>	<u>Length</u>
1	1	KEY > 0	1	1
		{ Data } ↔ KEY	2	KEY
		KEY > 0	3	1
		{ Data } ↔ KEY	4	KEY
		KEY < 0 EØR	5	1
	2	KEY > 0	6	1
		{ Data } ↔ KEY	7	KEY
		KEY < 0 EØR	8	1
		KEY = 0 EØF	9	1
2	1	KEY > 0	10	1
		{ Data } ↔ KEY	11	KEY
		KEY < 0 EØR	12	1
		KEY = 0 EØF	13	1
3		KEY = 0 EØF = EØD	14	1

Restrictions:

1. Enough core must be available to hold the longest record segment.
2. A FØRTRAN unit must be available. On the CDC, this means that the PRØGRAM Deck (NASTRAN) must be re-compiled and Link 0 re-done if the value used for parameter P2 is other than 11.

FUNCTIONAL MODULE INPUTT2

The logic by which the NASTRAN logical 'records' are interrogated is given in the sketch below:



4.99.8 Subroutines

None.

4.99.9 Design Requirements

Open core is defined at /INP2XX/ and must be sufficient to hold two GINØ buffers plus the longest FØRTRAN logical record on the User Tape. The 'User Tape' files may be on any FØRTRAN readable device.

4.99.10 Diagnostic Messages

Messages 2187, 2190, 3008, 4105, 4106, 4107, 4108, 4109, 4110, 4111, 4112, 4113, 4132, 4133, 4134, 4135, 4136, 4137, 4138, 4139, 4140, 4141, and 4142 may be issued.

FUNCTIONAL MODULE ØUTPUT1

4.100 FUNCTIONAL MODULE ØUTPUT1

4.100.1 Entry Point:

ØUTPT1

4.100.2 Purpose

Creates GINØ-written User Tapes containing data blocks (tables or matrices) as requested by the user via the DMAP instruction. These tapes are written on NASTRAN permanent GINØ files INPT, INP1, INP2, ---, and/or INP9. It is anticipated that these tapes will be read by the companion module INPUTT1 in a subsequent run.

4.100.3 DMAP Calling Sequence

ØUTPUT1 I1,I2,I3,I4,I5 // V,N,P1 / V,N,P2 / V,N,P3 \$

4.100.4 Input Data Blocks

Ii - Any data block which the user desires to be written on a User Tape. Purged inputs (either implicit or explicit) are ignored.

4.100.5 Output Data Blocks

None.

4.100.6 Parameters

P1 - Tape positioning option (Input, integer, default value = 0)

P2 - User Tape Code (Input, integer, default value = 0)

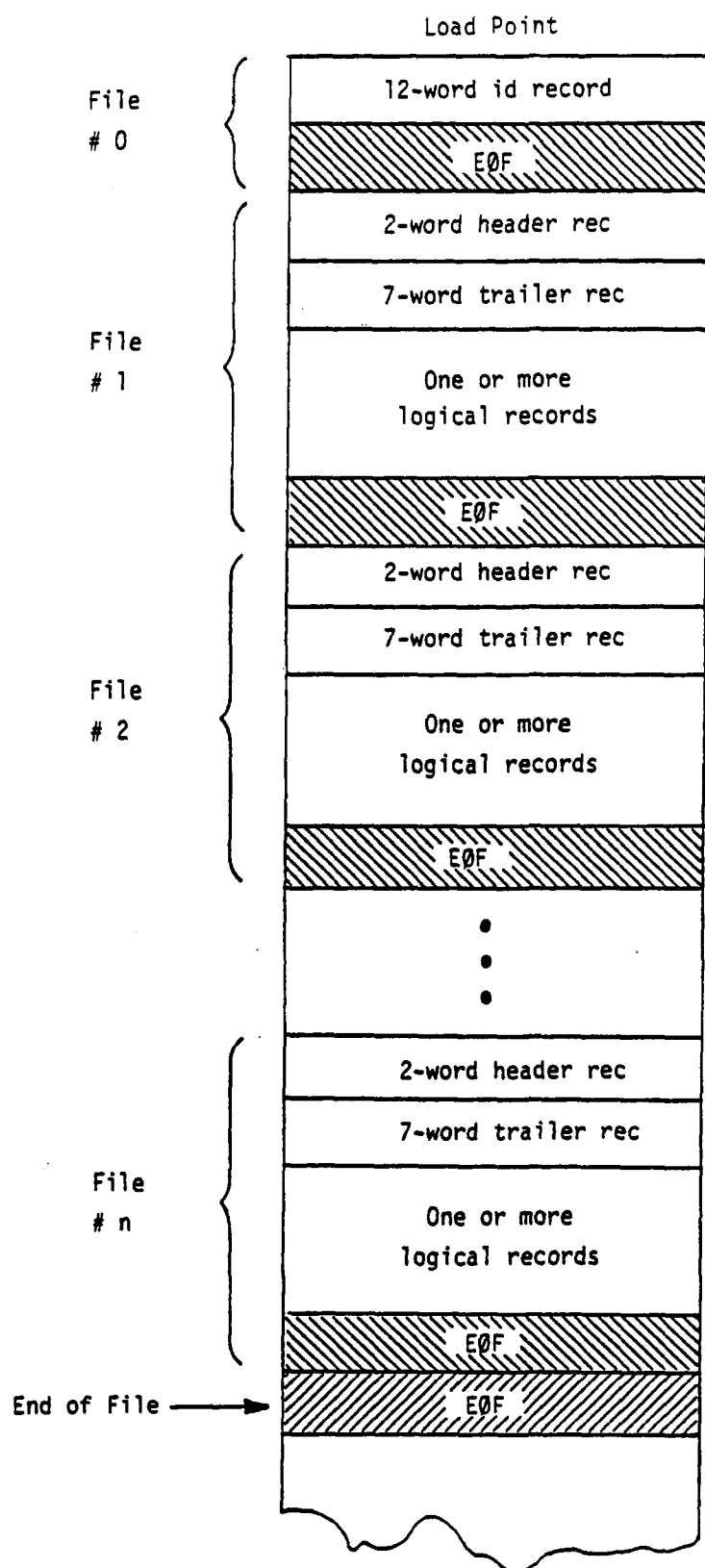
P3 - User Tape Label (Input, Alphanumeric, default value = 'XXXXXXXX')

4.100.7 Method

ØUTPT1 examines the first parameter and positions the User Tape designated by the second parameter (checking or writing the User Tape Label defined by the third parameter if appropriate). ØUTPT1 then writes onto the User Tape all non-purged input data blocks in the DMAP instruction. The User Tape is left positioned wherever it is when the DMAP instruction requirements are satisfied. In this way, multiple calls may be made to write as much material as desired on each User Tape.

MODULE FUNCTIONAL DESCRIPTIONS

The configuration of files and records on the User Tape is shown in the sketch below.



4.100.8 Subroutines

None.

4.100.9 Design Requirements

Open core is defined at /ØUT1XX/ and must be sufficient to hold two GINØ buffers plus one word of working core. Since blast I/Ø techniques are used, efficiency is enhanced by any additional core up to the longest logical record to be read in any one data block.

The User Tapes must be physical tapes.

4.100.10 Diagnostic Messages

Messages 3008, 4114, 4115, 4116, 4117, 4118, 4119, 4120, 4127, 4128, 4129, 4130, and 4131 may be issued.

In addition, when a file table of contents is requested, printout is generated giving the file number and the value of the first two words of the header record for each 'file' on the tape.

FUNCTIONAL MODULE ØUTPUT2

4.101 FUNCTIONAL MODULE ØUTPUT2

4.101.1 Entry Point:

ØUTPT2

4.101.2 Purpose

Creates FØRTRAN-written User Tapes containing data blocks (tables or matrices) as requested by the user via the DMAP instruction. Any legitimate FØRTRAN unit number not already utilized by NASTRAN may be used for this purpose. On the CDC machine, these unit numbers must also be compiled into the system in deck NASTRAN. Normally, it is anticipated that these files will be read by the companion module INPUTT2 in a subsequent run. It is expected, however, that users will want to generate their own User Tapes with external programs completely unrelated to NASTRAN. Towards this end, a scheme has been implemented by which NASTRAN-like logical files and records can be simulated by unformatted FØRTRAN I/Ø calls.

4.101.3 DMAP Calling Sequence

ØUTPUT2 I1,I2,I3,I4,I5 // V,N,P1 / V,N,P2 / V,N,P3 \$

4.101.4 Input Data Blocks

Ii - Any data block which the user desires to be written on a User Tape file. Purged inputs (either implicit or explicit) are ignored.

4.101.5 Output Data Blocks

None.

4.101.6 Parameters

P1 - File positioning option (Input, integer, default value = 0)

P2 - User Tape Code (Input, integer, default value = 11). This is the FORTRAN unit number for the file.

P3 - User Tape Label (Input, Alphanumeric, default value = 'XXXXXXXX')

MODULE FUNCTIONAL DESCRIPTIONS

4.101.7 Method

ØUTPT2 examines the first parameter and positions the User Tape designated by the second parameter (checking or writing the User Tape Label defined by the third parameter if appropriate). ØUTPT2 then writes onto the User Tape file all non-purged input data blocks in the DMAP instructions. The User Tape file is left positioned wherever it is when the DMAP instruction requirements are satisfied. In this way, multiple calls may be made to write as much material as desired on each User Tape file.

A description of the file configuration is given below for those FØRTRAN programmers who may wish to generate User Tape files with their own internal programs for input to NASTRAN.

Format of INPUTT2/ØUTPUT2 File

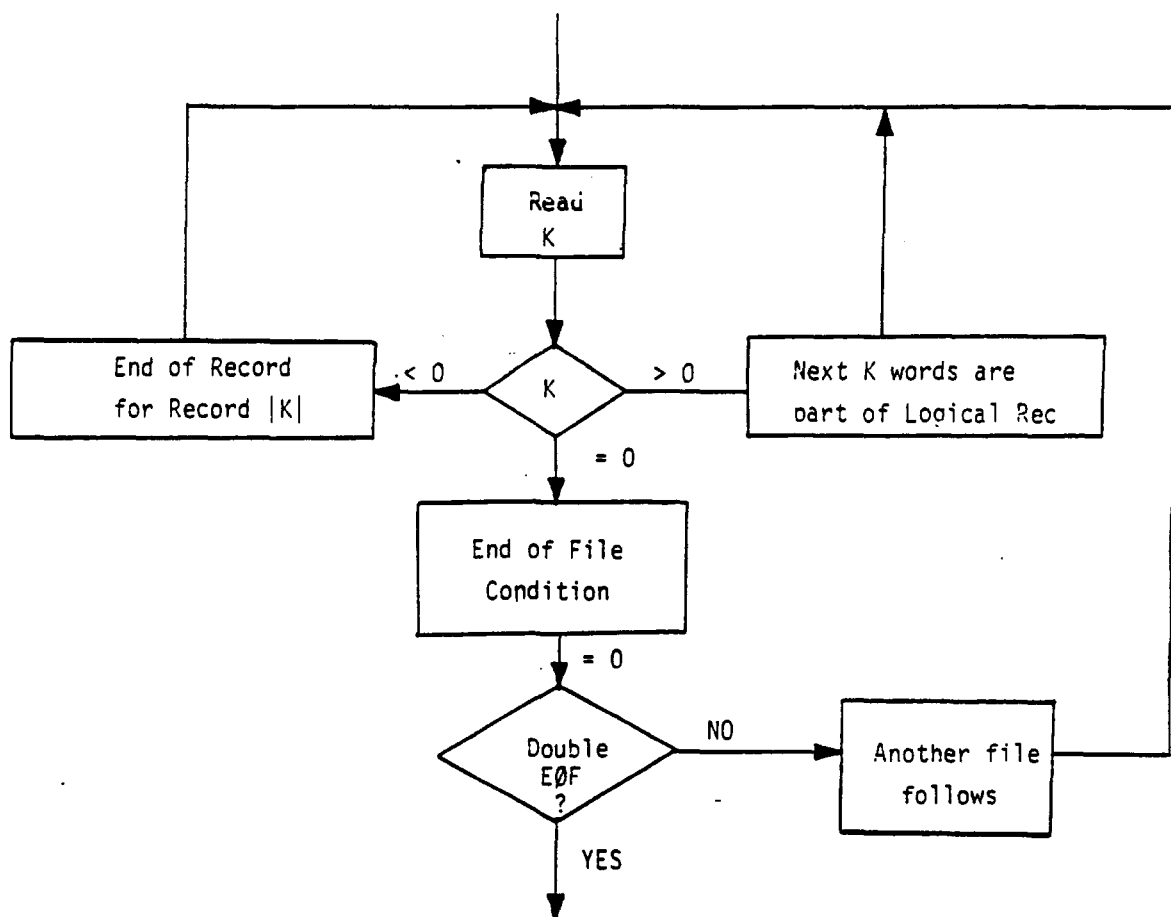
NASTRAN File	Rec		FØRTRAN Rec	Length
1	1	KEY > 0	1	1
		{Data} ↔ KEY	2	KEY
		KEY > 0	3	1
		{Data} ↔ KEY	4	KEY
		KEY < 0 EØR	5	1
	2	KEY > 0	6	1
		{Data} ↔ KEY	7	KEY
		KEY < 0 EØR	8	1
		KEY = 0 EØF	9	1
2	1	KEY > 0	10	1
		{Data} ↔ KEY	11	KEY
		KEY < 0 EØR	12	1
		KEY = 0 EØF	13	1
		KEY = 0 EØF = EØD	14	1

Restrictions:

1. Enough core must be available to hold the longest record segment.
2. A FØRTRAN unit must be available. On the CDC, FØRTRAN Unit 11 is assigned for this purpose.

FUNCTIONAL MODULE ØUTPUT2

The logic by which the NASTRAN logical 'records' are interrogated is given in the sketch below:



4.101.8 Subroutines

None.

4.101.9 Design Requirements

Open core is defined at /ØUT2XX/ and must be sufficient to hold two GINØ buffers plus the longest FØRTRAN logical record on the User Tape. The 'User Tape' files may be on any device accessible to the FØRTRAN I/Ø routines.

4.101.10 Diagnostic Messages

Messages 2187, 2190, 3008, 4114, 4115, 4116, 4118, 4119, 4120, 4128, 4129, 4130, and 4131 may be issued.

OUTPUT MODULE ØUTPUT3

4.102 OUTPUT MODULE ØUTPUT3

4.102.1 Entry Point:

ØUTPT3

4.102.2 Purpose

Punches onto DMI cards the contents of matrix data blocks. Single-precision values are output on double-field cards. If full square matrices are considered, a maximum order of 196 is imposed since a maximum of 10000 cards (including the header card) are allowed.

4.102.3 DMAP Calling Sequence

ØUTPUT3 I1,I2,I3,I4,I5 // V,N,PRINTØPT / V,N,N1 / V,N,N2 / V,N,N3 / V,N,N4 / V,N,N5 S

4.102.4 Input Data Blocks

Ii - Any real matrix data block. Only sufficiently small non-purged data blocks will be punched onto DMI cards.

4.102.5 Output Data Blocks

None.

4.102.6 Parameters

PRINTØPT - Print echo option (Input, integer, default value = 0)
If this parameter is negative, an echo of the DMI card images generated will be printed on the FØRTRAN unit given by -PRINTØPT.

Ni - Continuation string - (Input, alphanumeric, default values: N1,NO default; N2-N5, default value = 'XXX'). Used to form a unique continuation string for the DMI cards.

4.102.7 Method

ØUTPT3 reads each matrix, non-zero term by non-zero term (by column), and passes these items to the DMI card-punching subroutine PHDMIA.

MODULE FUNCTIONAL DESCRIPTIONS

4.102.8 Subroutines

4.102.8.1 Subroutine Name: PHDMIA

1. Entry Points: PHDMIA, PHDMIB, PHDMIC, PHDMID
2. Purpose: To collect and punch DMI card images.
3. Calling Sequence:

CALL PHDMIA - Initializes matrix.

CALL PHDMIB - Initializes non-null column.

CALL PHDMIC - Collect each non-zero term of column.

CALL PHDMID - Wraps up column.

COMMON / PHDMIX / N(2),C,IF0,TIN,TOUT,IR,IC,N0,KPP,NLP,ERN0,IC0L,IR0,XX

Communication area for PHDMIA.

4. Method: A single call is made to PHDMIA for each matrix data block to be punched.
A call is made to PHDMIB for each non-null column, followed by a call to PHDMIC for each non-zero term in the column, followed by a wrap-up call to PHDMID.

4.102.9 Design Requirements

Open core is defined at /OUT3XX/ and must be sufficient to hold a single GIN0 buffer.

4.102.10 Diagnostic Messages

Messages 3008, 4100, 4101, 4102, 4103, and 4104 may be issued.

OUTPUT MODULE TABPRT (FORMATTED TABLE PRINTER)

4.103 OUTPUT MODULE TABPRT (FORMATTED TABLE PRINTER)

4.103.1 Entry Point:

TABFMT

4.103.2 Purpose

To print selected table data blocks with format for ease of reading.

4.103.3 DMAP Calling Sequence

TABPRT TDB // V,N,KEY / V,N,ØPT1 / V,N,ØPT2 \$

4.103.4 Input Data Blocks

TDB - Table Data Block having a format which is processable by the routine.

4.103.5 Output Data Blocks

None.

4.103.6 Parameters

KEY - Keyword (Input, alphanumeric, no default value)
The value of KEYWORD identifies the format to be used in printing the table.

ØPT1 - Option (Input, integer, default value = 0)

ØPT2 - Option (Input, integer, default value = 0)

4.103.7 Method

TABFMT examines the parameters and selects a format for printing the contents of the input data block. The input data block is then read record by record and the contents printed according to the selected format. A line of printout may contain part of a logical record, a complete logical record, or more than one logical record. Coding or decoding of the data items encountered may be done as the programmer wishes. The trailer data are also printed.

4.103.8 Subroutines

TABFBD - Block Data routine to set tables for TABFMT in common block /TABFTX/ .

MODULE FUNCTIONAL DESCRIPTIONS

4.103.9 Design Requirements

Open core is defined at /TABFTZ/ and must be sufficient to hold one GINØ buffer plus a small number of words of data, the number of which depends on KEY.

4.103.10 Diagnostic Messages

Fatal message 3008 may be issued.

Warning messages 2094, 2095, 2096, 2097, 2098, and 2099 may be issued and will cause termination of the module but not of the program.

FUNCTIONAL MODULE RMG (RADIATION MATRIX GENERATOR)

4.104 FUNCTIONAL MODULE RMG (RADIATION MATRIX GENERATOR)

4.104.1 Entry Point: RMG

4.104.2 Purpose

This module processes the user supplied radiation exchange coefficients to produce temperature heat flux transfer matrices. The connections, area, and emissivity for each element are supplied in the HBDY section of the EST (Element Summary Table) data block. The radiation exchange coefficient matrix data is supplied by the MATPØØL (Matrix Pool Data Block).

This module is written such that either single or double precision operations may be chosen.

4.104.3 DMAP Calling Sequence

RMG HEST,MATPØØL,GPTT,HKGGX / HRGG,HQGE,HKGG / V,Y,TABS / V,Y,SIGMA / V,N,NLR / V,N,LUSET \$

4.104.4 Input Data Blocks

HEST	Element Summary Table
MATPØØL	Matrix Pool
GPTT	Grid Point Temperature Table
HKGGX	Element Conductivity Matrix [k_{gg}^x]

4.104.5 Output Data Blocks

HRGG	Radiation Transfer Matrix
HQGE	Element Heat Flux-Grid Point Temperature Transformation Matrix
HKGG	Total Linear Heat Transfer Matrix

4.104.6 Parameters

TABS	Input, value of user temperature origin in absolute scale.
SIGMA	Input, value of Stefan Boltzmann constant σ .
NLR	Output, radiation matrices exist if NLR = +1, otherwise NRL = -1.
LUSET	Input, length of g-sized vector.

4.104.7 Method

The following steps are taken in the order indicated:

1. The RADLST data are read from the MATP00L data block and stored in core. This is a simple list of integers, ER, of length N_e . The RADMTX data is in the form of the lower triangle part of a symmetric matrix. The packed triangular matrix is read and stored in core (single precision). The complete matrix, F, is formed a column at a time, packed, and output on scratch file SCRT1.
2. The section of the EST (Element Summary Table) containing data for the HBDY element is read an element at a time. If the element is part of the ER list, it is processed by the HBDY subroutine to produce the following data:

A_i	Total area of element i.
E_i	Emissivity of element surface.
$\{SIL_i\}$	One to four integers defining the connected degrees of freedom in the U_g vector.
$\{G_{gi}\}$	One to four values defining the fraction of the total area of element i associated with each connected point.

The data is stored in a table corresponding to the ER list of elements. If an element in the ER list is not defined in the EST, a fatal error occurs. In subsequent steps, the values A_i and E_i will be used as if they existed in separate diagonal matrices.

3. The calculation of the R^e matrix proceeds as follows:

The equation to be solved in this phase is

$$[R^e] = \sigma[A][E][I - XE], \quad (1)$$

where

$$[X] = \{[A] - [F]([I] - [E])\}^{-1}[F]. \quad (2)$$

The diagonal matrices $[A]$ and $[E]$ are stored in core in the tables $A_i, E_i, i = 1, 2,$

. . . N_e . The solution steps are:

- a. Form the matrix $[Y] = [A] - [F]([I] - [E])$ a column at a time by reading the matrix $[F]$ a column at a time. For each term F_{ij} ,

FUNCTIONAL MODULE RMG (RADIATION MATRIX GENERATOR)

$$Y_{ij} = -F_{ij}(1-E_j), i \neq j,$$

or

(3)

$$Y_{ij} = A_i - F_{ij}(1-E_j), i = j.$$

(Any null columns of Y_{ij} will cause a fatal error.)

- b. The DECØMP and GFBS subroutines are called to calculate the matrix $[X]$ where

$$[Y][X] = [F]. \quad (4)$$

(The matrix $[Y]$ may be nonsymmetric.)

- c. The matrix $[R^e]$ may be formed a column at a time by reading $[X]$ a column at a time and calculating

$$[R_{ij}^e] = -\sigma E_i E_j A_i X_{ij}, i \neq j,$$

or

(5)

$$[R_{ij}^e] = \sigma A_i E_i (1-E_i X_{ij}), i = j.$$

4. Calculation of output matrices - The requested temperature set for material properties is found in common block /SYSTEM/. The GPTT data block is read and the given values are stored as a vector, U_g^1 , with length of LUSSET. The terms in the diagonal matrix $[S]$ are

$$S_{gg} = 4(U_g^1 + T_a)^3, \quad (6)$$

where T_a is given by parameter TABS. The terms of S_{gg} replace the U_g^1 values in core. If no temperature set is requested, the values of U_g^1 are assumed to be zero.

5. The transformation from grid point temperatures to element temperatures is given by the matrix $[G_{ge}]$. This matrix is formed one column per element. The nonzero values in the column are the values G_{gi} ; the row numbers are $\{SIL\}_i$, where i is the column number.
6. The following output matrices are calculated by the matrix routines where

$$[Q_{ge}] = [G_{ge}][R^e], \quad (7)$$

$$[R_{gg}] = [Q_{ge}][G_{ge}]^T. \quad (8)$$

The list of elements, ER, is written on the lender record of the Q_{ge} file. The third output matrix may be calculated from the equation

$$[K_{gg}] = [K_{gg}^X] + [R_{gg}][S_{gg}]. \quad (9)$$

MODULE FUNCTIONAL DESCRIPTIONS

4.104.8 Subroutines

Utility subroutines DECØMP, GFBS, SSG2B, and TRANP1 are used for matrix manipulations. Subroutine HBDY is used for processing the HBDY elements. See the descriptions for these routines in Section 3.

4.104.9 Design Requirements

RMG requires six scratch files. Open core for RMG is defined at /RMGZZZ/. Core limitations are as follows:

1. The triangular form of the symmetric input matrix F (on RADMTX cards) must fit in core in packed form.
2. Eleven words (or 15 words if double precision is used) per HBDY element in RADLST data list must fit in core during the R^e matrix generation phase. Working core must also be available for DECØMP and GFBS during this phase.
3. In the output matrix processing phase, all of core is available to perform the matrix transpose and the MPYAD operations.

4.104.10 Diagnostic Messages

Messages 3071 thru 3079 may be generated.

FUNCTIONAL MODULE SSGHT (STATIC SOLUTION GENERATOR - HEAT TRANSFER)

4.105 FUNCTIONAL MODULE SSGHT (STATIC SOLUTION GENERATOR - HEAT TRANSFER)

4.105.1 Entry Point: SSGHT

4.105.2 Purpose

This module iterates to obtain the solution to the steady state nonlinear heat transfer problem. Radiation matrices and temperature dependent conductivities are allowed. The matrix operations are done in either single or double precision, depending on the type of matrices; the vector results and the input loads are single precision. The correction terms for nonlinear conductivities have internal double precision operations.

4.105.3 DMAP Calling Sequence

SSGHT HUSET,HSIL,GPTT,GM,HSET,MPT,DIT,HPF,HPS,HKFF,HKFS,HKSF,HKSS,HRPN,HRSN,HLLL,HULL /
 HUGV,HQG,HRULV / V,N,NLK / V,N,NLR / C,Y,EPSHT / C,Y,TABS / C,Y,MAXIT / C,Y,IRES /
 V,N,MPCF1 / V,N,SINGLE \$

4.105.4 Input Data Blocks

HUSET	Sets identification table for each degree of freedom
HSIL	Scalar Index Table
GPTT	Element and Grid Point temperature table. The second set of records contains the estimated temperature distribution.
GM	Multipoint constraint transformation matrix
HSET	Element Summary Table
MPT	Material Property Table
DIT	Direct INput Tables
HPF,HPS	Applied load vector partitions
HKFF,HKFS HKSF,HKSS	Partitions of linear heat transfer matrix
HRFN,HRSN	Radiation matrix partitions
HLLL,HULL	Decomposition products of matrix KFF

Note: USER, GPTT, KFF, LLL, and ULL may not be purged.

MODULE FUNCTIONAL DESCRIPTIONS

4.105.5 Output Data Blocks

UGV Vector of temperatures for all points
QG Forces of constraint vector (heat flow into boundary points)
RULV Residual load vector (units of heat flow)

Notes: UGV may not be purged; QG must be purged if no U_g points exist.

4.105.6 Parameters

NLK Input, integer. No nonlinear conductivity exists if value is -1.
NLR Input, integer. No radiation exists if value is -1.
EPSHT Input, real. Accuracy test parameter, default is .001.
TABS Input, real. Absolute temperature of user system, default is 0.0.
MAXIT Input, integer. Iteration limit, default = 4.
IRES Input, integer. Controls output of RULV. If value = 1, output occurs. If value = -1, no output occurs. Default = -1.
MPCF1 Input, integer. Indicates U_m points do not exist if value = -1. Default = 0.
SINGLE Input, integer. Indicates U_s points do not exist if value = -1. Default = 0.

4.105.7 Method

The module may be subdivided into three logical sections. The first section reads the input data, initializes the data, and generates convenience tables for use in the iteration loop. The second section is a long iteration loop which generates nonlinear loads, solves the solution equation, and tests for convergence. The third section expands the resulting solution vector and calculates forces of constraint.

4.105.7.1 Initialization Phase

1. The parameters NLK and NLR are tested to determine the existence of nonlinear conductivity and radiation, respectively.
2. The USET data block is read and tables are constructed to identify the various degrees of freedom. The U_g locations of the U_m points are placed in the MP0INT table. The U_g locations of the U_n points are placed in the NP0INT table. If the U_n point is also a U_f point, its value is set negative.

3. A table of length U_g is constructed which contains the U_n location for each U_m point. This is done by reading the GM matrix where each column is a U_n point and the location of each nonzero term in the column corresponds to an equivalent U_m point. A warning message is printed if a U_m point is associated with more than one U_n point.
4. The ninth word of /SYSTEM/ contains the user requested temperature set. The GPTT data block is searched to find the requested grid point temperature set (the second set of records). The temperature values are placed in core in the form of a vector, U_n^1 . The U_m values of temperature are ignored.
5. The linearized load vector is calculated by the equation

$$\{P_f^1\} = \{P_f\} - [K_{fs}] \{U_s^1\},$$

where $\{U_s^1\}$ is the vector of temperatures corresponding to the U_s points.

6. If nonlinear conduction exists, the EST data block is processed by subroutine SSGHT1 to produce a table (on scratch file 1) of element properties. The SIL values for each element are replaced by their U_n locations, the initial material properties are calculated using the U_n^1 temperature vector, and the data format is simplified.
 7. If radiation exists, a diagonal matrix, $[S_n]$, is placed in core. The equation for each term is
- $$S_{ni} = 4(U_{ni}^1 + T_0)^3,$$
- where T_0 = the value of parameter TABS.
8. A partitioning vector, UN, is built in core to define where each U_n point belongs in the U_s or U_f vectors.

4.105.7.2 Iteration Section

In this part of the subroutine, the solution to the nonlinear equation is extracted by an iteration procedure. The following data is stored in core:

1. The vector U_n^i , the last solution.
2. Material and tabular data selected by the HMAT subroutine.
3. UN - The partitioning table of length U_n .
4. S_n - A diagonal matrix of length U_n .

MODULE FUNCTIONAL DESCRIPTIONS

5. ΔU_n - The difference vector between subsequent solutions.
6. ΔP_n - The incremental load due to nonlinear effects.

These locations also may be used to store temporary data whenever they are available. The data stored on files for use in this process are:

1. P_f^1 - The linearized load.
2. RFN - The radiation matrix.
3. LLL and ULL - The decomposition products of KFF.
4. KFF - The linear heat transfer matrix.

Each iteration executes the following steps:

1. If nonlinear conduction exists, a vector ΔP_n is generated by reading the reduced element data on scratch file 1 with subroutine SSGHTP. This routine produces a difference load, ΔP_e , for each element and adds it to ΔP_n . The basic equation is:

$$\Delta P_e = [K(U_n^1) - K_1] \{U_n^1\},$$

where $[K(U_n^1)]$ is the element conductivity matrix when the material is evaluated for temperature set $\{U_n^1\}$, $[K_1]$ is the element conductivity matrix evaluated at the estimated temperature $\{U_n^1\}$, and $\{U_n^1\}$ is the current temperature vector.

This step is skipped on the first pass, since the results will be zero. The $\{\Delta P_n\}$ vector, if it exists, is partitioned into the vectors $\{\Delta P_f\}$ and $\{\Delta P_s\}$.

2. If radiation exists, the following equation is executed:

$$\{N_f^1\} = \{\Delta P_f\} + [R_{fn}] \{(U_n^1 + T_0)^4 - S_n U_n^1\},$$

where T_0 is the value of the parameter TABS. The matrix product is added to the ΔP_f vector, such that the N_f^1 terms replace the ΔP_f terms in core.

3. The linear load vector, P_f^1 , is read from its file and used in the equation:

$$\{\bar{P}_f\} = \{P_f^1\} - \{N_f^1\}.$$

The \bar{P}_f vector result replaces the N_f^1 terms in core. It is then copied in matrix format onto scratch file 2.

4. The loading error vector, δP_f , is now calculated by the equation:

$$\delta P_f = P_f - [K_{ff}] U_f^i.$$

U_n^i is partitioned at this stage into vectors U_f^i and U_s^i .

The error ratio for the loads is:

$$\epsilon_p = \frac{|\{\delta P_f\}|^2}{|\{P_f^1\}|^2}$$

If the RULV output residual vector is requested (IRES \neq -1), the $\{\delta P_f\}$ vector is appended to the RULV file.

5. Subroutine GFBS is called to solve for a new solution vector using the equation:

$$[LLL] [ULL] \{U_f^{i+1}\} = \{P_f\}.$$

6. The U_f^{i+1} vector is moved into the ΔP_n space in core and, except on the first pass, the following parameters are calculated:

$$\alpha = \{\bar{P}_f\}^T \{U_f^{i+1}\}$$

$$\beta = \{\bar{P}_f\}^T \{\delta u\}$$

$$\gamma = \{\bar{P}_f\}^T \{U_f^{i+1} - U_f^i\}$$

If $\gamma = 0$, λ_1 is set to 100.0 and ϵ_t is set to 0. Otherwise:

$$\lambda_1 = |\beta/\gamma|.$$

If $\alpha = 0$ or $\lambda_1 = 1$, ϵ_t is set to 100.0. Otherwise:

$$\epsilon_t = \left| \frac{\gamma}{(\lambda_1 - 1)\alpha} \right|$$

7. If any of the following tests pass, the loop is terminated.

- $\epsilon_t < \text{EPSHT}$ and $\epsilon_p < 10 \text{ EPSHT}$ and $\lambda_1 > 1$: Convergence is achieved.
- $i \geq \text{MAXIT}$: Maximum iteration limit is reached.
- $\lambda \leq 1$ and $i \geq 4$: Diverging solution.
- $\tau < 0$: Insufficient time.
- $\gamma = 0$: Maximum Convergence.

Note: i is the loop count and

$$\tau = 1 - \frac{\text{TIME} + \text{TIME FOR NEXT LOOP}}{0.8 (\text{TIME LIMIT})}$$

8. If the loop is to continue, the following equations are executed:

$$\{\delta u_f\} = \{U_f^{i+1}\} - \{U_f^i\}$$

$$\left\{ \begin{array}{c} U_f^{i+1} \\ -U_s \end{array} \right\} \{U_n^i\}$$

4.105.7.3 Output Data Generation

In this section, the force-of-constraint vector, $\{q_s\}$, is calculated and expanded to U_g size as the output data block $\{q_g\}$. The equation is:

$$\begin{aligned} \{q_s\} = & \{\Delta P_s\} - \{P_s^1\} + [K_{sf}] \{U_f^i\} + [K_{ss}] \{U_s^i\} \\ & + [R_{sn}] \{ (U_f^{i+1} + T_0)^* - S_n U_n^{i+1} \}. \end{aligned}$$

The solution vector $\{U_g\}$ is formed by merging the displacement vectors $\{U_n^{i+1}\}$ and $\{U_m\}$, where each value of U_m is the value of the equivalent U_n point.

4.105.8 Subroutines

Utility subroutine GFBS is used; see Section 3 for details.

4.105.8.1 Subroutine Name: SSGHTP

1. Entry Point: SSGHTP
2. Purpose: To partition a vector, U , into two vectors, U_a and U_b , where U exists in core and the resulting vectors will occupy the same space upon return.
3. Calling Sequence: CALL SSGHTP (PV,U,SIZE)

PV - Partitioning vector. The value of each location gives the location the corresponding term is to be moved to.

U - Vector to be partitioned.

SIZE - Length of PV and U.

4.105.8.2 Subroutine Name: SSGHT1

1. Entry Point: SSGHT1
2. Purpose: To process the EST (Element Summary Table) for three purposes:

FUNCTIONAL MODULE SSGHT (STATIC SOLUTION GENERATOR - HEAT TRANSFER)

- a) The data for all elements are arranged into a uniform format and the material conductivity properties for the reference temperature are calculated. If the material is not temperature dependent, the element calculations are skipped.
- b) The SIL values of the connected grid points are replaced by their equivalent indices in the U_n vector.
- c) If the element conductivity is temperature dependent, the data is written on a given file.

3. Calling Sequence: CALL SSGHT1 (IEST,FILE,NEQUIV)

IEST - File number of EST file must be opened and positioned at record 1. Input.

FILE - File number of output data file, must be opened and trailer written. Input.

NEQUIV - Array of integers of length U_g , each value is the equivalent U_n position. Input.

4.105.8.3 Subroutine Name: SSGHT2

1. Entry Point: SSGHT2

2. Purpose: To calculate heat flow loads due to a difference in temperature for elements with temperature dependent conductivity. See Section 8 of the NASTRAN Theoretical Manual for the equations. In general, the conductivity matrix for an element can be expressed by the equation

$$[K_e] = Vol[C]^T [K_m(\bar{T})] [C],$$

where $[C]$ is the transformation matrix between grid point temperatures and temperature gradients, $[K_m(\bar{T})]$ is the conductivity coefficient matrix evaluated at temperature \bar{T} , Vol is the volume of the element. The heat flow load for the element is

$$\{\Delta P_e\} = Vol[C]^T [K_m(\bar{T}_e^i) - K_m(\bar{T}_1)] [C] \{T_e^i\},$$

where $\{T_e^i\}$ is the current temperature vector for the connected points, \bar{T}_1 is the initial estimate of temperature, \bar{T}_1 is the average of the temperatures of the connected points, and $\{\Delta P_e\}$ is the difference heat flow for the particular element.

3. Calling Sequence: CALL SSGHT2 (FILE,DELTA,UNI)

FILE - File number of element data calculated by subroutine SSGHT1.

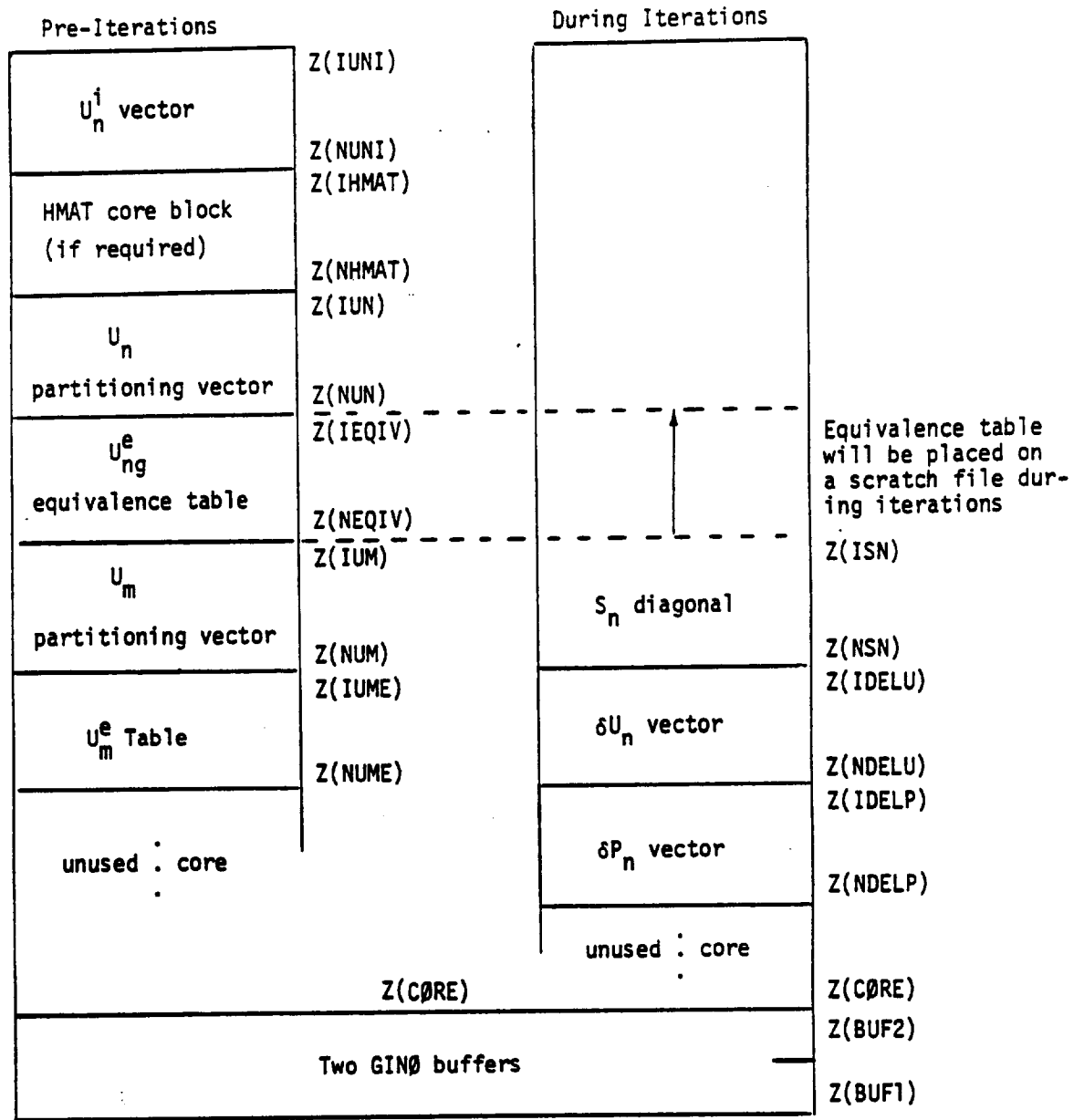
DELTA - Array in core of $\{\Delta P\}$ vector.

UNI - Array in core of $\{T^i\}$ vector.

MODULE FUNCTIONAL DESCRIPTIONS

4.105.9 Design Requirements

Open core is required for this module as shown below.



4.105.10 Diagnostic Messages

Messages 3081 thru 3087 may be issued.

FUNCTIONAL MODULE TRLG (TRANSIENT LOAD GENERATOR)

4.106 FUNCTIONAL MODULE TRLG (TRANSIENT LOAD GENERATOR)

4.106.1 Entry Point: TRLG

4.106.2 Purpose

To generate applied loads in a transient analysis. A combination of static and dynamic loading cards are used to generate load matrices for use in the transient solution and for output data calculations. The form of the output matrices is one column per time step.

4.106.3 DMAP Calling Sequence

TRLG {CASECC}{CASEXX}{HUSED}{HSETD}{DLT}{HDLT}{SLT}{HSLT}{BGPDT}{HSIL}{SIL}{CSTM}{HTRL}{TRL}{DIT}{GMD}{HGDD}{GDD}{PHIDH},
 {HEST}{EST},MGG/{HPP}{PPT}{HPS}{PST}{HPDD}{PDT}{HPD}{PD},PH,{HTOL}{TOL}/V,N,NØSET/V,N,PDEPDØ/V,N,NCØL \$

4.106.4 Input Data Blocks

CASEXX CASECC	}	- Case control data containing load set request and time step request
HUSED HSETD	}	- Set definition table, dynamics
DLT HDLT	}	- Dynamic Load Table
SLT HSLT	}	- Static Load Table
BGPDT		- Basic Grid Point Definition Table
SIL HSIL	}	- Scalar Index List
CSTM		- Coordinate System Table
TRL HTRL	}	- Transient Response List
DIT		- Direct Input Tables
GMD		- Multipoint constraint transformation matrix
GDD HGDD	}	- Structural partitioning transformation matrix
PHIDH		- Eigenvector transformation matrix
HEST EST	}	- Element Summary Table

MODULE FUNCTIONAL DESCRIPTIONS

MGG - Mass matrix - g set

- Notes:
1. CASECC, USETD, DLT, and TRL must be present.
 2. GMD must be present if m set is not null.
 3. GØD must be present if 0 set is not null.
 4. MGG must be present if gravity loads are requested.

4.106.5 Output Data Blocks

PPT or HPPØ - Matrix of output load vectors for all points.
PST or HPSØ - Matrix of output load vectors for U_s points.
PDT or HPDD - Matrix of output load vectors for analysis points.
PD, HPDT or PH - Matrix of load vectors on analysis points for all time steps.
HTØL or TØL - Table of output times.

- Notes:
1. PD may not be purged.
 2. PSØ may be purged only if s set is null.
 3. PPØ and PDØ may be purged.

4.106.6 Parameters

NØSET - Output-integer-default = -1. If NØSET = -1, d set equals p set.
PDEPDØ - Output-integer-default = -1. If PDEPDØ = -1, output time steps are all time steps.
NCØL - Input-integer-default = 0. If NCØL = 0, the initial time in the table of output times (TØL) is set to 0.0. If NCØL > 0, the initial time is set to the specified output time of the previously checkpointed run.

4.106.7 Method

The TRLG module consists of four major subroutines with a small module driver subroutine. The subroutines are described below in the order of their execution.

4.106.7.1 TRLGA

The purpose of this subroutine is to select the applied load functions and build a matrix defining the load factors for each point and each function of time. The steps followed are:

1. As preliminary processing, the CASEXX data is read; the load set Id and initial condition Id are found. The Static Load Table (SLT) is processed by subroutine SSGSLT to convert any heat transfer loads on elements to loads on grid points. The output of SSGSLT is on file NEWSLT.

FUNCTIONAL MODULE TRLG (TRANSIENT LOAD GENERATOR)

2. The Dynamic Loads Table (DLT) is processed to determine the requested load sets, their scale factors, and their location in the DLT data. A list is built of the requested static load sets.
3. The load factor vectors from the static load cards are built with the aid of subroutine EXTERN (it is also used in Module SSG1). A list is built of the requested simple loads which exist in the NEWSLT data, a scratch file is set up for the QVECT data, and a call EXTERN is made. The input files which are used are the NEWSLT, BGPDT, CSTM, SIL, and EST data blocks. Gravity loads are computed, but temperature or enforced deformation loads are not calculated. The output files are a set of load vectors in matrix format on file PG and the QVECT data.
4. The USETD data block is read and the extra points (U_e) are identified. A table is built to convert SIL to SILD values. The following steps are processed for each requested TLØAD record on the DLT data block. Two files are generated, the AP file contains the constant load factors and TMLDTB contains the function-of-time data.
5. Two d-set size vectors are set to zero for A (load factor) and T (time delay) data, which may be different for each point. The static load factor vector requested by the TLØAD card is found. The position (SIL) of each nonzero term in the PG vector is converted to a U_d (SILD) position and the term is placed in the A array.
6. The dynamic area factor data (DAREA) and delay data (DELAY) in the DLT for the TLØAD card are added to the A and T arrays. The points with zero A values are discarded from both lists and the lists are compressed to a table of triple values: the scalar index (position in the vector), the A value, and the T value.
7. The triple table is sorted on the values of T. Blocks of A values with unique T values are output as individual columns in the AP matrix. The position of each term is its scalar index. For each unique value of T, an entry is written on the TMLDTB file. Its contents are:
 1. Entry number
 2. TLØAD Id
 3. Type of TLØAD
 4. T
 - 5-10. TLØAD data (six words)
 11. QVECT pointer

MODULE FUNCTIONAL DESCRIPTIONS

If simple loads are being processed, the QVECT pointer is zero. If QVECT loads are being processed, this code has been entered from Step 3 and returns to the appropriate section of code.

8. If any QVECT data for the current TLØAD set has been requested, it has been written on the QVECT data file, identified by its static load Id. Each entry in the QVECT data has the format:

	1. Load set Id	
	2. Number of connected points, N	
	3. SIL ₁	Static scalar index, point 1
	:	
	A ₁	Area factor for point 1
	SIL ₂	Etc.
2N+3 Etc.	E ₁ , E ₂ , E ₃	Integer table numbers or real values defining the source vector
2N+7 Etc.	V ₁₁ , V ₁₂ , V ₁₃	Normal vector
2N+10 Etc.	V ₂₁ , V ₂₂ , V ₂₃ 2 ^d	Normal vector for elliptic cylinder

For each entry, the SIL values are converted to SILD values and the T value for each point is found to form a triple table as in Step 6. The last nine words of the QVECT data are written on scratch file 3, the QVECT pointer is set and Step 7 is repeated, and a return is made to process the next QVECT entry.

4.106.7.2 TRLGB

This module transforms the load factor matrix AP, which contains all degrees of freedom, into solution sized matrices. The various subroutines used in the SSG2 module are also used here. The basic equations are:

1. If multipoint constraints are used, the AP matrix is partitioned and combined as follows:

$$[A_p] \longrightarrow \begin{bmatrix} \bar{A}_{nd} \\ -A_m \end{bmatrix}$$

$$[A_{nd}] = [\bar{A}_{nd}] + [G_{md}]^T [A_m]$$

FUNCTIONAL MODULE TRLG (TRANSIENT LOAD GENERATOR)

2. If single point constraints are present, the matrix is partitioned:

$$[A_{nd}] \rightarrow \begin{bmatrix} A_{fd} \\ -A_s \end{bmatrix}$$

3. If structural partitioning is used, the $[A_{fd}]$ matrix is partitioned and combined as follows:

$$[A_{fd}] \rightarrow \begin{bmatrix} \bar{A}_d \\ -A_o \end{bmatrix}$$

$$[A_d] = [\bar{A}_d] + [G_{od}]^T [A_o]$$

4. If a modal formulation is the solution method, the A_d matrix is transformed by the equation:

$$[A_h] = [\phi_{dh}]^T [A_d]$$

4.106.7.3 TRLGC

This subroutine generates the functions of time associated with each column of the AP load factor matrix. The inputs are the TMLDTB file generated in Step 7 of 4.106.7.2, the time step data on data block TRL, the QVECT vector data on scratch file 3, and the DIT (Direct Input Tables) data block. The outputs are the [FS] matrix and the [FT] matrix, with each column corresponding to a time and each row associated with a unique time function, and the TØL output data block a list of the output times. The process is described below.

1. The TMLDTB data, the QVECT data, and the selected time step data (TSTEP card images) are read into open core. A list of the referenced tables is built from the TLØAD1 and QVECT data. The list of tables is used by utility subroutine PRETAB to bring the tabular functions of time into core.
2. A loop is performed over all requested time steps. The data on the TSTEP card are:

$$\Delta t_1, Nt_1, No_1$$

$$\Delta t_2, Nt_2, No_2$$

Etc.

$$\Delta t_n, N_{tn}, N_{on}$$

MODULE FUNCTIONAL DESCRIPTIONS

The solution times are calculated as follows:

$$t_i = t_{-1} + \Delta t_1 \text{ if } i \leq N_{t1},$$

$$\text{or } t_i = t_{-1} + \Delta t_1 N_{t1} + t_2(i - N_{t1}) \text{ if } i - N_{t1} \leq N_{t2},$$

$$\text{or } t_i = t_{-1} + \Delta t_1 N_{t1} + \Delta t_2 N_{t2} + \Delta t_3(i - N_{t2})$$

$$\text{if } i - (N_{t1} + N_{t2}) \leq N_{t3}$$

etc., where $i = 0, 1, 2, \dots, \sum N_{tn}$ and $t_{-1} = 0.0$ if $NC\emptyset L = 0$ and

t_{-1} = specified output time of the previously checkpointed run if $NC\emptyset L > 0$.

The output times are calculated in a similar fashion, except that a time increment of $N_{oi}\Delta t_i$ is used and the points where the step size changes are the same.

- For each solution time, a load is to be calculated. All functions are evaluated at each value of time. For load type = 1 (TLØAD1 data), the algorithm is:

$$F_{ji} = T_j(t_i - \tau_j),$$

where T_j is the tabular function and τ_j is the time lag referenced in the j^{th} entry of the TMLDTB data. For load type = 2 (TLØAD2 data), the algorithm is:

$$F_{ji} = x^n e^{dx} \cos [2\pi f x + \phi]$$

$$\text{if } T_{1j} < t_i - \tau_j < T_{2j}.$$

$$\text{or } F_{ji} = 0 \text{ if } t_i - \tau_j < T_1 \text{ or } t_i - \tau_j > T_2, \text{ and } x = t_i - T_1 - \tau_j$$

where $\tau, T_1, T_2, f, \phi, n$, and α are given in the j^{th} entry of the TMLDTB data.

- If the QVECT reference in the TMLDTB data is nonzero, the load function is modified by the factor $R(t)$ as follows:

$$a) \quad \text{Calculate } \{q\} = \begin{Bmatrix} E_1(t_i - \tau_j) \\ E_2(t_i - \tau_j) \\ E_3(t_i - \tau_j) \end{Bmatrix}.$$

where E are either table Id numbers or constant real numbers given in the referenced QVECT data.

- b) If $\{V_2\} \neq 0$ (elliptic cylinder case),

$$R(t) = \sqrt{(\{q\}^T \{V_1\})^2 + (\{q\} \{V_2\})^2}$$

- c) If $\{V_2\} = 0$ (nonelliptic cylinder case),

$$R(t) = 0, \text{ if } \{q\}^T \{V_1\} \geq 0,$$

$$\text{or } R(t) = -\{q\}^T \{V_1\} \text{ if } \{q\}^T \{V_1\} < 0.$$

The function $R(t) \cdot F_{ji}$ replaces the previously calculated value of F_{ji} .

5. If the current time step is an output time step (and the set of output times is different than the solution times), then the F_{ji} column is output on matrix file FØ. The column is always output on matrix file FT.

4.107.7.4 TRLGD

This subroutine generates the output load matrices PPT, PST, PDT, PD, and/or, for modal formulation, PH. The equations are simple matrix multiples as given below:

$$[P_{pt}] = [A_p] [F_t]$$

$$[P_{st}] = [A_s] [F_t]$$

$$[P_{dt}] = [A_d] [F_t]$$

$$[P_d] = [A_d] [F_s]$$

$$\text{and } [P_h] = [A_h] [F_s]$$

The following exceptions exist:

1. If all time steps are output times, $[F_s]$ is used instead of $[F_t]$, and $[P_d]$ is assumed equal to $[P_{dt}]$.
2. If any output file is purged, the corresponding matrix is not calculated.
3. If the d set equals the p set, $[P_{dt}]$ is assumed to be the same as $[P_{pt}]$.

4.106.8 Subroutines

This module uses the subroutines of modules SSG1 and SSG2 quite heavily. See Sections 4.41 and 4.42 for descriptions of subroutines SSGSLT, EXTERN, SSG2A, and SSG2B.

MODULE FUNCTIONAL DESCRIPTIONS

4.106.8.1 Subroutine Name - TRLGA

1. Entry Point: TRLGA
2. Purpose: To select transient load sets and build the area factor matrix AP and the time function tables TMLDTB.
3. Calling Sequence: CALL TRLGA (CASECC, USETD, DLT, SLT, BGPDT, SIL, CSTM, AP, TMLDTB, TRL, ISCR1, ISCR2, ISCR3, EST, NEWSLT, MGG, ISCR4)

With one exception, all inputs are GINØ file numbers; CASEXX, USETD, DLT, SLT, BGPDT, SIL, CSTM, TRL, EST and MGG are standard NASTRAN data blocks input to the subroutine. The others are:

AP	Output matrix of constant load factors
TMLDTB	Time function load table
ITRL	Transient response set Id (not a GINØ file).
ISCR1 ISCR2 ISCR3 ISCR4	Scratch file numbers for internal use
COMMON/SSGAIX/Z(1) - Open core for TRLGA and its subroutines	
COMMON/LØADX/ - EXTERN parameters; see Section 4.41.11.8	

4.106.8.2 Subroutine Name - TRLGB

1. Entry Point: TRLGB
2. Purpose: To reduce the columns of the AP matrix to solution size, and produce the AS matrix.
3. Calling Sequence: CALL TRLGB (USETD, AP, GMD, GØD, PHIDH, AS, AD, AH, IFLAG1, SCR1, SCR2, SCR3, SCR4)

With one exception, all inputs are GINØ file numbers; USETD, GMD, GØD, and PHIDH are standard NASTRAN data block file numbers. The others are:

AP	Input, matrix of constant load factors
AS	Output, matrix of factors on s-set points
AD	Output, matrix of load factors on d-set points
AH	Output, matrix of load factors on h-set points (Modal formulation only)
IFLAG1	Output, integer which is set to -1 if d set equals p set.
SCR1 SCR2 SCR3 SCR4	Scratch file numbers for internal use

FUNCTIONAL MODULE TRLG (TRANSIENT GENERATOR)

4.106.8.3 Subroutine Name - TRLGC

1. Entry Point: TRLGC
2. Purpose: To evaluate the functions of time at each solution value of time, and if necessary, to output the function values at each output time step.
3. Calling Sequence: CALL TRLGC (TMLDTB,TRL,DIT,ITRL,FCT,FCØ,TØL,IFLAG)

TMLDTB	Input, GINØ file number of the table of functions of time; each entry contains the contents of a TLØAD data card, a unique time lag, and a QVECT reference. The QVECT data is also appended to this table.
TRL	GINØ file number of the Transient Response List data block.
DIT	GINØ file number of the Direct Input Tables data block.
ITRL	Id number of the requested TRL data.
FCT	Output, GINØ file number of the matrix of time functions evaluated at all time steps, one column per value of time.
FCØ	Output, GINØ file number of the matrix of time functions evaluated at output time steps.
TØL	Output, GINØ file number of the table of output times.
IFLAG	Output, integer value of -1 if output times equal all times.

4.106.8.4 Subroutine Name - TRLGD

1. Entry Point: TRLGD
2. Purpose: To combine the various load factor matrices and the time function matrices to produce load versus time matrices.

MODULE FUNCTIONAL DESCRIPTIONS

3. Calling Sequence: CALL TRLGD (FCT,FCØ,AP,AS,AD,AH,PPT,PST,PDT,PD,PH,IFLAG1,SCR1,IFLAG)

Except for IFLAG and IFLAG1, all inputs are GINØ file numbers. The inputs are:

FCT	Function of time matrix, all time steps.
FCØ	Function of time matrix, output time steps.
AP	Load factor matrix, all points.
AS	Load factor matrix, s-set points.
AD	Load factor matrix, d-set points.
AH	Load factor matrix, h-set points (modal formulation only.)
IFLAG1	Integer, if value = -1, AD = AP and AS.
SCR1	Scratch file for internal use.
IFLAG	Integer, if value = -1, FCØ = FCT.

The outputs are:

PPT	Load versus time matrix, each column is a complete load vector at a specific output time.
PST	Load versus output time matrix, s-set points.
PDT	Load versus output time matrix, d-set points.
PD	Load versus all times matrix, d-set points.
PH	Load versus all times matrix, h-set points.

FUNCTIONAL MODULE TRLG (TRANSIENT LOAD GENERATOR)

4.106.9 Design Requirements

Nine scratch files are needed. Open core is defined as follows:

1. TRLGA allocates core for subroutines SSGSLT and EXTERN in its initial processing phase, see Section 4.41.12 for these descriptions. In the final processing phase, the core is allocated as follows for each dynamic load set:

A_p Vector (LUSET Size)
τ_p Vector (LUSET Size)
SIL, A, τ Compressed List
: : (Open :
SIL - SILD Table
Load Id's and Factors
4 Buffers

2. TRLGB uses COMMON/SSGA2/ as open core only as space for the SSG2A and SSG2B subroutines.

MODULE FUNCTIONAL DESCRIPTIONS

3. TRLGC open core is /TRLG1C/. The data space is allocated as follows:

Time Function Data (12 words/function)
QVECT Data (9 words/QVECT entry)
Time Step Data (3 words/card)
Table List (1 word/requested DIT table)
DIT Table Data (requested tables)
Function Values (1 word/function)
. : (Open) .
3 Buffers

4. TRLGD does not use open core. Its subroutines allocate their own core.

4.106.10 Diagnostic Messages

Messages 3001, 3002, 3003, 3008, 3031 and 3056 may be issued.

FUNCTIONAL MODULE TRHT (TRANSIENT RESPONSE, HEAT TRANSFER)

4.107 FUNCTIONAL MODULE TRHT (TRANSIENT RESPONSE, HEAT TRANSFER)

4.107.1 Entry Point: TRHT

4.107.2 Purpose

To calculate the transient solution of a heat transfer problem involving conduction capacity convection, and radiation. The user inputs are the initial conditions, heat flow input, time steps, and integration parameters. The outputs are temperatures, time change of temperatures, and nonlinear heat flows.

4.107.3 DMAP Calling Sequence

TRHT CASEXX,HUSETD,HNLFT,DIT,GPTT,HKDD,HBDD,HRDD,HPDT,HTRL/HUDVT,HPNLD/V,N,NLR /
V,Y,RADLIN \$

4.107.4 Input Data Blocks

CASECC	Case Control Table
HUSETD	Set Definition Table, dynamics
HNLFT	Nonlinear Function Table
DIT	Direct Input Tables
GPTT	Grid Point Temperature Tables - contains initial conditions
HKDD	Linear heat transfer matrix
HBDD	Capacity matrix
HRDD	Radiation matrix
HPDT	Applied heat flow load-versus-time matrix
HTRL	Transient Response List - contains time step data.

Note: Various files may be purged if the appropriate case control request is missing.

4.107.5 Output Data Blocks

HUDVT	Temperature and velocity vectors arranged in a matrix as two columns per time step. A null column for each time step is also output.
HPNLD	Nonlinear load vector matrix, one column per time step.

Note: HUDVT may not be purged.

MODULE FUNCTIONAL DESCRIPTIONS

4.107.6 Parameters

BETA Input, real, default = 0.55. Integration stability-accuracy control parameter.

TABS Input, real, default = 0.0. Absolute temperature of 0 degrees.

NLR Input, integer, default = -1. If value = -1, no radiation exists.

RADLIN Input, integer, default = -1. If value = +1, radiation effects are linearized.

4.107.7 Method

The module is subdivided into three major subroutines. The module itself is simply a driver routine which sets up files and calls the subroutines. The algorithms used are described as follows.

4.107.7.1 Initialization

Subroutine TRHT1A is used to set up the problem. The time step data is extracted from data block TRL, placed in the bottom of core, and written on scratch file 5. The data for each logical set is an open-ended card in groups of three numbers: N_t - number of steps, Δt - time increment, N_o - output increment.

The requested initial condition vector is built as follows: The USET data block is read into core and a table is built where the location of each g-set point is given the value of its location in the d-set vector. If a point is constrained, it is given a value of zero. The GPTT data block is opened and the requested initial condition set of temperatures are found. The format of the GPTT for this type of problem has a set of records defining the element temperatures followed by the same temperature sets in the format of grid point temperatures. The point number is used to locate the corresponding position in the u_d vector, and a vector is built in core. If the temperature for all points is defined, the vector is written on scratch file 5. If an estimated temperature is defined in case control ($TEMP(MAT) = n$), the process is repeated for this temperature set and the resultant vector is added to scratch file 5.

4.107.7.2 Matrix Processing

Subroutine TRHT1B processes the [K] and [B] matrices initially and for each time step change. If symmetric matrices are input, the symmetric decomposition routine FACTØR is used instead of FACTRV. The basic equations used are:

1. Add:

$$\frac{1}{\Delta t} [B] + \beta [K] \rightarrow [F].$$

2. Decompose:

$$[F] \rightarrow [L] [U].$$

3. Form the matrix:

$$\frac{1}{\Delta t} [B] - (1-\beta) [K] \rightarrow [A].$$

The outputs are [L], [U], and [A].

4.107.7.3 Integration

For each group of time steps, the TRHTC subroutine is used to integrate the equations. The basic operations performed in this routine are described as follows:

1. Core is allocated for four vectors and if nonlinear effects exist, space is allocated for three more vectors. The vectors are:

U_1 - The previous solution vector.

U_2 - The present solution vector.

P_1 - The previous load vector.

P_2 - The present load vector.

U_k - The estimated temperature vector.

N_1 - The previous nonlinear load.

N_2 - The present nonlinear load.

2. The initial solution vector is read into the U_1 space and if radiation exists, the estimated temperature vector is read into the U_k space. The output files are opened and a loop is performed over all time steps in a triple entry: $N_t, \Delta t, N_0$. The following steps describe the loop.

3. The nonlinear effects are calculated as follows:

If radiation exists and RADLIN = -1, the following equations are used:

- a) For each term, i , in the vector,

$$U_{ri} = -(U_{1i} + T_0)^4 + 4(U_{ki} + T_0)^3 U_{1i},$$

where T_0 is the value of parameter TABS.

MODULE FUNCTIONAL DESCRIPTIONS

b) Multiply:

$$\{N_r\} = [R_{dd}] \{U_r\}$$

If RADLIM $\neq -1$ and radiation exists,

$$\{N_r\} = [R_{dd}] \{-(U_{ki} + T_o)^4 + 4(U_{ki} + T_o)^3 U_{ki}\}.$$

This vector is a constant load, calculated in the first time step and stored in position N_2 .

If nonlinear functions are defined in data block TRL, subroutine TRDID is used to calculate a load $\{N_e\}$. This load is added to the $\{N_r\}$ vector and stored in position N_2 . On the first time step, the N_2 vector is also stored in position N_1 .

4. The load vectors P_1 and N_1 are calculated in a special manner on the first step of each group. On the first time step of the overall problem:

$$\{P_1\} = [K_{dd}] \{U_1\} - \{N_2\}$$

$$\{N_1\} = \{0\}.$$

On the first step after a time step change from Δt_1 to Δt , the load is:

$$\{P_1\} = [K + \frac{1}{\Delta t_1} B] \{U_1\} - (\frac{1}{\Delta t_1}) [B] \{U_{-1}\} - \{N_2\}.$$

$$\{N_1\} = (1 - \frac{\Delta t}{\Delta t_1}) \{N_2\} + (\frac{\Delta t}{\Delta t_1}) \{\bar{N}_1\},$$

where $\{\bar{N}_1\}$ and $\{U_{-1}\}$ are the previous value of $\{N_1\}$ and $\{U_1\}$ stored on a scratch file.

5. The load vectors are added and the next displacement vector is calculated as follows:

Multiply and add:

$$\{\Delta P\} = [A] \{U_1\} + (1-\beta) \{P_1\} + \beta(P_2) + (1-\beta) \{N_2\} - \{N_1\}$$

Solve for $\{U_2\}$:

$$[L] [U] \{U_2\} = \{\Delta P\}.$$

FUNCTIONAL MODULE TRHT (TRANSIENT RESPONSE, HEAT TRANSFER)

6. If output data is requested, the $\{U_1\}$ vector is written on the UDVT file followed by the velocity vector $\{u\}$ and a null acceleration vector. The velocity vector is:

$$\{u\} = \frac{1}{\Delta t}(\{U_2\} - \{U_1\}) .$$

If nonlinear loads exist, the vector $\{N_2\}$ is output on file NLFT.

7. The loop is continued until all time steps for each Δt set is finished. The pointers to data in core are switched such that

$$P_2 \rightarrow P_1$$

$$N_2 \rightarrow N_1$$

$$U_2 \rightarrow U_1$$

If the loop is finished, the time step data and the vectors $\{U_k\}$, $\{U_2\}$, $\{U_1\}$, $\{N_2\}$, and $\{N_1\}$ are written on a scratch file.

4.107.8 Subroutines

The main subroutines used by the module are TRHT1A, TRHT1B, and TRHT1C. These subroutines also call subroutines described elsewhere in the Programmer's Manual. They are:

SSG2C	A matrix addition routine described in Section 3.5.11.
FACTØR	Symmetric decomposition utility subroutine described in Section 3.5.23.
FACTRU	Unsymmetric decomposition version of FACTØR. Uses DECØMP, Section 3.5.15.
MATVEC	Multiplies a matrix times a vector in core and adds results to core. See Section 4.65.8.6.
TRDID	Computes nonlinear loads at each time step. See Section 4.65.8.9.
INTFBS	Unsymmetric forward-backward solution of a vector in core. Result is stored in core. See Section 4.65.8.8.
FBSINT	Symmetric version of INTFBS.

4.107.8.1 Subroutine Name: TRHT1A

1. Entry Point: TRHT1A
2. Purpose: To initialize the transient heat transfer solution.

MODULE FUNCTIONAL DESCRIPTIONS

3. Calling Sequence: CALL TRHT1A (CASEXX, USETD, GPTT, TRL, NGRØUP)

CASEXX GINØ file number of case control data block
USETD GINØ file number of set descriptor data block.
GPTT GINØ file number of grid point temperature table data blocks.
TRL GINØ file number of transient response list; contains time step data.
NGRØUP Number of groups of time step data.

4.107.8.2 Subroutine Name: TRHT1B

1. Entry Point: TRHT1B

2. Purpose: To generate the matrices used in the integration of the heat transfer solution.

3. Calling Sequence: CALL TRHT1B (IØF, DELTA)

IØF Not used.
DELTA New time increment, Δt .
/(blank)/ Beta, TABS, etc.
/TRHTX/ IK(7), IB(7), ICR1,...ICR5, ISYM, ICR6, ICR7, where:
 IK Matrix control block for K matrix.
 IB Matrix control block for B matrix.
 ICRi Scratch file numbers.
 ISYM Symmetric flag.

4.107.8.3 Subroutine Name: TRHT1C

1. Entry Point: TRHT1C

2. Purpose: To perform the integration of a transient heat transfer problem. This routine is re-entered for each time step change.

3. Calling Sequence: CALL TRHT1C (NGRØUP, UDVT, PD, RDD, ILØØP)

NGRØUP Number of groups of triple time step data.
UDVT GINØ file number of output solution vectors.
PD GINØ file number of input load vector matrix.
RDD GINØ file number of radiation matrix.
ILØØP Index of current time step group.

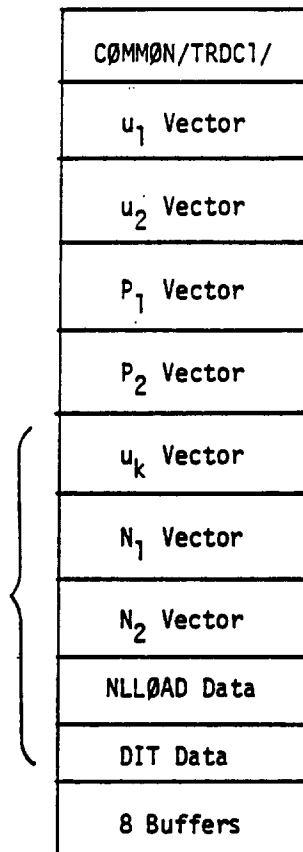
FUNCTIONAL MODULE TRHT (TRANSIENT RESPONSE, HEAT TRANSFER)

/TRDIX/ See above (Section 4.107.8.2).
/TRDC1/ Open core.
/(blank)/ BETA,TABS,NØRAD,RADLIN
/TRDD1/ Local variable storage space.

4.107.9 Design Requirements

Open core for subroutine TRHT1C is illustrated as follows:

Used only for
nonlinear loads
as required.



The requirements for the other subroutines are less critical.

4.107.10 Diagnostic Messages

Messages 3001, 3002, 3005, 3008, 3031 and 3045 may be issued.

FUNCTIONAL MODULE SDRHT - STRESS DATA RECOVERY, HEAT TRANSFER

4.108 FUNCTIONAL MODULE SDRHT - STRESS DATA RECOVERY, HEAT TRANSFER

4.108.1 Entry Point: SDRHT

4.108.2 Purpose

This module combines the heat flow data for the HBDY element and modifies the ØEF1 output data block. The sources of data are 1) the convective heat flux due to temperature difference across a boundary layer, 2) the radiation heat exchange, and 3) the applied heat flow due to external sources.

4.108.3 DMAP Calling Sequence

SDRHT HSIL,HUSET,HUGV,HØEF1,HSLT,HEST,DIT,HQGE,DLT / HØEFIX / V,Y,TABS \$

4.108.4 Input Data Blocks

HSIL	Scalar Index List
HUSET	Set definition table for each degree of freedom
HUGV	Temperature vector matrix. In transient, also contains velocity vectors.
HØEF1	Output file of element heat flow data
HSLT	Static Loads Table
HEST	Element Summary Table
DIT	Direct Input Tables
HQGE	Radiation heat flow versus temperature matrix
DLT	Dynamic Loads Table

4.108.5 Output Data Blocks

HØEFIX Same format as ØEF1, except HBDY elements have additional data.

4.108.6 Parameters

TABS Input, real number, absolute temperature value of user system.

4.108.7 Method

4.108.7.1 Element Data Calculations

The HBDY element data on the ØEF1 input file contains the elements selected by the user for output. The convective heat flow, F_h , into the element is also given. For each element, a 14-word entry in a core-held table is built. The contents are:

1.	Id	Element Id
2.	F_r	Radiation flow
3.	F_h	Convection flow
4.	F_a	Applied flow
5.	TYPE	HBDY type
6.	A	Element area (length if type = 6)
7.	α	Absorbtivity coefficient
8-10.	$\{V_1\}$	Normal vectors
11-13.	$\{V_2\}$	
14.	IDO	Output identification number

Items Id, F_h , and IDO are given by the ØEF1 file.

On the first solution vector, the EST data block is read, and for each HBDY element included in the output list, the items A, α , $\{V_1\}$, and $\{V_2\}$ are calculated using subroutine HBDY.

4.108.7.2 Radiation Heat Flow Calculations

The header record of the QGE file contains a list of HBDY elements used for radiation. The subsequent data contains a matrix column for each of the elements. The radiation heat flow, F_{ri} , into each element is:

$$F_{ri} = \{Q_{gei}\}^T \{(u + Tabs)^4\}, \text{ where}$$

$\{Q_{gei}\}$ is the column of QGE corresponding to element i.

$\{u\}$ is the solution vector from the UGV file.

$\{(u + Tabs)^4\}$ is the vector of absolute temperatures to the fourth power.

4.108.7.3 Applied Heat Flow Calculations-Statics

In statics, the applied loads are calculated in a manner similar to the calculations of module SMA1. On the first pass, a table is built for each possible load set request containing:

Position	
i	Master set Id ₁
i+1	Number of subload Id's -n
.	Subload - Id ₁
.	Factor - S ₁
.	Record number in SLT data
	:
	:
	:
i+3u+1	Factor u
	(Repeated for each LOAD card image.)
	:
	:
	:
	:
	:
k	Load set Id
	u = 1
k+3	Load set Id
	S ₁ = 1.0
	Record number

For each solution subcase, the applied load set Id is found in the above table. The applied loads are calculated and added to the F_a position in the element table with the following algorithms:

The SLT data contains QBDY1, QBDY2, and QVECT data card images. For each requested load set, these cards are found and if the element is included in the element able, the load is calculated as follows:

1. QBDY1 cards:

$$\Delta F_a = A Q_o S$$

MODULE FUNCTIONAL DESCRIPTIONS

2. QBDY2 cards:

$$\Delta F_a = AS(\sum_{i=1}^M Q_i)/M,$$

where M is the number of points on the element.

3. QVECT cards (TYPE \neq 6):

$$C = \{V_1\}^T \{E\},$$

$$\Delta F_a = -AQ_0 C, C < 0,$$

$$\text{or } \Delta F_a = 0, C \geq 0.$$

4. QVECT cards (TYPE = 6):

$$\Delta F_a = AQ_0 (\{V_1\}^T \{E\})^2 + (\{V_2\} \{E\})^2$$

4.108.7.4 Applied Heat Flow Calculations (Transient)

The transient applied load terms are calculated with the same code as the static terms except for the following operations.

The load set table is constructed from the DLT data block. The requested load set number is used to find a DLØAD or TLØAD data card image on the DLT header record. A table is built with an 11-word entry for each requested TLØAD set number. The contents are:

TLØAD Id
Factor - S(t)
Static load set
TLØAD type
TLØAD factor, C_f , (from DLØAD card)
TLØAD time function data (6 words)

The static load table is used to determine the static load set record. The TLØAD Id is replaced with the static set Id and the static load set Id is replaced with the static load set record number.

FUNCTIONAL MODULE SDRHT - STRESS DATA RECOVERY, HEAT TRANSFER

The TLØAD time function data contains:

<u>Word</u>	<u>TLØAD1</u>	<u>TLØAD2</u>
6	Id_t - (Table of $F(t)$)	T_1 Begin time
		T_2 End time
		f Frequency
		ϕ Phase angle
		n Exponent
		α Growth coefficient

A list is built of the referenced tables and subroutine PRETAB is called to read the DIT data block. The referenced tables are stored in core.

For each solution at a particular value of time, t , the factor, S , on the applied loads is a function of time. The equations are:

TLØAD1:

$$S(t) = C_f F(t) \text{ (Table evaluated using TAB.)}$$

TLØAD2:

$$S(t) = C_f \Delta t^n e^{\alpha \Delta t} \cos(2\pi f \Delta t + \phi)$$

$$\text{if } \Delta t > 0 \text{ and } t < T_2,$$

$$\text{or } S(t) = 0 \text{ if } \Delta t < 0 \text{ or } t > T_2,$$

$$\text{where } \Delta t = t - T_1.$$

After the time factors are calculated, the static loads are processed as in the statics case. An exception occurs when a QVECT data card references a tabular function of time. If any of the values E_i of the flux vector on the QVECT card are integers, their value is replaced by the value of the referenced table evaluated at $t = \text{time}$.

4.108.7.5 Output Format

After each solution vector has been processed, the following data is output on the ØEFIX file for each element:

$$ID0, F_n, F_r, F_a, (F_n + F_r + F_a).$$

MODULE FUNCTIONAL DESCRIPTIONS

Heat flux data for other element types are simply copied across from the ØEF1 data block to the ØEFIX data block.

4.108.8 Subroutines

Utility subroutine HBDY is used to process the HBDY element data. See the description in Section 3.

4.108.9 Design Requirements

SDRHT requires no scratch files. Its open core requirements are:

Statics

Element Table 14 Words/Element
Ug Vector (Unpacked)
Static Load Set Table 5 Words/Simple Load Id 5 x n words/LØAD
Open
3 Buffers

Dynamics

Element Table 14 Words/Element
Ug Vector (Unpacked)
Static Load Set Table
Dynamic Load Set Table 11 Words/TLØAD Card
Direct Input Tables
Open
3 Buffers

4.108.10 Diagnostic Messages

Messages 3063 thru 3069 may be issued.

FUNCTIONAL MODULE GPCYC (GEOMETRY PROCESSOR FOR CYCLIC PROBLEMS)

4.109 FUNCTIONAL MODULE GPCYC (GEOMETRY PROCESSOR FOR CYCLIC PROBLEMS)

4.109.1 Entry Point: GPCYC

4.109.2 Purpose

This module processes the user supplied information about degrees of freedom which are to be constrained between the segments for cyclic symmetry problems. This is a preprocessor for the CYCT2 module. It identifies the constraints between the degrees of freedom in the analysis sets for the cosine (symmetric) and sine (antisymmetric) models. It is placed outside the loop (i.e., solution for values of the cyclic index k) in static analysis.

4.109.3 DMAP Calling Sequence

GPCYC GEOM4,EQEXIN,uset / CYCD / V,Y,CTYPE / V,N,NØGØ \$

4.109.4 Input Data Blocks

GEOM4	Contains CYJØIN data cards for cyclic constraints
EQEXIN	Equivalence of external to internal indices
uset	Displacement set definitions table

4.109.5 Output Data Block

CYCD	Cyclic component constraint data
------	----------------------------------

4.109.6 Parameters

CTYPE	Input-BCD-RØT for rotational, DIH, DRL, or DSA for dihedral - no default.
NØGØ	Output-Integer-+1 unless an error has occurred, in which case it will be -1 - no default.

4.109.7 Method

The output data block CYCD will be a code, having one entry for each point in the u_a set of the NASTRAN model. The code is different for rotational (CTYPE = Rxxx) and dihedral (CTYPE = Dxxx) problems. The type is put into the data block trailer, along with the length of u_a set.

MODULE FUNCTIONAL DESCRIPTIONS

For rotational, use 1 and for dihedral use 2 for coding into the trailer.

Rotational - The CYCD for the n^{th} degree of freedom in the u_a set is the n^{th} word of CYCD data block. Its value is

0 if n is interior
 m if n is on side 1
 $-m$ if n is on side 2

where m is the number of the connected degree of freedom. The following method can be used. SIDE 1 and SIDE 2 lists are specified on CYJØIN cards on GEØM4. If none ore more than one of each, a fatal message occurs.

a. Put the SIDE 1 list and the second record of EQEXIN in core. The second EQEXIN record has $10 \times \text{SIL} + \text{code}$ (code = 1 for a grid and 2 for a scalar point). Read a SIDE 2 entry and the corresponding entry from SIDE 1. Put on a scratch file a five word list containing:

Code		
Internal Index	}	SIDE 1
Grid Number		
Internal Index	}	SIDE 2
Grid Number		

Continue for all SIDE 2 entries. A fatal error occurs (the NØGGØ parameter is set to -1) if the code does not match for a pair, if the two points are the same, or if the lists are of different length. Grid numbers will be carried along to give (to the user) meaningful diagnostic messages.

b. Next put USET in core. Read each mask, starting from the top, and replace the masks of the points in the u_a set by -1, -2, -3, ..., $-\text{SIL}_a$ (the number of points found is the length of the u_a set). Each entry from step "a" will produce 6 (code = 1) or 1 (code = 2) possible pairs of SIL numbers to be joined. Pass through the possible SIL pairs to see if both are in the u_a set. If neither is in u_a set, ignore the pair and continue. If one is in u_a set and the other is not, output a warning message that no connection will be made (identify grid point, component and set of the two points), and continue. If both are in the u_a set, than a pair to be joined has been found. Output on a scratch file a pair of SIL_a numbers (i.e., the positive value of the index in the u_a set, along with the grid point identification number to be used for diagnostics:

[SIL _a Index	}	SIDE 1
	Grid Number		
	SIL _a Index	}	SIDE 2
	Grid Number		

It should be noted that it is legal to have no pairs. A warning message is output if a grid point produced no pairs.

c. Clear an area in core the length of u_a set for the CYCD. For each pair in the list formed in step (b), check that the CYCD in the locations given by the two SIL_a numbers are zero. If they are, put the value of SIL_a Side 2 into the location SIL_a Side 1. Then put (-1) times the value of SIL_a Side 1 into the location SIL_a Side 2. If non-zero values were found, a fatal message identifying the grid point listed more than once is produced. The resulting list is the CYCD data block for the rotational case.

Dihedral - The value of CYCD for the n^{th} degree of freedom in the u_a set is

- 0 if interior
- 1 if on Side 1 and an even component
- 2 if on Side 1 and an odd component
- 3 if on Side 2 and an even component
- 4 if on Side 2 and an odd component

The following method can be used. SIDE 1 and SIDE 2 lists are on GEOM4. If at least one of each is not found a fatal message occurs. It is legal to have many cards of either type.

a. Put the second record of EQEXIN in core. Output on a scratch file a five word list (for every entry on either SIDE 1 or SIDE 2)

[Side
	Coord.Sys.(blank → 0, R, C → 1 and S → 2)
	Code = 1,2 (grid or scalar)
	Internal Index
	Grid Number

MODULE FUNCTIONAL DESCRIPTIONS

b. Next put USET into core, and replace u_a set masks with $-SIL_a$ numbers as for rotational. Each entry from step (a) will produce 6 or 1 (Code = 1 or 2) possible boundary points. Only points in u_a set are to be joined. Pass through the entries from step "a" to see which produce SIL values in the u_a set. If none (from a grid point) are found, produce a warning message identifying the grid point. If points in u_a set are found, output a three word record on a scratch file containing

```
[ CYCD
   $SIL_a$  Index
  Grid Point
```

CYCD is the output code defined above. The component is "even" if Code = 2 or (Code = 1) and Coord Sys = 1, Components 1, 3, 5) or (Code = 1 and Coord Sys = 2, Components 1, 2, 6). Otherwise it is "odd". If Code = 1 and Coord Sys = 0, a fatal message occurs.

c. Clear an area in core the length of u_a set for the CYCD. For each item in the list formed in step "b", check that the location SIL_a has zero value. If it does not, then produce a fatal message identifying the grid point which has been listed twice. If the location was zero, put the CYCD in the location. Continue until done. The result is the CYCD data block for the dihedral case.

4.109.8 Subroutines

Utility subroutines BISRCH and PRELOC are used.

4.109.9 Design Requirements

Two scratch files are required. Open core required at /GPCYCX/

4.109.10 Diagnostic Messages

Message numbers 4024, 4025, 4026, 4027, 4028, 4029, 4030, 4032, 4037 and 4039 may occur.

FUNCTIONAL MODULE CYCT1 (TRANSFORMATION TO CYCLIC COMPONENTS - PHASE I)

4.110 FUNCTIONAL MODULE CYCT1 (TRANSFORMATION TO CYCLIC COMPONENTS - PHASE I)

4.110.1 Entry Point: CYCT1

4.110.2 Purpose

This module is used to transform loads and displacements between physical and symmetric components representations. The module is capable of transforming in either direction. It is capable of transforming for either cyclic or dihedral symmetry. In the dihedral case, the "physical" representation can be resolved in terms of left and right halves, or in terms of symmetric and antisymmetric components for the model's N segments. The module is capable of handling multiple loading conditions, and a restricted range of the cyclic parameter K. Both input and output vectors are "analysis-size" (u_a) vectors.

4.110.3 DMAP Calling-Sequence

```
CYCT1  {PL } / {PX } {GCYCF } / V,Y,CTYPE / C,N,FØRE / V,Y,N / V,Y,KMAX / V,Y,NLØAD /  
      {UXV } {ULV } {GCYCB }  
      V,N,NØGØ $
```

4.110.4 Input Data Blocks

PL - Load vector matrix giving static loads on & set.

UXV - Displacement vector matrix giving displacement on symmetric components.

4.110.5 Output Data Blocks

PX - Loads on the symmetric components.

ULV - Displacements on the physical components.

GCYCF - Transformation Matrix (may not be purged)

GCYCB - Transformation Matrix for displacements on physical components (may not be purged).

4.110.6 Parameters

CTYPE Input-VCD-no default-code specifying transformation type. RØT for rotational, DRL for dihedral with right or left, or DSA for dihedral with symmetric and antisymmetric.

CDIR Input-integer-no default-number of segments.

N Input-integer-no default-number of segments.

MODULE FUNCTIONAL DESCRIPTIONS

KMAX Input-integer-no default-maximum value of cyclic parameter K if positive. If negative, the only value of K is /KMAX/.

NLOAD Input-integer-default = 1 - number of loading conditions.

NØGØ Output-integer-no default - +1 unless an error has occurred, in which case it will be -1.

4.110.7 Method

The computations in this module should not be attempted if (a) the second output data block is purged, or (b) the input and first output data blocks are not purged and the input matrix is incompatible with the parameters. The number of columns in the input matrix must be the same as the number of rows of GCYC, which is:

CDIR CTYPE	FØRE	BACK
RØT	N·NLØAD	FACT·NLØAD
DRL or DSA	2·N·NLØAD	2·FACT·NLØAD

where FACT is given by

$$FACT = \begin{cases} 2 \cdot KMAX + 1 & 2 \cdot KMAX < N \\ N & 2 \cdot KMAX = N \end{cases}$$

If $2 \cdot KMAX > N$, or $KMAX < 0$, or $N \leq 0$, a fatal message should be given. "Fatal" messages will set the NOGO parameter, and then return.

First compute $C_n = \cos(2\pi n/N)$ and $S_n = \sin(2\pi n/N)$ for $n = 0, 1, \dots, (N-1)$. The method will be to use a recursion relationship. First compute $C_0 = 1.0$, $S_0 = 0.0$, $C_1 = \cos(2\pi/N)$ and $S_1 = \sin(2\pi/N)$. Then

$$\left. \begin{aligned} C_n &= C_1 C_{n-1} - S_1 S_{n-1} \\ C_{N-n} &= C_N \\ S_n &= S_1 C_{n-1} + C_1 S_{n-1} \\ S_{N-n} &= -S_N \end{aligned} \right\} n \leq N/2$$

FUNCTIONAL MODULE CYCT1 (TRANSFORMATION TO CYCLIC COMPONENTS - PHASE 1)

If N is even, set $C_{N/2} = -1$, $S_{N/2} = 0$. Once C_n and S_n are known for $n \leq (N/2)$, the values should be calculated for the rest of the values $n \leq (N-1)$ from

$$C_{N-n} = C_n$$

$$S_{N-n} = -S_n$$

Once the values are tabulated for $0 \leq n \leq (N-1)$, C_n and S_n can be found for arbitrary integer index since

$$C_n = C_{(n \bmod N)}$$

$$S_n = S_{(n \bmod N)}$$

The next step is to compute the "compact" transformation matrices. This is best shown by examining Figure 1. There are two types called FØRE and BACK, which are quite similar. Since it will be used to post-multiply another matrix for the transformation, it must be assembled by columns. The following comments help to define the matrices:

Type FØRE

1. There is a scalar multiplier of $2/N$.
2. There are N rows.
3. The first column, called $K = 0$, consists of $1/2$'s.
4. Additional columns come in pairs. The first pair is called $K = 1$, the next pair $K = 2$, etc. Each pair consists of a cosine column and a sine column. Going down the column the index n of C_n and S_n is incremented by K .
5. If N is odd, the last pair is $K = (N-1)/2$. If N is even, there is an additional column $K = N/2$, which consists of alternating $1/2$'s and $-1/2$'s.
6. Compute only the columns $K < KMAX$. The dotted line in Figure 3 corresponds to the boundary for $KMAX = 2$.

Type BACK

1. This is like the transpose of FØRE, except there is no scalar multiplier, nor factor of $1/2$ in the first and last rows.
2. The parameter $KMAX$ will limit the length of the columns. There will always be N columns.

MODULE FUNCTIONAL DESCRIPTIONS

The "compact" transformation matrix is only valid for CTYPE = RØT and NLØAD = 1. If CTYPE = DRL or DSA or NLØAD > 1, the matrix will have to be expanded. If (and only if) CTYPE = DRL, the matrix must be expanded as follows. This will involve making two rows and two columns for each element. The pattern can be seen below:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \rightarrow \begin{bmatrix} a & a & b & b \\ a & -a & b & -b \\ c & c & d & d \\ c & -c & d & -d \end{bmatrix}$$

The basic pattern for expansion is the same for FØRE or BACK. A scalar multiplier of 1/2 is to be included for type FØRE.

If CTYPE = DSA, double the value of NLØADS. If NLØADS > 1, a further expansion is required. (Note that if CTYPE = DRL and NLØAD > 1, both types of expansion will be applied.) The basic pattern can be seen below:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \rightarrow \begin{bmatrix} a & 0 & 0 & b & 0 & 0 \\ 0 & a & 0 & 0 & b & 0 \\ 0 & 0 & a & 0 & 0 & b \\ c & 0 & 0 & d & 0 & 0 \\ 0 & c & 0 & 0 & d & 0 \\ 0 & 0 & c & 0 & 0 & d \end{bmatrix}$$

The example shown is for NLØAD = 3. For the general case, each row (or column) will be expanded into NLØAD rows and columns.

The transformation matrix is now complete. If the second output data block is not purged, the transformation matrix should be there.

If the input and first output data blocks are not purged, then

$$[PX] = [PL] [GCYCF] ,$$

where GCYCF is the transformation matrix.

4.110.8 Subroutines

Subroutine SSG2B is called.

4.110.9 Design Requirements

One scratch file is required. The numerical calculations for GCYCF are done in double-precision. The matrix is produced according to the precision selected by the SYSTEM cell.

Open core is used at /CYCT1Z/.

4.110.10 Diagnostic Messages

The following messages may be issued by CYCT1:

3008, 4063, 4064, 4065, 4066 and 4067.

FUNCTIONAL MODULE CYCT1 (TRANSFORMATION TO CYCLIC COMPONENTS - PHASE I)

$$\frac{2}{N} \begin{bmatrix} \frac{1}{2} & C_0 & S_0 & C_0 & S_0 & C_0 & . & . & . & \frac{1}{2} \\ \frac{1}{2} & C_1 & S_1 & C_2 & S_2 & C_3 & . & . & . & -\frac{1}{2} \\ \frac{1}{2} & C_2 & S_2 & C_4 & S_4 & C_6 & . & . & . & \frac{1}{2} \\ \frac{1}{2} & C_3 & S_3 & C_6 & S_6 & C_9 & . & . & . & -\frac{1}{2} \\ \frac{1}{2} & C_4 & S_4 & C_8 & S_8 & . & . & . & . & . \\ \frac{1}{2} & C_5 & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . \\ \frac{1}{2} & C_{N-1} & S_{N-1} & . & . & . & . & . & . & . \end{bmatrix}$$

Type FØRE

$$\begin{bmatrix} 1 & 1 & 1 & 1 & . & . & . & 1 \\ C_0 & C_1 & C_2 & C_3 & . & . & . & C_{N-1} \\ S_0 & S_1 & S_2 & S_3 & . & . & . & S_{N-1} \\ C_0 & C_2 & C_4 & C_6 & . & . & . & . \\ S_0 & S_2 & S_4 & S_6 & . & . & . & . \\ \hline C_0 & C_3 & C_6 & C_9 & . & . & . & . \\ 1 & -1 & 1 & -1 & 1 & . & . & . \end{bmatrix}$$

Type BACK

Figure 1. Phase 1 - Transformation Matrices

FUNCTIONAL MODULE CYCT2 (TRANSFORMATION TO CYCLIC COMPONENTS - PHASE 2)

4.111 FUNCTIONAL MODULE CYCT2 (TRANSFORMATION TO CYCLIC COMPONENTS - PHASE 2)

4.111.1 Entry Point: CYCT2

4.111.2 Purpose

This module is used to transform cyclic problems to the solution set ("independent") variables, and transform the results back to the cyclic components. The module is general purpose, but can provide the following specific tasks:

- a. Transform a load vector and stiffness matrix to the solution set for static analysis.
- b. Transform mass and stiffness matrices to the solution set for eigenvalue analysis.
- c. Transform displacements back to the cyclic transform degrees of freedom for static analysis.
- d. Transform eigenvectors and eigenvalues back to the cyclic variables for eigenvalue analysis.

Static analysis is expected to be a "looping" rigid format, where the loads to be transformed in task (a) are extracted from a matrix of load vectors; also the resulting displacements are appended in task (c), to the appropriate locations.

4.111.3 DMAP Calling Sequence

Forward - Statics

```
CYCT2      CYCD,KAA,,PX,, / KKK,,PK,, / C,N,FØRE / V,Y,NSEGS=-1 / V,N,KINDEX /  
           V,Y,CYCSEQ=-1 / V,Y,NLØAD=1 / V,N,NØGØ $
```

Backward - Statics

```
CYCT2      CYCD,,,UKV,RUKV, / ,,UXV,RUXV, / C,N,BACK / V,Y,NSEGS / V,N,KINDEX /  
           V,Y,CYCSEQ / V,Y,NLØAD / V,N,NØGØ $
```

Forward - Eigenvalues

```
CYCT2      CYCD,KAA,MAA,,, / KKK,MKK,,, / C,N,FØRE / V,Y,NSEGS=-1 / V,Y,KINDEX=-1 /  
           V,Y,CYCSEQ=-1 / C,N,1 / V,N,NØGØ $
```

Backward - Eigenvalues

```
CYCT2      CYCD,,,PHIK,LAMK / ,,PHIA,LAMA / C,N,BACK / V,Y,NSEGS / V,Y,KINDEX /  
           V,Y,CYCSEQ / C,N,1 / V,N,NØGØ $
```

4.111.4 Input Data Blocks

CYCD	Cyclic Component Constraint Data
KAA	Matrix (Stiffness)
MAA	Matrix (Mass)
PX	Load on symmetric components
UKV	Displacements on solution set
RUKV	Residual vector on solution set
PHIK	Eigenvectors on solution set
LAMK	Eigenvalues for PHIK

4.111.5 Output Data Blocks

KKK	Transformed Matrix (Stiffness)
MKK	Transformed Matrix (Mass)
PK	Transformed loads on solution set
UXV	Displacements on symmetric components
RUXV	Residual vector on symmetric components
PHIA	Eigenvectors on analysis set
LAMA	Eigenvalues for PHIA

4.111.6 Parameters

CDIR	Input-BCD code-no default-FØRE or BACK
NSEGS	Input-integer-no default-number of segments
KINDEX	Input-integer-no default-cyclic index
CYCSEQ	Input-integer-default-alternate = -1 - an integer code for sequence
NLØAD	Input-integer-default = 1 - number of loading conditions
NØGØ	Output-integer-no default - +1 unless an error has occurred, in which case it will be -1.

4.111.7 Method

Using information from the CYCD data block, two (sometimes one) sparse transformation matrices are formed. These are the same for "FØRE" and "BACK", but will depend upon K, N, and CYCSEQ. The type (rotational or dihedral) is specified in the matrix trailer of the CYCD data block. A fatal

FUNCTIONAL MODULE CYCT2 (TRANSFORMATION TO CYCLIC COMPONENTS - PHASE 2)

error (NØGØ parameter = -1) occurs if $K > N/2$ or $K < 0$ or $N \leq 0$. The transformation matrices can be constructed column by column using information from the CYCD data block.

1. Transformation Matrices, Rotational. The general case is $0 < K < N/2$. There are two matrices to be produced, called G_C and G_S . The number of rows is equal to the u_a size, which should be in the trailer record of CYCD.

There will be two columns produced for every non-negative number of CYCD; one is the cosine column and the other is the sine column. If $CYCSEQ = +1$, put all cosine columns before the sine columns; if $CYCSEQ = -1$, alternate. Let $c = \cos(2\pi K/N)$ and $s = \sin(2\pi K/N)$. The nonzero numbers in G_C and G_S are:

- a. Interior points ($CYCD_n = 0$) put a 1 in the n^{th} row of G_C if a cosine column and a 1 in the n^{th} row of G_S if a sine column.
- b. Points on Side 1 ($CYCD_n = m > 0$) put a 1 in the n^{th} row and a c in the m^{th} row of G_C and a $-s$ in the m^{th} row of G_S if a cosine column. Put an s in the m^{th} row of G_C and a 1 in the n^{th} row and a c in the m^{th} row of G_S if a sine column.
- c. Points on Side 2 ($CYCD_n = m < 0$) there are no columns associated with points on Side 2.

The cases $K = 0$ and $2 \cdot K = N$ require special treatment. There is no G_S matrix, and there are only cosine columns. Note that if $K = 0$, $c = 1.0$, $s = 0.0$, and if $2 \cdot K = N$, $c = -1$, $s = 0$.

2. Transformation Matrices, Dihedral. The general case is $0 < K < N/2$. There are two matrices to be produced, called G_S and G_A . The number of rows is the same as in the rotational case. There will be two columns for each interior point ($CYCD = 0$) called the S column and the A column. There is one column for each side point ($CYCD > 0$), which will be called an S column. If $CYCSEQ = +1$, order so that all S columns come first; if $CYCSEQ = -1$, intermix the S and A columns. Let $c = \cos(\pi K/N)$ and $s = \sin(\pi K/N)$. The nonzero numbers in G_S and G_A are:

- a. Interior points ($CYCD_n = 0$) put a 1 in the n^{th} row of G_S if an s column, and a 1 in the n^{th} row of G_A if an a column.
- b. Edge Points. If $CYCD_n = 1$, put a c in the n^{th} row of G_S and an s in the n^{th} row of G_A . If $CYCD_n = 2$, put a $-s$ in the n^{th} row of G_S and a c in the n^{th} row of G_A . If $CYCD_n = 3$, put a 1 in the n^{th} row of G_S . If $CYCD_n = 4$, put a 1 in the n^{th} row of G_A .

Cases $K = 0$ and $2 \cdot K = N$ require no special treatment. Note that for $K = 0$, $C = 1.0$, $S = 0.0$, and for $2 \cdot K = N$, $C = 0.0$, $S = 1.0$.

3. Transformation of Matrices. There are two inputs and two outputs used for transforming matrices. This is done only if $CDIR = FØRE$, and the matrices are not purged. Let $G_1 = G_C$ and $G_2 = G_S$ ($RØT$); $G_1 = G_S$ and $G_2 = G_A$ (DIH). Then

$$K_{KK} = G_1^T K_{aa} G_1 + G_2^T K_{aa} G_2 .$$

A similar formula holds for M_{aa} , the other matrix.

4. Transformation of Loads and Displacements. The trick here is to locate the quantities to be transformed, and the location to put the results. The subcases are arranged according to type (rotational or dihedral), index K , and kind (cosine or sine, or symmetric or antisymmetric). Also, there is the complication of multiple loads. The equations can be written as:

CDIR CTYPE	FØRE	BACK
RØT	$P_K = G_C^T P_C + G_S^T P_S$	$u_C = G_C u_x$ $u_S = G_S u_x$
DIH	$P_{K1} = G_S^T P_{CS} + G_A^T P_{SA}$ $P_{K2} = -G_A^T P_{CA} + G_S^T P_{SS}$	$u_{CS} = G_S u_{x1}$ $u_{CA} = -G_A u_{x2}$ $u_{SS} = G_S u_{x2}$ $u_{SA} = G_A u_{x1}$

The vectors will have $NLØADS$ columns. The subscript K refers to the local indices. The locations of the P_C, \dots, P_{SA} in the P_x load vector can be determined. The u_C, \dots, u_{SA} displacements are put back into the corresponding columns of the output data block. Thus, when $CDIR = BACK$, the data block will be of type append. In the loads, put the x_1 vectors all before the x_2 vectors.

5. Transformation of eigenvectors. The calculation is the same here except for the rule as to where the vectors come from and go to. The columns are alternately assigned to P_c and P_s . For type RØT, BACK, an even number of columns will be produced, which are alternately u_c and u_s . Thus, eigenvectors put in terms of cosine and sine segments will appear in the output in consecutive subcases. For type RØT, there is an exception to the above rule when $2K = 0$ or N , in which case G_s does not exist, and there is a one-to-one correspondence between input and output. In type dihedral, it will be assumed that P_{K2} and u_{x2} terms vanish. The columns must be even in number, and are assigned alternately to the cS and sA models.

6. Transformation of eigenvalues. This is only done when CDIR = BACK. When there is a two-to-one ratio of eigenvectors, the same will be done to eigenvalues. This will occur except when CTYPE = RØT and $2K = 0$ or N .

4.111.8 Subroutines

Utility subroutine SSG2B is used.

4.111.8.1 Subroutine Name: CYCT2A

1. Entry Point: CYCT2A

2. Purpose: To evaluate the matrix equation

$$K_{KK} = G_1^T K_{aa} G_1 + G_2^T K_{aa} G_2$$

3. Calling Sequence: CALL CYCT2A

(KAA,KXX,G1,G2,SCR1,SCR2,SCR3), where KAA, KXX, G1, G2, SCR1, SCR2, SCR3 are GINØ file names for their appropriate data blocks.

4. Method: $[X] = [KAA] [G_1]$ is formed first, then $[Y] = [G_1]^T [X]$ is computed. If necessary, the second term is evaluated in a similar order.

4.111.8.2 Subroutine Name: CYCT2B

1. Entry Point: CYCT2B

2. Purpose: To copy columns from one matrix to another.

3. Calling Sequence: CALL CYCT2B

(INPUT,ØUTPUT,NCØL,IZ,MCB), where INPUT and ØUTPUT are the GINØ file names of the INPUT

MODULE FUNCTIONAL DESCRIPTIONS

matrix (previously opened) and the OUTPUT matrix (previously opened) NCOL is the number of columns to copy. IZ is open core to use for the copy. MCB is a properly initialized matrix control block for the output matrix. COMMON/UNPAKX/ITC,II,JJ,INCR,ITC and INCR must be properly set for UNPACK.

4.111.9 Design Requirements

CYCT2 requires six scratch files. Matrices are built in the precision specified by the system precision cell.

4.111.10 Diagnostic Messages

Messages 3001, 3002, 3003, 3007, and 3008 may be issued by this module.

FUNCTIONAL MODULE APD (AERODYNAMIC POOL DISTRIBUTOR)

4.112 FUNCTIONAL MODULE APD (AERODYNAMIC POOL DISTRIBUTOR)

4.112.1 Entry Point: APD

4.112.2 Purpose

The principal function of APD is to generate aerodynamic elements. New tables are assembled to account for the element coordinates. Bulk data cards which control the solution of aerodynamic problems are processed and assembled into various data blocks for convenience and efficiency in the solution of the aerodynamic problem.

4.112.3 DMAP Calling Sequence

APD EDT,EQDYN,ECT,BGPD,T,SILD,USED,CSTM,GPLD/EQAERØ,ECTA,BGPA,SILA,
USETA,SPLINE,AERØ,ACPT,FLIST,CSTMA,GPLA,SILGA/V,N,NK/V,N,NJ/
V,N,LUSETA/V,N,BØV \$

4.112.4 Input Data Blocks

EDT - Aerodynamic bulk data cards
EQDYN - Equivalence between external points and scalar index values - dynamics
ECT - Element connection table
BGPD,T - Basic grid point definition table
SILD - Scalar index list - dynamics
USED,T - Displacement set definitions table - dynamics
CSTM - Coordinate system transformation matrices
GPLD - Grid point list - dynamics

4.112.5 Output Data Blocks

EQAERØ - Equivalence between external points and scalar index values - aerodynamics.
ECTA - Element connection table - aerodynamics
BGPA - Basic grid point definition table - aerodynamics
SILA - Scalar index list - aerodynamics
USETA - Displacement set definition table - aerodynamics
SPLINE - Contains SET1, SET2, SPLINE, SPLINE1, SPLINE2 and SPLINE3 cards. External coordinate numbers are changed to internal coordinate numbers and referenced CAERØ cards are modified and appended to the referencing cards.
AERØ - Control information for control of aerodynamic matrix generation and flutter analysis.

MODULE FUNCTIONAL DESCRIPTIONS

ACPT - Information pertaining to each independent group of aerodynamic elements
FLIST - Contains AERØ, FLFACT and FLUTTER cards copied from EDT
CSTMA - Coordinate system transformation matrices - aerodynamics
GPLA - Grid point list - aerodynamics
SILGA - Scalar index list - Structural grid point and aerodynamic points

4.112.6 Parameters

NK - Output - integer - no default. Degrees of freedom in the u_k displacement set.
NJ - Output - integer - no default.

$$N_j = \sum N_b + \sum_{i=1}^{N_{sb}} f_i + \sum_{j=1}^{N_{ib}} g_j$$

where N_b = number of aero boxes

N_{sb} = number of slender bodies

N_{ib} = number of interference bodies

f_i = degrees of freedom for each slender body

g_j = degrees of freedom for each interference body

LUSETA - Output - integer - no default. Degrees of freedom in the PA displacement set.

BØV - Output - real - no default. Conversion factor from circular frequency to reduced frequency (k).

4.112.7 Method

4.112.7.1 General

Subroutine APD is the module driver. It allocates twelve buffers and positions files for the different methods to write on. Boxes are generated by algorithms determined by aerodynamic theories or methods such as Doublet Lattice, Piston, etc. Each method has a method driver subroutine. CØMMØN/APD1C/ is used to pass pointers and file names to the various method drivers. Coordinate system ID's are assigned starting with one larger than the biggest existing ID. SIL's are assigned for SILA at one larger than existing SIL. SIL's are assigned for SILGA at one larger than existing G-size SIL. External ID of boxes start with external ID of CAERØ card. Internal ID of boxes are assigned starting at one larger than G-size.

FUNCTIONAL MODULE APD (AERODYNAMIC POOL DISTRIBUTOR)

Before any method driver starts, subroutine APDR is used to read the AEFACT cards, all the CAERØ cards and all the PAERØ cards into main memory and set up pointers to each. Then, if any CAERØ cards exist for a method, its driver is called until all the CAERØ cards have been processed. APD then finishes writing its output files.

4.112.7.2 Generation of Data Blocks

EQAERØ is generated from SCR1. SCR1 has the external ID plus 10*SILD+TYPE. SCR1 is read into main memory with space left for a third entry. The third entry is built as an internal index (without extra points). This list is sorted and words one and three become Record 1 of EQAERØ. Record 2 is words 1 and 2 of the list.

MODULE FUNCTIONAL DESCRIPTIONS

THIS PAGE HAS BEEN LEFT BLANK INTENTIONALLY.

FUNCTIONAL MODULE APD (AERODYNAMIC POOL DISTRIBUTOR)

ECTA is a copy of ECT with aero box elements added by the method drivers. An ECTA entry for an aero box is external ID plus 5 internal aero grid point ID's describing the box. The fifth grid point is the center.

BGPA is a copy of BGPDT with aero grid points added by the method driver.

SILA is a copy of SILD with aero grid points added (D-set) plus a second record copied from SCR2 with the equivalent SIL without extra points.

USETA is a copy of USETD with the aero grid point masks added (6 per grid point) by the method driver.

SPLINE has input data cards modified for easy interpretation by module GI. Subroutine APD converts external grid point ID to internal plus attaches the referenced CAERØ card to the SPLINE1, SPLINE2 and SPLINE3 cards. In addition, a record is built with all the k-set points with three words per k-point. The format is External ID, Internal ID, and starting k column number associated with the grid point.

AERØ has the MKAERØ1 or MKAERØ2 cards.

ACPT has data for module AMG, one record for each macro aeroelastic element. The data is arranged for ease of handling by the method drivers of AMG.

FLIST has the AERØ, FLFACT and FLUTTER cards.

CSTMA is a copy of CSTM with the aero coordinate systems added by the method drivers. Coordinate ID starts at one larger than currently on CSTM.

GPLA is a copy of GPLD plus the aero grid points added by the method drivers. Box corners start at 1,000,000; box centers at CAERØ ID.

SILGA is the SIL's with G and aero points only (no extra points).

4.112.8 Subroutines Called

4.112.8.1 Subroutine Name: APD

1. Entry Point: APD
2. Purpose: Module driver for aeroelastic box generation
3. Calling Sequence: CALL APD

MODULE FUNCTIONAL DESCRIPTIONS

4.112.8.2 Subroutine Name: APDF

1. Entry Point: APDF
2. Purpose: To determine the chordwise and spanwise locations of the aerodynamic boxes.
3. Calling Sequence: FSJ1 = APDF(FST,1,NSPAN)

4.112.8.3 Subroutine Name: IAPD

1. Entry Point: IAPD
2. Purpose: To determine the internal coordinate numbers of the box coordinates for the ECTA table.
3. Calling Sequence: CIO = IAPD(I,J,NC,NCRDP)

4.112.8.4 Subroutine Name: EMSG

1. Entry Point: EMSG
2. Purpose: To write a standard NASTRAN error message.
3. Calling Sequence: CALL EMSG(NCHAR,MSNØ,ISYS,IWF,ITEXT)

NCHAR - number of characters of text in array ITEXT.

MSNØ - message number to be printed.

ISYS - 1 - User
2 - System

IWF - 1 - Warning
2 - Fatal

ITEXT - Array containing NCHAR characters of text

If NCHAR = 0, EMSG will return after printing the message number. If MSNØ < 0, EMSG will call MESSAGE with -61 for a fatal termination.

4. Method: EMSG computes the number of lines of text to be printed (using /SYSTEM/ for NCPW) card calls EJECT for page control. It then prints the message using a machine-dependent format.

4.112.8.5 Subroutine Name: APDR

1. Entry Point: APDR
2. Purpose: Read a card type into main memory.
3. Calling Sequence: CALL APDR(FILE,Z,CØRE,IN,ØUT,WR,BUF,TYPE)

FUNCTIONAL MODULE APD (AERODYNAMIC POOL DISTRIBUTOR)

FILE - GINØ name of file to read

Z - beginning of open core

CØRE - length of open core available from IN to end - on output decremented by WR.

IN - 0 if card not found; offset from Z where card is, if found - output

ØUT - last location used in Z buffer read; after read = ØUT + WR

WR - number of words read from file - output

BUF - offset from Z where LØCATE buffer is

TYPE - locate codes for card type to read

4. Method: WR and IN are set to zero. LØCATE is called to find card type. If found, the record is read into core at Z(ØUT+1); IN is set, WR is set; ØUT is set; and core is decremented. EOF on Read or insufficient core give fatal error messages.

4.112.8.6 Subroutine Name: APDØE

1. Entry Point: APDØE

2. Purpose: Find an open-ended card

3. Calling Sequence: CALL APDØE(ID,Z,START,END,FØUND,CØUNT)

ID - ID of card wanted

Z - Start of open core

START - offset from Z where first card is located

END - offset from Z where end of cards is located

FØUND - offset from Z where start of card is; 0 if card ID not found - output

CØUNT - number of data items on card not counting the ID

4.112.8.7 Subroutine Name: APDCS

1. Entry Point: APDCS

2. Purpose: Build panel coordinate system

3. Calling Sequence: CALL APDCS

4. Method: All communication to APDCS is through common blocks /APD1C/ and /APD1D/ .

Given points 1 and 4 of a panel, these points are converted to basic from the CSTM entry for the coordinate system they are located in. Points 1 thru 4 are then located in the AERØ coordinate system. Then points 2 and 3 are located in basic. From here a panel coordinate system is defined and the CSTM entry is written. For Bodies, the program returns after finding point 1 in the AERØ system.

MODULE FUNCTIONAL DESCRIPTIONS

4.112.8.8 Subroutine Name: APD12

1. Entry Point: APD12
2. Purpose: Process CAERØ1 and CAERØ2 cards
3. Calling Sequence: CALL APD12
4. Method: A list of CAERØ1 cards and a list of CAERØ2 cards is built in core and sorted by IGID. The CAERØ1 cards are then processed in IGID sorted order. Subroutine APD1 is called for each card. When cards with the same IGID are exhausted, APD1 is flagged to put out the ACPT record; unless this IGID is the same as a CAERØ2 card. If it is the same, APD2 is called to combine the two types of cards. When all CAERØ1 cards have been processed, APD2 is called to process any remaining CAERØ2 cards.

4.112.8.9 Subroutine Name: APD1

1. Entry Point: APD1
2. Purpose: Process CAERØ1 cards
3. Calling Sequence: CALL APD1(FST,NS,FCT,NC,LS,LC)
FST - span any from AEFACT card
NS - number of spans
FCT - chord any from AEFACT card
NC - number of chords
LS - start of IGID flag
LC - end of IGID flag
4. Method: All of the card type processors have the following tasks in common:
 - a. Call APDCS to build a CSTM entry,
 - b. Write a BGPA, GPLA, USETA, SILA, SCR2 and SCR1 entry for all grid points generated,
 - c. Write an ECTA entry for all boxes (elements) generated, and
 - d. Write an ECPT entry when necessary.

For CAERØ1 cards, boxes are created in the chordwise direction then the spanwise direction. Box corners are put in the PSA set and box centers have components 3.5 in the UK set. Scratch files 303, 304, and 305 are used to hold data for the ACPT table from each CAERØ1 card until all CAERØ1 cards with the same IGID are processed. Then the ACPT record is written.

FUNCTIONAL MODULE APD (AERODYNAMIC POOL DISTRIBUTOR)

4.112.8.10 Subroutine Name: APD2

1. Entry Point: APD2
2. Purpose: Process CAERØ2 cards
3. Calling Sequence: CALL APD2(IØPT,CAØ1,CAØ2,NCØRE,ID)
IØPT - option flag 0 (CAERØ2 only); 1 (CAERØ2 with CAERØ1)
CAØ1 - start of CAERØ1 sorted list
CAØ2 - start of CAERØ2 sorted list
NCØRE - offset to first location of open core available
ID - if option = 1; IGID of CAERØ1 to attach to.
4. Method: APD2 performs steps similar to APD1. All CAERØ2 cards with the same IGID are processed together to build an ACPT record. If IØPT is 1, only the CAERØ2 with IGID's equal to ID are processed. If IØPT is 0, all remaining CAERØ2 cards are processed. Slender body centers have components in the UK set depending on body type: 3 and 5 for Z; 2, 3, 5, 6 for Z-Y; and 2, 6 for Y.

4.112.8.11 Subroutine Name: APD3

1. Entry Point: APD3
2. Purpose: Process CAERØ3 cards
3. Calling Sequence: CALL APD3
4. Method: APD3 performs the same general steps as APD1 except that an ACPT record is written for each CAERØ3 card. The control points (not box centers) only have component 3 in the u_k set.

4.112.8.12 Subroutine Name: APD4

1. Entry Point: APD4
2. Purpose: Process CAERØ4 cards
3. Calling Sequence: CALL APD4
4. Method: APD4 performs the same steps as APD3 except control points have components 3 and 5 in the u_k set; and component 6 if the control point is on a control surface.

FUNCTIONAL MODULE APD (AERODYNAMIC POOL DISTRIBUTOR)

4.112.8.13 Subroutine Name: APD5

1. Entry Point: APD5
2. Purpose: Process CAERØ5 cards
3. Calling Sequence: CALL APD5
4. Method: Same as APD4

FUNCTIONAL MODULE APD (AERODYNAMIC POOL DISTRIBUTOR)

THIS PAGE HAS BEEN LEFT BLANK INTENTIONALLY.

MODULE FUNCTIONAL DESCRIPTIONS

4.112.9 Design Requirements

4.112.9.1 Allocation of Open Core

APD uses 5 scratch files.

Open core is allocated in APD as follows:

COMMON/ZAPD/ Z(1)	
A11 AEFACT Cards	
CSTM Table	
A11 CAERØ cards	
A11 PAERØ cards	
Scratch Area	
12 Buffers	

4.112.9.2 Communication Common Blocks

COMMON/ZAPD/ Z(1) open core - all offsets are from here

COMMON/APD1C/EID,PID,CP,NSPAN,NCHØRD,LSPAN,LCHØRD,IGID,X1,Y1,Z1,X12,X4,Y4,Z4,X43,XØP,X1P,ALZØ,
 MCSTM,NCST1,NCST2,CIDBY,ACSID,IACS,SILB,NCRD,SCR1,SCR2,SCR3,SCR4,SCR5,ECTA,BGPA,
 GPLA,USETA,SILA,CSTMA,ACPT,BUF1Ø,BUF11,BUF12,NEXT,LEFT,ISILN,NCAM,NAEF1,NAEF2,
 NCA1,NCA2,CA2S1,CA2E,CA3S,CA3E,CA4S,CA4E,NPA1,NPA2,PA2S,PA2E,PA3S,PA3E,PA4S,PA4E,
 CA5S,CA5E,PA5S,PA5E

As each CAERØ card is processed, the 16 words are put in /APD1C/ starting at EID.

MCSTM - last coordinate system ID used
 NCST1-NCST2 - start and end of CSTM data
 CIDBY - last external ID assigned
 ACSID - aero coordinate system ID
 IACS - pointer to core of CSTM for ACSID
 SILB - start of last SILA assigned
 NCRD - last external grid point ID assigned
 SCR1-ACPT - GINØ file numbers
 BUF1Ø-BUF12 - three buffers available for use
 NEXT - offset to next core location available
 LEFT - words from NEXT to END

FUNCTIONAL MODULE APD (AERODYNAMIC POOL DISTRIBUTOR)

ISILN - start of last SILGA assigned
NCAM - number of CAERØ1 cards
NAEF1-NAEF2 - start and end of AEFACT cards
NCA1-PA5E - starts and ends of CAERØ and PAERØ cards

CØMMØN/APDID/ICPL(14),DUM(6),RA1(3)

ICPL - CSTM entry for panel last generated
DUM - communication between APDCS and card processors as needed
RA1 - location of point 1 in aero coordinate system for last panel or body processed.

4.112.9.3 Diagnostic Messages

Since APD is a data card processor, it can produce many diagnostic messages. All of APD messages are fatal. A partial list of messages follows. (Some messages have not been given message numbers.)

2025, 3001, 3002, 3003, 3008, 2318, 2319, and 2322-2328.

4.113 FUNCTIONAL MODULE GI (GEOMETRY INTERPOLATOR)

4.113.1 Entry Point: GI

4.113.2 Purpose

To generate the transformation matrix $[G_{ka}]^T$. (GTKA)

4.113.3 DMAP Calling Sequence

GI SPLINE, USET, CSTMA, BGPA, SIL, xxx, GM, GØ/GTKA/V,N,NK/V,N, LUSET \$

4.113.4 Input Data Blocks

SPLINE - Splining Data Table.

USET - Displacement set definition table.

CSTMA - Coordinate System Transformation Matrices - aerodynamics.

BGPA - Basic grid point definition table - aerodynamics.

SIL - Scalar index list.

xxx - Available for future use.

GM - Multipoint constraint transformation matrix.

GØ - Structural matrix partitioning transformation matrix.

Notes:

1. SPLINE, USET, BGPA, SIL and ECTA cannot be purged.
2. CSTMA may be purged if all points are in the basic system.
3. GM and GØ may be purged if there are no multipoint constraints or no omitted points.

4.113.5 Output Data Blocks

GTKA - Aerodynamic transformation matrix - k set to a set.

Notes: GTKA cannot be purged.

4.113.6 Parameters

NK - Input-integer-no default. Degrees of freedom in k aerodynamic set.

LUSET - Input-integer-no default. Degrees of freedom in g displacement set.

4.113.7 Method

The GI Module runs in four steps. Subroutine GI is the module driver and after initializing the common block GICØM, it calls a subroutine to perform each step. The steps are:

1. For each SPLINE data card, determine the structural (g points) and aerodynamic (k points) connected to it. (GIGGKS)

MODULE FUNCTIONAL DESCRIPTIONS

2. Form $[G_{kg}^T]$ with SPLINE3 columns - other columns null (GIGTKG).
3. Form $[G_{ka}^T]$ - interpolation matrices from all splines into k set by G set matrix. (GIPSST)
4. Reduce to analysis (a) set degrees of freedom and output $[G_{ka}^T]$. If there are no multi- or single-point constraints and no omits, part four is not performed. (GIGTKA)

4.113.7.1 GIGGKS

Subroutine GIGGKS first determines the internal numbers of the g points and puts each set on a scratch file. SET1 cards are copied onto the scratch file directly with an end-of-record after each set.

For SET2 processing, the CSTMA and BGPA are read into core. Then for each SET2 card:

1. Points 1 and 4 of the associated CAERØ element are found in the CAERØ coordinate system, giving two vectors R_1^E and R_4^E .
2. Two other vectors R_2^E and R_3^E are found from the two vectors above and the data on the CAERØ card for locating points 2 and 3.
3. Then the vectors defining the corners of the prism defined on the SET2 card is found from the following equation:

$$\{R\}_i^P = \begin{Bmatrix} x \\ y \end{Bmatrix}_i^P = (1-\xi_1)(1-\eta_1) R_1^E + (1-\xi_1)\eta_1 \{R\}_2^E + \xi_1 \eta_1 \{R\}_3^E + \xi_1(1-\eta_1) \{R\}_4^E \quad (1)$$

where R_1 through R_4 are the CAERØ corner locations in element coordinates and ξ and η are determined from the SET2 card using the following table:

	ξ_1	η_1
1	SP1	CH1
2	SP1	CH2
3	SP2	CH2
4	SP2	CH1

4. The system of inequalities for points within the prism in the element coordinate system is:

$$[C_E]\{R_E\} \geq \{b_E\} \quad (2)$$

where:

$$[C_E] = \begin{bmatrix} y_1 - y_2 & x_2 - x_1 & 0 \\ y_2 - y_3 & x_3 - x_2 & 0 \\ y_3 - y_4 & x_4 - x_3 & 0 \\ y_4 - y_1 & x_1 - x_4 & 0 \\ 0 & 0 & -1 \\ 0 & 0 & 1 \end{bmatrix} \quad (3)$$

$$\{b_E\} = \begin{pmatrix} x_2 \cdot y_1 - x_1 \cdot y_2 \\ x_3 \cdot y_2 - x_2 \cdot y_3 \\ x_4 \cdot y_3 - x_3 \cdot y_4 \\ x_1 \cdot y_4 - x_4 \cdot y_1 \\ -z_{\max} \\ z_{\min} \end{pmatrix} \quad (4)$$

In these definitions, x and y are for the corners of the prism, z_{\max} and z_{\min} are from SET2 data card, and $\{R_E\}$ is the location $[x,y,z]$ of a point to be checked in the element coordinate system. If $z_{\max} = 0.0$ (implying $+\infty$) or $z_{\min} = 0.0$ (implying $-\infty$), the fifth or sixth (or both) rows of the inequality are ignored. The inequality will pass if all equations are satisfied.

5. The inequalities are transformed to basic coordinates

$$C_B = C_E T_E^T \quad (5)$$

$$b_B = b_E + C_E T_E^T R_{OE} \quad (6)$$

6. Now all grid points in BGPA are checked. Those grid points which satisfy all inequalities are in the set; the inequalities are in the basic coordinate system. The test is

$$[C_B]\{x_B\} \geq \{b_B\} \quad (7)$$

where $\{x_B\}$ is the location vector of a grid point to be checked.

7. Write each set on a scratch file.

MODULE FUNCTIONAL DESCRIPTIONS

After the sets have been written on scratch files the complete SIL and part of the SPLINE associated with aerodynamic mid-points (k-points) are read into core.

The rest of the subroutine is a loop to process each SPLINE data card. Utility subroutine PRELOC (LOCATE) is used to find the SPLINE card records. The SPLINE1 or SPLINE2 card is read into the beginning of core. The associated g points are found from a scratch file containing the sets. If the proper SET card cannot be found or if the set is null, an appropriate message is produced. The g set is sorted by internal number and left in core. Then the k set is determined from information on the SPLINE card and its associated CAEROi card. Appropriate error messages occur if the proper boxes on the CAEROi element cannot be found. The associated external k set numbers can be found by evaluating the following equation for all values of I and J between their maximum and minimum.

$$ID_k = ID_E + (I-1) + NCNORD(J-1) \quad (8)$$

$$1 \leq I \leq NCNORD$$

ID_E = CAEROi element number

where

NCNORD comes from CAEROi card

$$I_{min} = \text{Box 1} - ID_E / NCNORD + 1$$

$$I_{max} = \text{Box 1} - ID_E - NCNORD(J_{min} - 1) + 1$$

The external numbers are converted to internal numbers from the SPLINE table; then they are sorted and put behind the g set. Using the SIL table, a list of SIL numbers for each g and k is put in core behind the first set. Then a record of the following is written in SCR1, and the process repeats for each SPLINE card.

Words

1-10	SPLINE card as received except word 2 was changed to type of spline.
11-26	CAEROi card referenced by above SPLINE card.
27	NGSET - number of grid points in g set
28	NKSET - number of grid points in k set
29	g set points
.	.
.	.
29+NGSET	k set points

FUNCTIONAL MODULE GI (GEOMETRY INTERPOLATOR)

```

:
:
NSKET      n set SIL
:
:
NGSET      g set SIL
:
:
NKSET

```

The length of the longest record is put in word 1 of the trailer for SCR1. For SPLINE3 cards, all the cards are read into core. The external aero ID's are converted to k-set numbers and the g-points to SIL numbers. The cards are then written on SCR3 sorted by k-set numbers.

4.113.7.2 GIGTKG

Subroutine GIGTKG builds on SCR2 $[G_{kg}^T]$. If any SPLINE3 data cards were used, SCR3 is read into core. As null k-columns are output, a check is made to see if a SPLINE3 card defines this column. If it does, this vector is built in core by inserting the proper rows' terms from the SPLINE3 card, and this column is then packed out.

4.113.7.3 GIPSST

Subroutine GIPSST determines the amount of open core. Core is allocated for three buffers, the CSTMA and BGPA. If this plus twice the maximum record size of SCR1 will not fit in core, a fatal message occurs. BGPA and CSTM (if it exists) is read into core. The rest of the subroutine is a loop on each record on SCR1.

As each record is read, pointers are set up for g points, k points, g SIL's, k SIL's and the coordinate system transformation matrices. A branch on type is then made. For the following equation, E means aero element, L means linear spline, G means global and B means basic for matrix or vector subscripts. For the surface splines, all g and k points must be located in the spline (or element) coordinate system:

$$\begin{Bmatrix} x \\ y \\ z \end{Bmatrix}_E = [T_E]^T \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}_B - \{R_{\emptyset E}\} \quad (9)$$

only the x and y location are calculated and saved. Then a core check is made, so the SSPLINE routine will not give an error. Then SSPLINE is called to give $[G_{kgi}^T]$, with no symmetric options, transpose option, slope option.

MODULE FUNCTIONAL DESCRIPTIONS

For the linear spline, all g and k points must be located in the spline coordinate system, but this may not be the coordinate system for the element. For bodies, the aero coordinate system is used for all calculations. The z-axis will always be perpendicular to the defining element. The y-axis of the element is the projection of the y-axis of the coordinate system (L) referenced on the SPLINE2 data card.

Definition of terms:

$$[T_k] = [e_{x,k} \mid e_{y,k} \mid e_{z,k}] \quad (10)$$

where k may be E, L or b. Then $R_{\emptyset E}$ in Equation (2) is redefined for linear splines

$$\{R_{\emptyset E}\} = \{R_{\emptyset L}\} - [\{e_{z,E}\} \cdot \{R_{\emptyset L} - R_{\emptyset E}\}][E_{z,E}] \quad (11)$$

To get $[T_e]$

$$\{e_{z,E}\} = \{e_{z,E}\} \quad (12)$$

$$\{e_{x,E}\} = \{e_{y,L} \times e_{z,E}\} / |e_{y,L} \times e_{z,E}| \quad (13)$$

$$\{e_{y,E}\} = \{e_{z,E} \times e_{x,E}\} \quad (14)$$

Equation (2) is the calculation for the linear spline with only the x and y locations being calculated. Core check is made before LSPLIN is called to give G_{kgi}^T with no symmetry, both slopes and transposed for panels (see Section 3.4.86). For bodies, the x-location is placed in the y-position and the x-position set to 0. Only the y-slope is calculated.

For panels, this matrix has two columns for slopes and these must be transformed to element coordinates (which the surface spline is in).

Two scalars are needed:

$$TyL = \{e_{y,E}\} \cdot \{e_{y,L}\} \quad (15)$$

$$TxL = \{e_{x,E}\} \cdot \{e_{x,L}\} \quad (16)$$

FUNCTIONAL MODULE GI (GEOMETRY INTERPOLATOR)

The two columns are then merged into one column (naturally the rest of the matrix must move up).

$$C\emptyset L_1(I) = C\emptyset L_1(I) \cdot TyL + C\emptyset L_2(I) \cdot TxL \quad (17)$$

$I = 1$ to length of column. For bodies, this matrix is ready to merge in to G_{kg}^T .

This gives the proper G_{kgi}^T to merge into the full G_{kg}^T .

As each spline is processed, appropriate null columns are replaced in G_{kg}^T (if a column to replace is not null an error is given).

Before inserting into G_{kg}^T , each degree of freedom must be transformed to the global coordinates for the g points. The equation to solve is

$$\{U\}_g = [T_G]^t [T_E] \{U\}_E \quad (18)$$

where $\{U\}_g$ = six degrees of freedom per grid point and $\{U\}_E$ = six degrees of freedom per grid point from G_{kgi}^T for each row in G_{kgi}^T for surface splines:

$$\{U\}_e = \begin{Bmatrix} 0 \\ 0 \\ z \\ 0 \\ 0 \\ 0 \end{Bmatrix} \text{ where } z = \text{term in } G_{kgi}^T \quad (19)$$

Only the nonzero terms in Equation (18) are solved for and the terms are placed in the full G_{kg}^T from the SIL value.

For the linear spline, each grid point has three row positions in G_{kgi}^T , so:

$$\{U\} = \begin{Bmatrix} 0 \\ 0 \\ z \\ \theta_x \\ \theta_y \\ 0 \end{Bmatrix} \text{ where } z, \theta_x, \theta_y \text{ are sequential row terms in } G_{kgi}^T \quad (20)$$

Equation (18) is solved and the terms are placed in the full G_{kg}^T from the SIL value. For bodies, $[T_G]^T$ must be modified depending on body type. The process is repeated for each row in G_{kgi}^T for surface splines or each three rows of G_{kgi}^T for linear splines. This is repeated again for each column of G_{kgi}^T .

The final output is on SCR2.

MODULE FUNCTIONAL DESCRIPTIONS

4.113.7.4 GIGTKA

Subroutine GIGTKA reduces $[G_{kg}^T]$ to $[G_{ka}^T]$, much like module SSG2, using the following matrix operations:

$$[G_{kg}^T] \rightarrow \begin{bmatrix} G_{kn} \\ G_{km} \end{bmatrix}$$

$$[G_{kn}^T] = [G_m^T][G_{km}^T] + [G_{kn}^T]$$

$$[G_{kn}^T] \rightarrow \begin{bmatrix} G_{kf}^T \\ G_{ks}^T \end{bmatrix}$$

$$[G_{kf}^T] \rightarrow \begin{bmatrix} G_{ka}^T \\ G_{ko}^T \end{bmatrix}$$

$$[G_{ka}^T] = [G_o^T][G_{ko}^T] + [G_{ka}^T]$$

At each step where a matrix multiply is indicated, the multiply is skipped if the result is known to be zero (i.e., U_m or U_o are null).

4.113.8 Subroutines

Utility subroutines PRELOC, LOCATE and SORT are used by GIGGKS. Utility subroutines PRETRS, TRANSS, GMMATS, SSPLIN, LSPLIN are used by GIPSST. INVERS and GMMATS are used by SSPLIN and LSPLIN. Utility subroutines CALCV, SSG2A and SSG2B are used by GIGTKA. In addition, PARTN is used by CALCV and MPYAD by SSG2B.

4.113.8.1 Subroutine Name: GI

1. Entry Point: GI
2. Purpose: Module Driver
3. Calling Sequence: CALL GI

4.113.8.2 Subroutine Name: GIGGKS

1. Entry Point: GIGGKS
2. Purpose: See discussion above.
3. Calling Sequence: CALL GIGGKS

FUNCTIONAL MODULE GI (GEOMETRY INTERPOLATOR)

4.113.8.3 Subroutine Name: GIPSST

1. Entry Point: GIPSST
2. Purpose: See discussion above.
3. Calling Sequence: CALL GIPSST

4.113.8.4 Subroutine Name: GIGTKA

1. Entry Point: GIGTKA
2. Purpose: See discussion above.
3. Calling Sequence: CALL GIGTKA

4.113.8.5 Subroutine Name: GIGTKG

1. Entry Point: GIGTKG
2. Purpose: See discussion above
3. Calling Sequence: CALL GIGTKG

4.113.9 Design Requirements

Each step in the module has its own open core common block, the use of which is discussed under the method section.

In addition, a communication common block GICØM is used to communicate between steps. GICØM has the following format:

One word for each of the following GINØ file numbers: SPLINE, USET, CSTMA, BGPA, SIL, ECTA, GM, GØ, GTKA, SCR1, SCR2, SCR3, SCR4, SCR5

NS1 - number of SPLINE1 cards (used by GIGGKS only)

NS2 - number of SPLINE2 cards (used by GIGGKS only)

KSIZE - number of degrees of freedom in k set - input (moved from blank common)

GSIZE - number of degrees of freedom in g set - input (moved from blank common)

GI uses six scratch files.

4.113.10 Diagnostic Messages

Since this modules does some data checking while processing SPLINE cards, the module tries to continue where possible to look for errors after a fatal error is found.

MODULE FUNCTIONAL DESCRIPTIONS

The error messages for which processing must stop are 3001, 3002, 3003, 3007, 3008 and 3061. Error messages for which some error checking will continue are 2258 - 2263. In addition, the warning message 2257 can occur. Detailed descriptions for these messages can be found in Section 6 of the User's Manual.

FUNCTIONAL MODULE AMG (AERODYNAMIC MATRIX GENERATOR)

4.114 FUNCTIONAL MODULE AMG (AERODYNAMIC MATRIX GENERATOR)

4.114.1 Entry Point: AMG

4.114.2 Purpose

To generate a list of aerodynamic influence matrices (AJJL) and the transformation matrices needed to convert these to the interpolated structural system (SKJ, DIJK, D2JK).

4.114.3 DMAP Calling Sequence

AMG AERØ,ACPT/AJJL,SKJ,S1JK,D2JK/V,N,NK/V,N,NJ \$

4.114.4 Input Data Blocks

AERØ - Aerodynamic matrix generation data

ACPT - Aero connection and property data.

Note: Neither AERØ or ACPT may be purged.

4.114.5 Output Data Blocks

AJJL - Aerodynamic influence matrix list

SKJ - Integration matrix list

DIJK - Real part of downwash matrix

D2JK - Imaginary part of downwash matrix

4.114.6 Parameters

NK - Input - integer - no default, number of degrees of freedom in k-set

NJ - Input - integer - no default,

$$N_j = \sum N_b + \sum_{i=1}^{N_{sb}} f_i + \sum_{j=1}^{N_{ib}} g_i$$

where N_b = number of aero boxes

N_{sb} = number of slender bodies

N_{ib} = number of interference bodies

f_i = degrees of freedom for each slender body

g_i = degrees of freedom for each interference body

FUNCTIONAL MODULE AMG (AERODYNAMIC MATRIX GENERATOR)

4.114.7 Method

Module AMG is broken into two sections. Section one outputs AJJL and SKJ and Section two outputs D1JK and D2JK. Each section has a branch on method to output columns of the matrices. Each section has a common block to communicate between the module driver and the method dependent code. (See description of design requirements.)

FUNCTIONAL MODULE AMG (AERODYNAMIC MATRIX GENERATOR)

THIS PAGE HAS BEEN LEFT BLANK INTENTIONALLY.

MODULE FUNCTIONAL DESCRIPTIONS

The flow for Section one is as follows: Four buffers are allocated from the bottom of core and ACPT, AERØ, AJJL, and SKJ are opened. Record 1 of AERØ is read into /AMGMN/ at ND and the header and trailer for AJJL are initialized by passing the ACPT and counting the method groups.

The rest of Section one involves a loop where pairs of m and k are read into /AMGMN/ and then each record in the ACPT is processed. For each record of ACPT processed, one word (the method) is read and then a branch on method is taken. Each method is expected to:

1. Read its record of the ACPT and leave it positioned to read the next record.
2. Output columns of AJJL and SKJ of proper size.

The row number to start at is NRØW+1 for AJJL and ISK for SKJ.

3. Always increment NRØW, ISK and NSK by the number of rows added.

Once an end of file is reached in AERØ, AERØ, AJJL and SKJ are closed and Section two starts.

The flow in Section two is as follows: Three buffers are allocated and D1JK and D2JK are opened (ACPT is left open). The trailers are initialized and the method of ACPT is read. The program branches on method and then loops until all records on the ACPT have been processed. When an EØF is reached the files are closed and their trailers written.

4.114.7.1 Doublet Lattice Method without Bodies

The flow for Section one of the Doublet Lattice method is as follows: Subroutine DLAMG is the driver for this method. It reads in the ACPT record for this method and sets up the pointers to the various arrays in common DLCØM. Columns of SKJ are output. If there is enough core available, GEND is called to output one matrix of the AJJL list. When GEND is through, DLAMG bumps NRØW and returns.

Subroutine GEND outputs a row of the AJJL matrix for each box on the CAERØ1 element. To do this, it picks up the proper strip and panel data and calls DPPS once for each box. A row is packed out after each call to DPPS.

Subroutine DPPS is also in a loop. There is one row element for each box and DPPS prepares the variables necessary for the computation of each element and calls SUBP to calculate the element. When the row is done DPPS returns to GEND.

Subroutine SUBP computes the downwash factor elements by calling subroutines SNPDF and INCRØ to compute the indicated components that make up the element. SNPDF computes the steady downwash factors and INCRØ computes the unsteady downwash factors for one receiving-sending point combination.

FUNCTIONAL MODULE AMG (AERODYNAMIC MATRIX GENERATOR)

Each combination has four influence quadrants (upper left, upper right, lower left, lower right), so these routines must be called four times for each element and then the result summed before SUBP returns. Subroutine INCRØ uses subroutines TKER, IDF1, and IDF2 to compute the final result.

The flow for Section two of the Doublet Lattice method is as follows: Subroutine DLPT2 prepares all the computations necessary. DLPT2 reads the record of ACPT and then loops through each box packing out a column of D1JK and D2JK for each box.

The row position of each pair of values for a column is $2*(\text{box number}-1) + 1$. Successive rows of SKJ (output in Section one) have the following form:

$$\text{SKJ} \rightarrow \begin{bmatrix} 2.0 * \text{EE}_{\text{strip}} * \text{DELX}_{\text{box}} \\ \text{-----} \\ \text{EE}_{\text{strip}} * \text{DELX}_{\text{box}}^2 / 2.0 \end{bmatrix} \quad . \quad (1)$$

Successive rows of D1JK have the following form:

$$\text{D1JK} \rightarrow \begin{bmatrix} 0.0 \\ \text{-----} \\ 1.0 \end{bmatrix} \quad . \quad (2)$$

Successive rows of D2JK have the following form:

$$\text{D2JK} \rightarrow \begin{bmatrix} -2.0/\text{REFC} \\ \text{-----} \\ \text{DELX}_{\text{box}}/2.0*\text{REFC} \end{bmatrix} \quad . \quad (3)$$

4.114.7.2 Doublet Lattice Method with Bodies

The flow for Section one of the Doublet Lattice method with bodies is as follows: Subroutine DLAMBG is the driver for this method. It reads the ACPT record and sets up the pointers to the various arrays in common DLBDY. The flow then depends on the type of problem submitted. If panels exist then GENDSB is called to build part of AJJL on a scratch file. If panels and slender bodies are used, AMGRØD is called to build this combination on a scratch file. Then AMGSBA is called to output this method's part of AJJL. AMGBFS is called to build SKJ for this method. Up to four additional buffers may be needed by this method.

Figure one shows the subroutines and scratch files used to put AJJL together.

MODULE FUNCTIONAL DESCRIPTIONS

	SCR5				
NTP	DPPSB	DZPY	DYPZ	DZYMAT	
	SCR1			SCR1	SCR2
NTZ	DPZY				
	SCR1				
NTY	DPZY				
	SCR4				
NTZS	0	0	AMGSBA		
NTYS	0	0			
AJJL					

NTP, NTZ, NTY, NTZS, and NTYS are lengths.

SCR1 - SCR5 are scratch files.

Other names are subroutines.

Figure 1. Subroutines and Scratch files for AJJL

Subroutine GENDSB calls DPPSB once for each row on SCR1 (NTP times), and DPZY once for each NTZ and NTY rows on SCR1 and SCR4. Then DZPY once for each row on SCR2 and DYPZ once for each row on SCR3. Once this process is done, GENDSB builds the data collected from SCR1, SCR2, SCR3 and SCR4 on SCR5.

Subroutine AMGRØD calls DZYMAT once to build SCR1 and once to build SCR2. Then AMGSBA is called to put SCR1, SCR2 and SCR5 together plus AMGSBA's part of AJJL onto AJJL.

Matrix SKJ is built by AMGBFS. It calls BFSMAT, builds some data on SCR1, then SKJ is built and packed out. (See the Theoretical Manual for description of SKJ.)

Figure 2 shows the various calls that may take place during Section one of this method.

For Section two DLBPT2 is called and the ACPT is read. Then the pointers to the necessary arrays are computed and D1JK and D2JK are packed out. D1JK and D2JK look like the Doublet Lattice Method without Bodies except D1JK has a -1. instead of a 1.0 for D1JK on y-slender elements, and D2JK has a 0.0 for slender elements instead of $DELX/2*REFC$.

FUNCTIONAL MODULE AMG (AERODYNAMIC MATRIX GENERATOR)

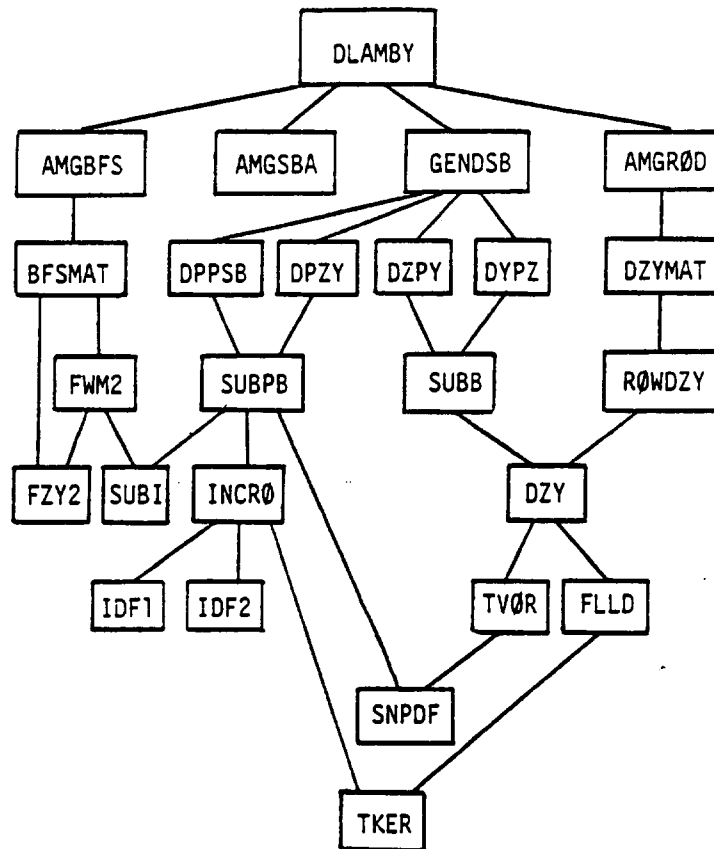


Figure 2. Section One Calls for the Doublet Lattice Method with Bodies

4.114.7.3 Mach Box Method

The flow for Section one of the Mach Box Method is as follows: Subroutine MBAMG is called to read the ACPT and set up pointers to arrays in common MBAMGX. MBAMG then makes calls to various subroutines to get AJJL built. Most of the arrays used by the Mach Box method are generated by subroutine MBREG, so, in general, fixed dimensions are used. One additional buffer is used.

Once the pointers, common MB0XN and common MB0XL are set up, MBAMG calls MBGE0D to set up the regions and geometry for the problem in common MB0XA. Then MBAMG calls MBREG to generate the boxes. MBREG fills in the box arrays based on the area to cover, Mach number, and number of boxes requested. MBCTR1 and MBCTR2 can be called to make box arrays if the control surface exists. MBPLOT is then called to print a picture of the planform regions.

MBAMG then calls MBM0DE to generate mode-like data on SCR2. The SSPLIN routine is used to spline from the Mach box points to the input control points for the wing, control one and control two separately.

MODULE FUNCTIONAL DESCRIPTIONS

The influence coefficients are computed by MBCAP, then SKJ (Identity) is output, and finally MBDDPDH is called to compute and output the AJJL contribution.

Section two is a call to STPPT2 with outputs DIJK (Identity) and D2JK (Null).

4.114.7.4 Strip Theory Method

Section one of the Strip Theory Method is driven by Subroutine STPDA. STPDA reads the ACPT, fills in common STRIPL, and sets up pointers to common STRIPX where the various arrays will be stored. After all the input arrays have been set up an SKJ (Identity) matrix is built.

STPDA then calls: STPBG to build a BM and GM matrix for each strip; STPPHI to build the PHI functions for each strip; and finally STPAIC to combine these matrices and build AJJL.

Section two is a call to STPPT2 which output DIJK (Identity) and D2JK (Null).

4.114.7.5 Piston Theory Method

Section one of the Piston Theory method is driven by subroutine PSTAMG. PSTAMG reads the ACPT and sets up the core pointer to the arrays. Then SKJ (Identity) is output and PSTA is called to build AJJL.

Section two is a call to STPPT2 with outputs DIJK (Identity) and D2JK (Null).

4.114.8 Subroutines

Besides the module driver AMG, the subroutines of Section one are divided into groups by method:

For the Doublet Lattice Methods five subroutines are shared:

SNPDF, INCRØ, TKER, IDF1, and IDF2

The Doublet Lattice Method without Bodies also uses:

DLAMG, GEND, DPPS, and SUBP

The Doublet Lattice Method with Bodies also uses:

DLAMBY, SUBI, AMGBFS, FZY2, FWMW, BFSMAT, AMGRØD, AMGSBA, GENDSB, DPPSB, DPZY, DYPZ, DZPY, SUBB, SUBPB, DZY, FLLD, TVØR, DZYMAT, and RØWDZY

FUNCTIONAL MODULE AMG (AERODYNAMIC MATRIX GENERATOR)

The Mach Box Method uses:

MBAMG, MBPRIT, MBGEØD, MBREG, MBCTR1, MBCTR2, MBPLØT, MBMØDE, MBCAP, MBBSLJ, GØ, ZJ, MBDPDH, MBGAE, MBGAW, MBGATE, SUMPHI, and TRAILÉ

The Strip Theory Method uses:

STPDA, STPBG, STPPHI, STPAIC, STPK, STPBSØ, and STPBS1

The Piston Theory Method uses:

PSTAMG and PSTA

For Section two the subroutines are DLPT2, STPPT2 and DLBPT2.

4.114.8.1 Subroutine Name: AMG

1. Entry Point: AMG
2. Purpose: Module driver for AMG - see description above.
3. Calling Sequence: CALL AMG

4.114.8.2 Subroutine Name: DLAMG

1. Entry Point: DLAMG
2. Purpose: Output AJJL and SKJ parts for Doublet Lattice without bodies.
3. Calling Sequence: CALL DLAMG (ACPT, AJJL, SKJ)
ACPT - GINØ file number of ACPT.
AJJL - GINØ file number of AJJL.
SKJ - GINØ file number of SKJ.
4. Core Requirement: Core needed is four buffers plus record of ACPT plus 2*NJ.

4.114.8.3 Subroutine Name: GEND

1. Entry Point: GEND
2. Purpose: Output all the columns of AJJL associated with a record on ACPT.
3. Calling Sequence: CALL GEND (NCARAY,NBARAY,YS,ZS,SG,CG,DT,WØRK,MATØUT)
NCARAY, NBARAY, YS, ZS, SG, and CG are the locations of these arrays from the ACPT record.
DT - location to put column of AJJL - complex
WØRK - start of open core
MATØUT - GINØ file number of AJJL.

MODULE FUNCTIONAL DESCRIPTIONS

4.114.8.4 Subroutine Name: DPPS

1. Entry Point: DPPS
2. Purpose: Compute the elements in a row of AJJL.
3. Calling Sequence: CALL DPPS (KS,I,J1,J2,SGR,CGR,YS,ZS,NBARAY,NCARAY,DT,WØRK)
KS - Strip number in which receiver point I lies
I - Box number of receiver point
J1 - 1
J2 - Number of boxes
SGR - Sine of dihedral angle of receiver strip (from SG array)
CGR - Cosine of dihedral angle of receiver strip (from CG array)
YS, ZS, NBARAY, NCARAY - location of these arrays from ACPT record
DT - location to start putting elements of column - complex
WØRK - start of open core

4.114.8.5 Subroutine Name: SUBP

1. Entry Point: SUBP
2. Purpose: Compute downwash factor element.
3. Calling Sequence: CALL SUBP(I,L,LS,J,SGR,CGR,YREC,ZREC,SUM,XIC,DELX,EE,XLAM,SG,CG,YS,ZS)
I - Box number of receiving point
L - Panel number in which sending point J lies
LS - Strip number in which sending point J lies
J - Box number of sending point (also row number of output column)
SGR - Sine (see DPPS)
CGR - Cosine (see DPPS)
YREC - YS(KS) - y coordinate from ACPT array
ZREC - ZS(KS) - z coordinate from ACPT array
SUM - Output element - complex
XIC,DELX,EE,XLAM,SG,CG,YS,ZS - locations of these arrays for ACPT record.

4.114.8.6 Subroutine Name: SNPDF

1. Entry Point: SNPDF
2. Purpose: Compute the steady downwash factors for one receiving-sending point combination.

FUNCTIONAL MODULE AMG (AERODYNAMIC MATRIX GENERATOR)

3. Calling Sequence: CALL SNPDP (SL,CL,TL,SGS,CGS,SGR,CGR,XO,YO,ZO,ES,DIJ,BETA,CV)

- SL - Sine of sweep angle of sending box
- CL - Cosine of sweep angle of sending box
- TL - Tangent of sweep angle of sending box (from ACPT)
- SGS - Sine of dihedral angle of sending point
- CGS - Cosine of dihedral angle of sending point
- SGR - Sine of dihedral angle of receiving point
- CGR - Cosine of dihedral angle of receiving point
- XO - X coordinate of receiving point, X coordinate of "center" of sending point
- YO - Y coordinate of receiving point, Y coordinate of "center" of sending point
- ZO - Z coordinate of receiving point, Z coordinate of "center" of sending point
- ES - Sending point strip half width
- DIJ - Steady contribution to downwash - output
- BETA - Square root of $1.0-M^2$
- CV - Chord of sending point

4.114.8.7 Subroutine Name: INCRØ

1. Entry Point: INCRØ
2. Purpose: Computes the unsteady downwash factor for one receiving-sending point combination
3. Calling Sequence: CALL INCRØ (AX,AY,AZ,AX1,AY1,AZ1,AX2,AY2,AZ2,SGR,CGR,SGS,CGS,KR,RL,BETA,
SDELX,DELY,DELR,DELI)

- AX - XO
- AY - YO
- AZ - ZO
- AX1 - $XO+ES*TL$
- AY1 - $YO+ES*CGS$
- AZ1 - $ZO-ES*SGS$
- AX2 - $XO-ES*TL$
- AY2 - $YO-ES*CGS$
- AZ2 - $ZO-ES*SGS$
- SGR -
- CGR -
- SGS -
- CGS -

See definitions for SNPDP (Section 4.114.8.6).

MODULE FUNCTIONAL DESCRIPTIONS

KR - Reduced frequency
 RL - REFC
 BETA - Square root of $1.0-M^2$
 SDELX - Box chord of sending point
 DELY - $2.0 \times$ sending point strip half width
 DELR - Output - real part of downwash factor
 DELI - Output - imaginary part of downwash factor

4. Method: INCRØ calls TKER for three points for each receiving-sending box combination (the center, the inboard point, and the outboard point). Then INCRØ calls IDF1 and IDF2 to perform the integration of the kernels.

4.114.8.8 Subroutine Name: TKER

1. Entry Point: TKER
2. Purpose: Compute incremental oscillating kernel
3. Calling Sequence: CALL TKER(X,Y,Z,KR,BR,SGR,CGR,SGS,CGS,T1,T2,M)

<p> X - AX, AX1, or AX2 for center, inboard, outboard Y - AY, AY1, or AY2 for center, inboard, outboard Z - AZ, AZ1, or AZ2 for center, inboard, outboard </p>	}	see INCRØ (see Section 4.114.8...
--	---	-----------------------------------

KR - Reduced frequency
 BR - REFC/2.0
 SGR, CGR, SGS, CGS - See SNPDF (section 4.114.8.6)
 T1 - Output - cosine $(\gamma_r - \gamma_s)\gamma$ - dihedral angle (receiving (r) or sending (s))
 T2 - Output - $[(Z \cos \gamma_r - Y \sin \gamma_r) \times (Z \cos \gamma_s - Y \sin \gamma_s)] / (BR/M)^2$
 M - Mach number
4. Method: Kernel components are returned in common DLM.

4.114.8.9 Subroutine Name: IDF1

1. Entry Point: IDF1
2. Purpose: Integration of the planar parts of the kernels
3. Calling Sequence: CALL IDF1(EE,E2,ETA01,ZET01,ARE,AIM,BRE,BIM,CRE,CIM,R1SQX,XIIJR,XIIJI)

EE - Sending strip half width
 E2 - EE^2

FUNCTIONAL MODULE AMG (AERODYNAMIC MATRIX GENERATOR)

ETA01 - $AY \cos \gamma_r + AZ \sin \gamma_r$ (AY and AX, see INCR0)(γ see TKER) (Sections 4.114.7, 4.114.8)

ZET01 - $AZ \cos \gamma_r - AY \sin \gamma_r$

ARE, AIM, BRE, BIM, CRE, CIM - coefficients of the parabola for planar part

R1SQX - $AY^2 + AZ^2$

XIIJR - output - real part of planar integral contribution

XIIJI - output - imaginary part of planar integral contribution

4.114.8.10 Subroutine Name: IDF2

1. Entry Point: IDF2
2. Purpose: Integration of the nonplanar parts of kernels
3. Calling Sequence: CALL IDF2(EF,E2,ETA01,ZET01,A2R,A2I,B2R,B2I,C2R,C2I,R1SQX,DIIJR,DIIJI)
EF, E2, ETA01, ZET01 - same as IDF1 (Section 4.114.8.9)
A2R, A2I, B2R, B2I, C2R, C2I - coefficients of the parabola for the nonplanar part
R1SQX - See IDF1 (Section 4.114.8.9)
DIIJR - output - real part of nonplanar integral contribution
DIIJI - output - imaginary part of nonplanar integral contribution

4.114.8.11 Subroutine Name: DLPT2

1. Entry Point: DLPT2
2. Purpose: To output the Doublet Lattice without Bodies parts for matrices D1JK and D2JK.
3. Calling Sequence: CALL DLPT2(ACPT,D1JK,D2JK)
ACPT - GIN0 number
D1JK - GIN0 number
D2JK - GIN0 number

4.114.8.12 Subroutine Name: DLAMBY

1. Entry Point: DLAMBY
2. Purpose: Output AJJL and SKJ parts for Doublet Lattice with Bodies
3. Calling Sequence: CALL DLAMBY(ACPT,AJJL,SKJ)
ACPT, AJJL, and SKJ are GIN0 file numbers
4. Core Requirements: Four buffers plus record of ACPT plus 4*NJ.

MODULE FUNCTIONAL DESCRIPTIONS

4.114.8.13 Subroutine Name: GENDSB

1. Entry Point: GENDSB
2. Purpose: Generate part of the AJJL influence coefficient matrix
3. Calling Sequence: CALL GENDSB(NCARAY,NBARAY,SG,CG,NFL,NBEA1,NBEA2,IFLA1,IFLA2,DT,DPY)
NCARAY to IFLA2 - the locations of these arrays from ACPT record
DT - storage for 2*NJ words
DPY - storage for 2*NJ words
4. Core Requirements: Up to 4 buffers may be used (2 for Y bodies, 1 for Z bodies, and 1 for panels).

4.114.8.14 Subroutine Name: DPPSB

1. Entry Point: DPPSB
2. Purpose: Compute the element in a panel on panel row of AJJL.
3. Calling Sequence: CALL DPPSB(KS,I,J1,J2,SGR,CGR,YS,ZS,NBARAY,NCARAY,DT,WORK)
Same as DPPS

4.114.8.15 Subroutine Name: DPZY

1. Entry Point: DPZY
2. Purpose: Compute the elements in an interference element on a panel in AJJL
3. Calling Sequence: CALL DPZY(KB,IZ,I,J1,J2,IFIRST,ILAST,YB,ZB,AVR,ARB,TH1A,TH2A,NT121,NT122,NBARAY,NCARAY,NZYKB,DPZ,DPY)
KB - Body number in which receiving point I lies
IZ - Body element number of body KB in which I lies
I - Receiving point
J1 - Starting element number
J2 - Ending element number
IFIRST - θ_1 starting element
ILAST - θ_1 ending element
YB to NCARAY - locations of arrays in ACPT record
NZYKB - Z-Y flag
DPZ - Storage for row of AJJL
DPY - Storage for row of AJJL

FUNCTIONAL MODULE AMG (AERODYNAMIC MATRIX GENERATOR)

4.114.8.16 Subroutine Name: DZPY

1. Entry Point: DZPY
2. Purpose: Compute the elements in a column of AJJL for Z interference elements
3. Calling Sequence: CALL DZPY(KB,KS,LS,I,J1,J2,NYFLAG,SGR,CGR,FMACH,ARB,NBEA1,DT)
KB - See DPZY (Section 4.114.8.15)
KS - index of receiving point Y-Z coordinates
LS - strip number
I,J1,J2 - See DPZY (Section 4.114.8.15)
NYFLAG- Type to build
SGR,CGR - See DPPS (Section 4.114.8.4)
FMACH - Mach number
ARB,NBEA1 - location of arrays in ACPT record
DT - Storage for row of AJJL

4.114.8.17 Subroutine Name: DYPZ

1. Entry Point: DYPZ
2. Purpose: Compute the elements in a column of AJJL for Y-interference elements
3. Calling Sequence: CALL DYPZ(KB,KS,LS,I,J1,J2,NYFLAG,SGR,CGR,FMACH,ARB,NBEA1,LB0,LSO,JBO,DT)
KB to NBEA1 - See DZPY (Section 4.114.8.16)
LB0 - first body with Y orientation
LSO - Z-Y coordinate index for first element of LB0
JBO - Sending point index for first Y oriented body element
DT - Storage for row of AJJL

4.114.8.18 Subroutine Name: SUBPB

1. Entry Point: SUBPB
2. Purpose: Compute downwash factor elements on panels
3. Calling Sequence: CALL SUBPB(I,L,LS,J,SGR,CGR,YREC,ZREC,SUM,XIC,DELX,EE,XLAM,SG,CG,YS,ZS,
NAS,NASB,AVR,ZB,YB,ARB,XLE,XTE,X,NB)
I to SUM - See SUBP (Section 4.114.8.5)
XIC to X - location of arrays from ACPT record
NB - number of bodies

MODULE FUNCTIONAL DESCRIPTIONS

4.114.8.19 Subroutine Name: SUBB

1. Entry Point: SUBB
2. Purpose: Compute downwash factor elements on bodies
3. Calling Sequence: CALL SUBB(KB,KS,I,J,JB,LS,NDY,NYFL,PI,EPS,SGR,CGR,AR,BETA,SUM,RIA,DELX,YB,ZB,YS,ZS,X)

KB - index of receiving body

KS - strip number of receiving point

I - receiving point index

J - sending point index

JB - sending point index

LB - body number of sending point

NDY - Z-Y flag

NYFL - type to build

PI - π

EPS - .00001

SGR - CGR - See DPPS (Section 4.114.8.4)

AR - aspect ratio of body

BETA - See SNPDF (Section 4.114.8.6)

SUM - Output

RIA-X - locations of arrays in ACPT record

4.114.8.20 Subroutine Name: SUBI

1. Entry Point: SUBI
2. Purpose: Compute the image point coordinates inside associated bodies on MU-Z and MU-Y.
3. Calling Sequence: CALL SUBI(DA,DZB,DYB,DAR,DETA,DZETA,DCGAM,DSGAM,DEE,DXI,TL,DETAI,DZETAI,DLGAMI,DSGAMI,DEEI,DTLAMI,DMUY,DMUZ,INFL,IOTFL)


See Reference 1 (Section 4.114.11) for argument list.

4.114.8.21 Subroutine Name: DZY

1. Entry Point: DZY
2. Purpose: Calculated effect of slender body element on a panel element
3. Calling Sequence: CALL DZY(X,Y,Z,SGR,CGR,SII,XI2,ETA,ZETA,AR,AO,KR,REFC,BETA,FMACH,LNS,1DZDY,DZDYR,DZDYI)

See Reference 1 (Section 4.114.11) for argument list.

4.114-14 (12/31/77)



FUNCTIONAL MODULE AMG (AERODYNAMIC MATRIX GENERATOR)

4.114.8.22 Subroutine Name: TVØR

1. Entry Point: TVØR
2. Purpose: Calculate normalwash at a point due to a trapezoidal unsteady index ring
3. Calling Sequence: CALL TVØR(SL1,CL1,TL1,SL2,CL2,TL2,SGS,CGS,SGR,CGR,X01,X02,Y0,Z0,E,
BETA,REFC,FMACH,KR,BRE,BIM)

See Reference 1 (Section 4.114.11) for argument list.

4.114.8.23 Subroutine Name: FLLD

1. Entry Point: FLLD
2. Purpose: Calculate the velocity normal to a surface due to a finite length line doubled.
3. Calling Sequence: CALL FLLD(X01,X02,Y0,Z0,SGR,CGR,SGS,CGS,KR,REFC,FMACH,E,L,KD1R,KD1I,
KD2R,KD2I)

See Reference 1 (Section 4.114.11) for argument list.

4.114.8.24 Subroutine Name: AMGRØD

1. Entry Point: AMGRØD
2. Purpose: Calculate normalwash at panels and interference elements due to slender elements
3. Calling Sequence: CALL AMGRØD(D,BETA)
D - storage for a row of AJJL
BETA - square root of $1.-M^2$

4.114.8.25 Subroutine Name: DZYMAT

1. Entry Point: DZYMAT
2. Purpose: Calculate a slender element column of AJJL
3. Calling Sequence: CALL DZYMAT(D,NFB,NLB,NTZYS,IDZDY,NTAPE,X,BETA,IPRT,NS,NC,YS,ZS,SG,CG,
YB,ZB,NBEA1)
D - storage for a row of AJJL
NFB - number of first body
NLB - number of last body
NTZYS - number of slender elements
IDZDY - Z-Y flag
NTAPE - GINØ file for output
X - location of array from ACPT record

MODULE FUNCTIONAL DESCRIPTIONS

BETA - see AMGRØD

IPRT - print flag

NS to NBEA1 - locations of array from ACPT record

4.114.8.26 Subroutine Name: RØWDZY

1. Entry Point: RØWDZY
 2. Purpose: To set up call to DZY
 3. Calling Sequence: CALL RØWDZY(NFB,NLB,RØW,NTZYS,D,DX,DY,DZ,BETA,IDZDY,NTAPE,SG,CG,IPRT,
YB,ZB,ARB,NSBEA,XIS1,XIS2,AØ)
- NFB,NLB,NTZYS,D,BETA,IIDZD1,NTAPE,IPRT - same as DZYMAT
- RØW - row position of answer
- DX,DY,DZ - X, Y, Z of receiving point
- SG,CG,YB,ZB,ARB,NSBEA,XIS1,XIS2,AØ - locations of arrays from ACPT record

4.114.8.27 Subroutine Name: AMGSBA

1. Entry Point: AMGSBA
 2. Purpose: Add slender body terms and pack out final AJJL for bodies
 3. Calling Sequence: CALL AMGSBA(AJJL,AØ,AR,NSBE,A)
- AJJL - GINØ file number
- AØ,AR - locations of arrays from ACPT record
- NSBE - number of slender body elements
- A - storage for a row of AJJL

4.114.8.28 Subroutine Name: AMGBFS

1. Entry Point: AMGBFS
 2. Purpose: Build the SKJ matrix for bodies
 3. Calling Sequence: CALL AMGBFS(SKJ,EE,DELX,NCARAY,NBARAY,XIS2,XIS1,AØ,AØP,NSBE)
- SKJ - GINØ file number
- EE to AØP - locations of arrays from ACPT record
- NSBE - number of slender body elements

FUNCTIONAL MODULE AMG (AERODYNAMIC MATRIX GENERATOR)

4.114.8.29 Subroutine Name: BFSMAT

1. Entry Point: BFSMAT
2. Purpose: Form force matrices for slender elements
3. Calling Sequence: CALL BFSMAT(ND,NE,NB,NP,NTP,LENGTH,NT0,SCR1,JF,JL,NAS,FMACH,YB,ZB,YS,ZS,X,DELX,EE,XIC,SG,CG,AR,RIA,NBEA1,NBEA2,NASB,NSARRAY,NCARAY,BFS,AVR,REFC,A0,XIS1,XIS2,KR,NSBEA,NT0)

See Reference 1 (Section 4.114.11) for argument list.

4.114.8.30 Subroutine Name: FWMW

1. Entry Point: FWMW
2. Purpose: Add in images, symmetry, plane and ground effects
3. Calling Sequence: CALL FWMW(ND,NE,SGS,CGS,IRB,A0,ARB,XBLE,XBTE,YB,ZB,XS,YS,ZS,NAS,NAS3,KR,BETA2,REFC,AVR,FWZ,FWY)

See Reference 1 (Section 4.114.11) for argument list.

4.114.8.31 Subroutine Name: FZY2

1. Entry Point: FZY2
2. Purpose: Calculate the force numbers for FWMW
3. Calling Sequence: CALL FZY2(XIJ,X1,X2,ETA,ZETA,YB,ZB,A,BETA2,REFC,KR,FZZR,FZZI,FZYR,FZYI,FYZR,FYZI,FYYR,FYYI)

See Reference 1 (Section 4.114.11) for argument list.

4.114.8.32 Subroutine Name: DLBPT2

1. Entry Point: DLBPT2
2. Purpose: Output the Doublet Lattice with Bodies parts of D1JK, D2JK
3. Calling Sequence: CALL DLBPT2(ACPT,D1JK,D2JK)
ACPT, D1JK, D2JK - GINØ file numbers

4.114.8.33 Subroutine Name: MBAMG

1. Entry Point: MBAMG
2. Purpose: Driver for Mach Box Method
3. Calling Sequence: CALL MBAMG(ACPT,AJJL,SKJ)
ACPT, AJJL, SKJ - GINØ file numbers

MODULE FUNCTIONAL DESCRIPTIONS

4.114.8.34 Subroutine Name: MBPRIT

1. Entry Point: MBPRIT
2. Purpose: Print geometry data
3. Calling Sequence: CALL MBPRIT(AW,AC,AT)
AW - area of wing
AC - area of control one
AT - area of control two

4.114.8.35 Subroutine Name: MBGEØD

1. Entry Point: MBGEØD
2. Purpose: Compute the geometry of the planform
3. Calling Sequence: MBGEØD

4.114.8.36 Subroutine Name: MBREG

1. Entry Point: MBREG
2. Purpose: Compute the limits of the region and the percentage of box in each
3. Calling Sequence: CALL MBREG(IREG,NW1,NWN,NC21,NC2N,NC1,NCN,ND1,NDN,XK,YK,XK1,YK1,XK2,
YK2,XWTE,YWTE,KTE,KTE1,KTE2,PAREA)
IREF - flag for MBREG success - 2 = fail
NW1 - PAREA - location of arrays which MBREG is to build

4.114.8.37 Subroutine Name: MBCTR1

1. Entry Point: MBCTR1
2. Purpose: Compute the region calculations for control one
3. Calling Sequence: CALL MBCTR1(IC1,IR1,NCN,NC1,NWN,NW1,PAREA)
IC1 - starting box number for control one
IR1 - ending box number for control one
NCN to PAREA - locations of arrays which MBCTR1 is to build

4.114.8.38 Subroutine Name: MBCTR2

1. Entry Point: MBCTR2
2. Purpose: Compute the region calculations for control two

FUNCTIONAL MODULE AMG (AERODYNAMIC MATRIX GENERATOR)

3. Calling Sequence: CALL MBCTR2(IL2,IR2,NC2N,NC21,NWN,NW1,PAREA)

IL2 - starting box for control two

IR2 - ending box for control two

NC2N to PAREA - location of arrays which MBCTR2 is to build

4.114.8.39 Subroutine Name: MBPLØT

1. Entry Point: MBPLØT

2. Purpose: Print a representation of the planform

3. Calling Sequence: CALL MBPLØT(NW1,ND1,NWN,NC21,NC2N,NC1,NCN,NDN)

NW1 - NDN - locations of arrays which define planform boxes

4.114.8.40 Subroutine Name: MBMØDE

1. Entry Point: MDMØDE

2. Purpose: Build the mode-like data from surface interpolation

3. Calling Sequence: CALL MBMØDE(ACPT,SCR2,ICØR,NCØR,Z,NI,ND,XD,YD,IS,CR)

ACPT, SCR2 - GINØ file numbers

ICØR - first available location in MBAMGX

NCØR - last available location in MBAMGX

Z - start of open core

NI - number of independent points

ND - number of dependent points

XD - X location of dependent points

YD - Y location of dependent points

IS - singularity flag

CR - non-dimensionalizing number

4.114.8.41 Subroutine Name: MBCAP

1. Entry Point: MBCAP

2. Purpose: Compute the velocity potential influence coefficients

3. Calling Sequence: CALL MBCAP(NPNI,CAPPNI)

NPNI - number of coefficients computed

CAPPNI - location to store coefficients

MODULE FUNCTIONAL DESCRIPTIONS

4.114.8.42 Subroutine Name: MBBSLJ

1. Entry Point: MBBSLJ
2. Purpose: Compute even-ordered Bessel Functions
3. Calling Sequence: CALL MBBSLJ(ARG,N,BSL)

ARG - input argument

N - order

BSL - storage for answers (length N)

4.114.8.43 Subroutine Name: ZJ

1. Entry Point: ZJ
2. Zero order Bessel Function
3. Calling Sequence: X=ZJ(ARG)

ARG - input argument

4.114.8.44 Subroutine Name: GØ

1. Entry Point: GØ
2. Purpose: Evaluate an Expression $[\psi(\Omega, n_u) - \psi(\Omega, n_l)]$
3. Calling Sequence: ANS=GØ(R,ETAR,ETAL,EKM)

R = ψ

ETAR = n_u

ETAL = n_l

EKM = Ω

4.114.8.45 Subroutine Name: MBDPDH

1. Entry Point: MBDPDH
2. Purpose: Driver for computing and outputting the terms of AJJL for Mach Box Method
3. Calling Sequence: CALL MBDPDH(AJJL,F,DF,F1,DF1,F2,DF2,XWTE,YWTE,PAREA,CAPPNI,DPNITE,
DSS,Q,Q1,Q2,NDN,ND1,NW1,NWN,KTE,KTE1,KTE2,NTE,NNCB,
NNSBD,IW17,IBUF,A)

AJJL - GINØ file number

F - NTE - locations of array for Mach box

NNCB - number or chordwise boxes

FUNCTIONAL MODULE AMG (AERODYNAMIC MATRIX GENERATOR)

NNSBD - number of spanwise boxes

IW17 - GINØ file number

IBUF - pointer to a buffer

A - storage for a row of AJJL

4.114.8.46 Subroutine Name: MBGAE

1. Entry Point: MBGAE
2. Purpose: Final calculation and output for AJJL
3. Calling Sequence: CALL MBGAE(AJJL,IN17,A,F,DF,F1,DF1,F2,DF2,Q,Q1,Q2,MØØØ)
AJJL,IN17 - GINØ file numbers
F to Q2 - locations of Mach box arrays
MØØØ - row number of AJJL

4.114.8.47 Subroutine Name: MBGATE

1. Entry Point: MBGATE
2. Purpose: Compute sum on trailing edge
3. Calling Sequence: CALL MBGATE(NTØTE,DPHITE,N,YWTE,Q,Q1,Q2,KTE,KTE1,KTE2)
NTØTE - number of trailing edge terms
DPHITE - KTE2 - locations of Mach Box arrays

4.114.8.48 Subroutine Name: MBGAW

1. Entry Point: MBGAW
2. Purpose: Compute sum on wing
3. Calling Sequence: CALL MBGAW(BØXL,DPHI,WS,PAW,PAF1,PAF2,Q, Q1,Q2,M,KC,KC1,KC2)
BØXL - box length
DPHI - Q2 - location of Mach Box arrays
M - KC2 - indexes to arrays

4.114.8.49 Complex Function Name: SUMPHI

1. Entry Point: SUMPHI
2. Purpose: Compute sum of $(N \cdot \Delta H)$ on the wing
3. Calling Sequence: SUM=SUMPHI(IXR,IYR,ND1,NDN,CAPPNI,DSS,N,M,ASYM)
IXR,IYR,N,M,ASYM - index and flags
ND1,NDN,CAPPNI,DSS - location of arrays

MODULE FUNCTIONAL DESCRIPTIONS

4.114.8.50 Complex Function Name: TRAILE

1. Entry Point: TRAILE
2. Purpose: Compute sum of $(N \cdot \Delta H)$ on tip
3. Calling Sequence: `SUM=TRAILE(X,J,N,P,M,BØXL)`
J,M,N - pointers
X - values
P - location of array
BØXL - box length

4.114.8.51 Subroutine Name: STPDA

1. Entry Point: STPDA
2. Purpose: Driver for Section one of Strip Theory
3. Calling Sequence: `CALL STPDA(ACPT,AJL,SKJ)`
ACPT,AJL,SKJ - GINØ file numbers

4.114.8.52 Subroutine Name: STPBG

1. Entry Point: STPBG
2. Purpose: Builds two intermediate matrices for Strip Theory calculations (BM and GM)
3. Calling Sequence: `CALL STPBG(BM,GM,NS,BLØC,D,CA,NSIZE)`
BM - storage for BM matrix
GM - storage for GM matrix
NS - number of strips
BLØC - array of semi-chord lengths for strips
D - array of hinge line lengths
CA - array of control surface chords
NSIZE - array of strip types

4.114.8.53 Subroutine Name: STPPHI

1. Entry Point: STPPHI
2. Purpose: Calculate the ϕ functions
3. Calling Sequence: `CALL STPPHI(CA,BLØC,PM,NS)`
CA,BLØC,NS - See STPBG
DM - Storage for ϕ functions

FUNCTIONAL MODULE AMG (AERODYNAMIC MATRIX GENERATOR)

4.114.8.54 Subroutine Name: STPAIC

1. Entry Point: STPAIC
2. Purpose: Calculate and output AJJL for Strip Theory
3. Calling Sequence: CALL STPAIC(BLØC,DY,NSIZE,GAP,BM,GM,PM,NS,CLA,AJJL)
BLØC,NSIZE,NS,BM,GM - See STPBG
GAP - array of control surface gap
PM - See STPPHI
DY - array of Strip widths
CLA - array of lift curve slopes
AJJL - GINØ file numbers

4.114.8.55 Subroutine Name: STPK

1. Entry Point: STPK
2. Purpose: Calculate the K-matrix for Strip Theory
3. Calling Sequence: CALL STPK(EK,N,NSTØP,NØPEN,NSTED,TSR,PM,CR,CI,IM,J1)
EK - modified reduced frequency
N - strip number
NSTØP - strip type
NØPEN - control surface flag
NSTED - reduced frequency flag
TSR - $.5 * \text{GAP} / \text{BLØC}$
PM - ϕ
CR - Theodorsen Function
CI - 0.
IM - k-size
J1 - J-size

4.114.8.56 Subroutine Name: STPBSO

1. Entry Point: STPBSO
2. Purpose: J and Y Bessell functions of order zero
3. Calling Sequence: CALL STPBSO(X,NCØDE,BJO,BYO)
X - input argument

MODULE FUNCTIONAL DESCRIPTIONS

NCODE - flag

BJ0 - J Bessel

BY0 - Y Bessel

4.114.8.57 Subroutine Name: STPBS1

1. Entry Point: STPBS1
2. Purpose: J and Y Bessel Function of first order
3. Calling Sequence: CALL STPBS1(X,NCODE,BJ1,BY1)
X - input argument
NCODE - flag
BJ1 - J Bessel
BY1 - Y Bessel

4.114.8.58 Subroutine Name: STPPT2

1. Entry Point: STPPT2
2. Purpose: Output D1JK and D2JK for Strip Theory, Mach Box and Piston Theory
3. Calling Sequence: CALL STPPT2(ACPT,D1JK,D2JK)
ACPT,D1JK,D2JK - GINØ file numbers

4.114.8.59 Subroutine Name: PSTAMG

1. Entry Point: PSTAMG
2. Purpose: Driver for Section one of Piston Theory
3. Calling Sequence: CALL PSTAMG(ACPT,AJJL,SKJ)
ACPT,AJJL,SKJ - GINØ file numbers

4.114.8.60 Subroutine Name: PSTA

1. Entry Point: PSTA
2. Purpose: Calculate and output AJJL for Piston Theory
3. Calling Sequence: CALL PSTA(DELT,BI,CA,ALPH,THI,AJJL)
DELT - array of strip width
BI - array of semi-chord lengths for strips
CA - array of chord lengths of each strip

FUNCTIONAL MODULE AMG (AERODYNAMIC MATRIX GENERATOR)

ALPH - Alpha array (angle of attack)

THI - Theta array (thickness ratio)

AJL - GINØ file number

4.114.9 Design Requirements

For Section one, four buffers are allocated at the bottom of core. For Section two, three buffers are allocated at the bottom of core. Each method may have its own open core common block but they must not overlap these buffers.

4.114.9.1 Common Blocks

AMGMN - Doublet Lattice without Bodies Communication

Words

1-7	MCB	- Trailer for AJL	
8	NRØW	- Last row number output for any method on AJL	
9	ND	- Y-symmetry flag	} 1 record of AERØ Data Block
10	NE	- Z-symmetry flag	
11	REFC	- Reference card	
12	FMACH	- Mach number (M)	} Pairs from 2 record of AERØ Data Block
13	RFK	- Reduced frequency	
14-20	TSKJ	- Trailer for SKJ	
21	ISK	- Row number to start building on SKJ	
22	NSK	- Last row number output for any method on SKJ	

AMGP2 - Section Two Communication

Words

1-7	TW1JK	- trailer for D1JK
8-14	TW2JK	- trailer for D2JK

DLCØM - Doublet Lattice without Bodies Communication

Words

1	NP	- number of panels
2	NSTRIP	- number of strips

MODULE FUNCTIONAL DESCRIPTIONS

DLCOM (Cont'd.)

Words

- | | | |
|----|---------|--|
| 3 | NTP | - number of boxes |
| 4 | F | - fraction of box chord |
| 5 | NJJ | - NJ (Input parameter) |
| 6 | NEXT | - first location of open core available after allocation |
| 7 | LENGTH- | number of boxes along longest panel (from NCARAY) |
| 8 | INC | - pointer to NCARAY array |
| 9 | INB | - pointer to NBARAY array |
| 10 | IYS | - pointer to YS array |
| 11 | IZS | - pointer to ZS array |
| 12 | IEE | - Pointer to EE array |
| 13 | ISG | - pointer to SG array |
| 14 | ICG | - pointer to CG array |
| 15 | IXIC | - pointer to XIC array |
| 16 | IDELX | - pointer to DELX array |
| 17 | IXLAM | - pointer to XLAM array |
| 18 | IDT | - complex pointer to storage for column of AJJL |

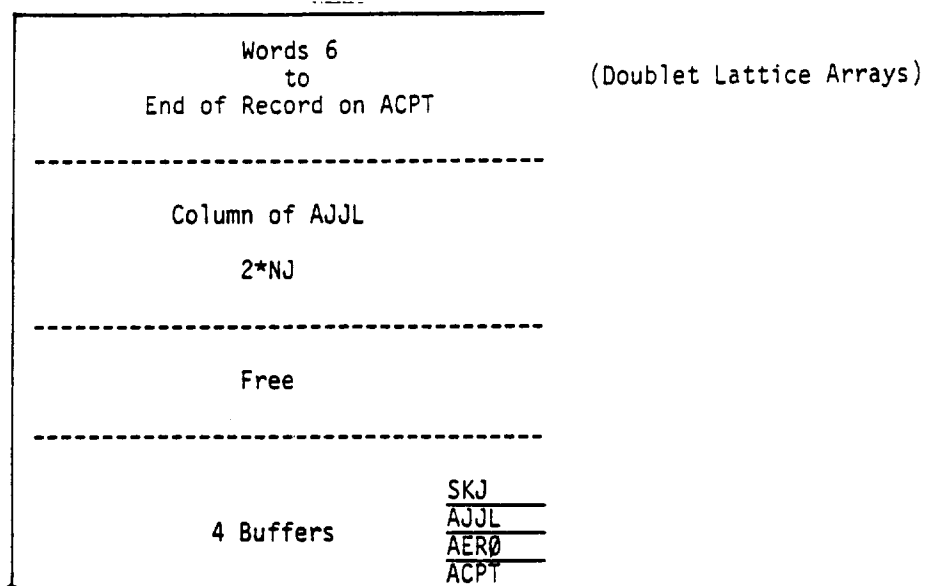
DLM - Both Doublet Lattice methods

Words

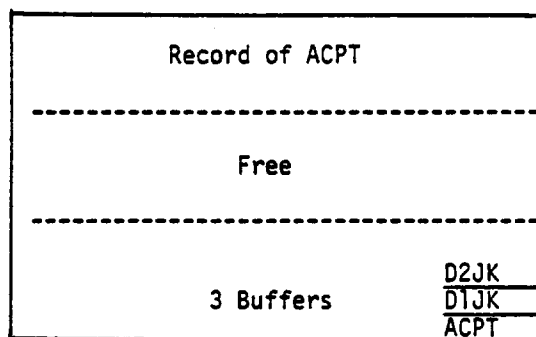
- | | | |
|---|----------------|---|
| 1 | K10 | - Planar part of steady contribution to the kernel |
| 2 | K20 | - Nonplanar part of steady contribution to the kernel |
| 3 | K1RT1 | } Unsteady parts of modified kernel |
| 4 | K1IT1 | |
| 5 | K2RT2P- | |
| 6 | K2IT2P- | |
| 7 | K10T1 - K10*T1 | } T1 and T2 are defined under TKER |
| 8 | K20T2P- K20*T2 | |

FUNCTIONAL MODULE AMG (AERODYNAMIC MATRIX GENERATOR)

DLAXX - Open Core for Doublet Lattice without Bodies



DLP2X - Open core for Section two



KDS - Both Doublet Lattice Methods

Words

- 1 IHD - 0 = total kernel, 1 = incremental part only
- 2 KD1R - real part of k_1
- 3 KD1I - imaginary part of k_1
- 4 KD2R - real part of k_2
- 5 KD2I - imaginary part of k_2

MODULE FUNCTIONAL DESCRIPTIONS

DLBDY - Doublet Lattice with Bodies Communication

Words

1-12 Words 2-13 of ACPT record

13-51 pointers into DLBXX for arrays on ACPT

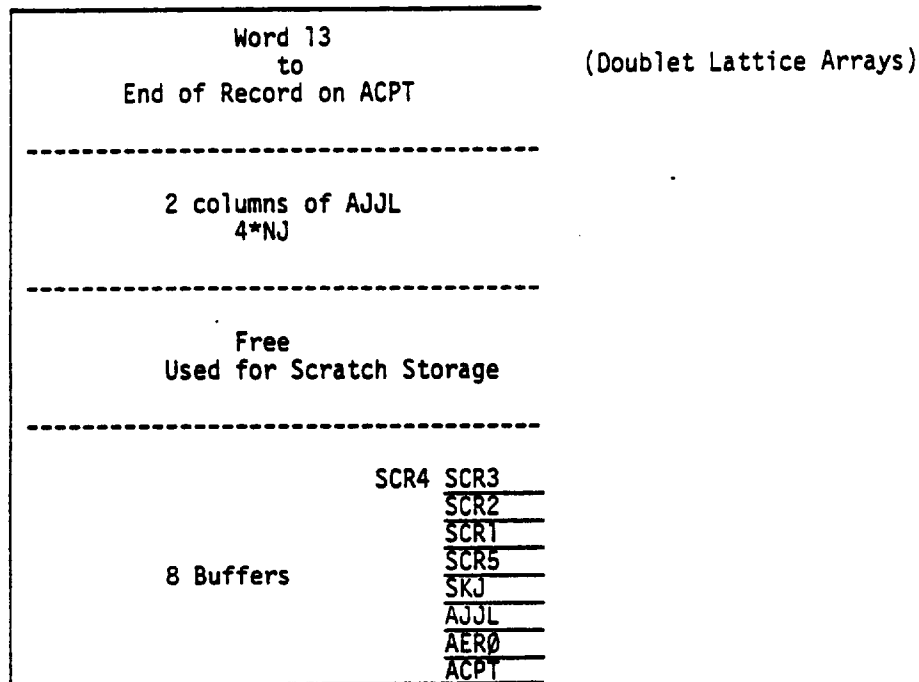
52 ECØRE - end of core in DLBXX

53 NEXT - next available location in DLBXX

54-58 SCR1-SCR5 - GINØ file numbers for scratch files

59 NTBE - number of columns to add to AJJL

DLBXX - Open core for Doublet Lattice with Bodies



MBØXA - Mach Box Wing Definitions

Words

1-12 X - X locations of wing

13-24 Y - Y locations of wing

25-34 TANG - Tangents of wing sweep angles

35-44 ANG - Sweep angles of wing

45-54 CØTANG- Cotangents of wing sweep angles

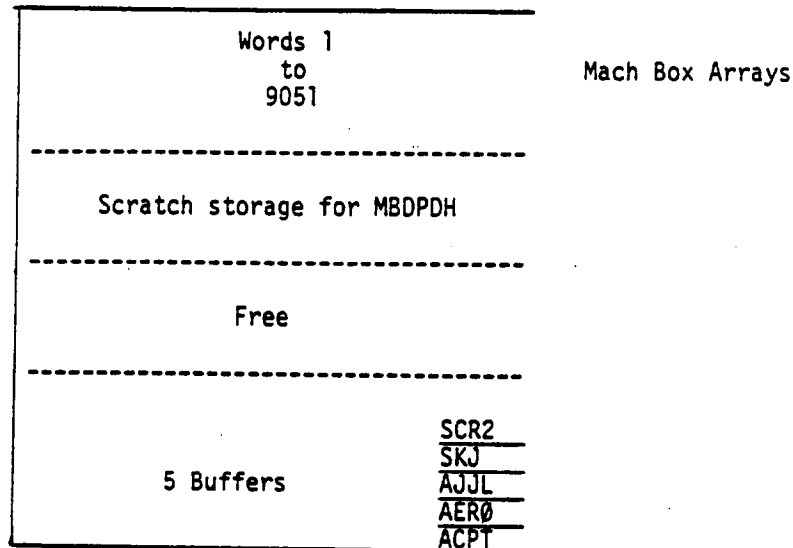
FUNCTIONAL MODULE AMG (AERODYNAMIC MATRIX GENERATOR)

MBØXC - Mach Box Communications

Words

- 1-9 Words 2-10 of ACPT record
- 10-30 Intercommunication between Mach Box subroutines

MBAMGX - Open core for Mach Box Method



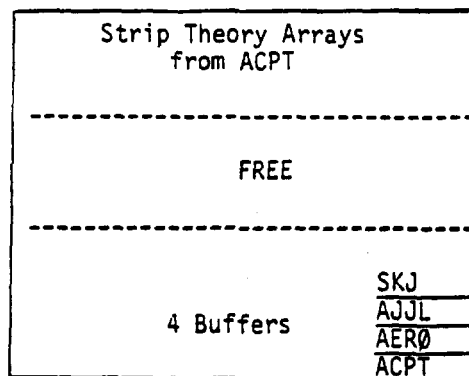
STRIPC - Strip Theory Communications

Words

- 1 NS - number of strips
- 2 BREF - reference chord/2.0
- 3 CLAM - cosine of sweep angle
- 4 FM - Mach number
- 5 NCIRC - Theodorsen function selection
- 6 NNCIRC - NCIRC+1
- 7 EKR - reduced frequency
- 8 Not Used
- 9-12 BB(u) - b's for approximate function
- 13-16 BETA(u) - B's for approximate function
- 17-48 EKM(u,u) complex - storage for STPK output (k matrix)

MODULE FUNCTIONAL DESCRIPTIONS

STRIPX - Strip Theory Open Core

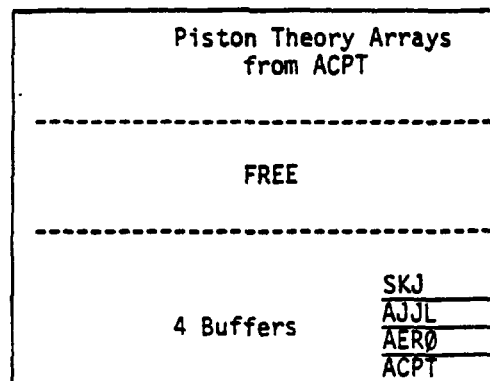


PSTØNC - Piston Theory Communication

Words

1-9 Words 2-10 of ACPT record

PSTØNX - Piston Theory Open Core



4.114.10 Diagnostic Messages

System fatal messages 3001, 3002, 3003, 3007, 3008 and (10) 3061. User fatal messages 2264 and 2265.

FUNCTIONAL MODULE AMG (AERODYNAMIC MATRIX GENERATOR)

4.114.11 References

Most of the equations and code for the Doublet Lattice Method were taken from

- (1) Giesing, J.P., Kalman, T.P., Rodden, W.P., "Application of the Doublet-Lattice Method and the Method of Images to Lifting-Surface/Body Interference," AFFDL-TR-71-5, Part II, Vol. 1, April 1972.

Most of the equations and code for the Mach Box Method were taken from

- (2) Donato, V.W., Huhn, C.R., Jr., "Supersonic Unsteady Aerodynamics for Wings with Trailing Edge Control Surfaces and Folded Tips," AFFDL-TR-68-30, August 1968.

Most of the equations and code for the Strip Theory Method were taken from

- (3) Albano, E., "Strip Theory Aerodynamic Influence Coefficients for Wings with Aerodynamically Balanced Control Surfaces," Northrop Corporation, Moran Division Report NOR 68-125, August 1968.

Most of the equations and code for the Piston Theory Method were taken from

- (4) Rodden, W.P., Forkos, E.F., Malcom, H.A., and Kliszcwski, A.M., "Aerodynamic Influence Coefficients from Piston Theory: Analytical Development and Computational Procedure," Space Systems Division, United States Air Force Report No. TDR-169 (3230-11)TN-2, August 1962.

FUNCTIONAL MODULE AMP (AERODYNAMIC MATRIX PROCESSOR)

4.115 FUNCTIONAL MODULE AMP (AERODYNAMIC MATRIX PROCESSOR)

4.115.1 Entry Point:

AMP

4.115.2 Purpose

The purpose of this module is to produce "modal" aerodynamic matrices. This requires the combination of matrices from four sources.

1. The aerodynamic matrices for aerodynamic cells, produced by the Aerodynamic Matrix Generator (AMG) module.
2. The interpolation from the structure to the aerodynamic cells, produced by the Geometry Interpolator (GI) module.
3. The modes of the structure, produced by the Real Eigenvalue Analysis (READ) module, and selected by GKAM.
4. The matrix of downwashes due to "extra" points, which may be supplied by the user via module INPUTT2. These extra points in NASTRAN are used for control systems and other special effects.

4.115.3 DMAP Calling Sequence

```
AMP  AJJL,SKJ,D1JK,D2JK,GTKA,PHIDH,D1JE,D2JE,USETD,AERØ/QHHL,QKHL,QHJL/V,N,NØUE/V,N,XQHHL/  
      V,N,GUSTAERØ $
```

4.115.4 Input Data Blocks

AJJL	Aerodynamic influence matrix list
SKJ	Integration matrix
D1JK	Real part of downwash matrix
D2JK	Complex part of downwash matrix
GTKA	Aerodynamic transformation matrix k-set to a-set
PHIDH	Transformation between modal and physical coordinates
D1JE	Downwash factors due to extra points; real
D2JE	Downwash factors due to extra points; complex
USETD	Displacement sets definition - dynamics

MODULE FUNCTIONAL DESCRIPTIONS

AERØ Aerodynamic matrix generation data

Notes:

1. AJJL, SKJ, DIJK, D2JK, GTKA, PHIDH, USETD, and AERØ may not be purged.
2. DIJE and D2JE are used only if NØUE > 0. Even in this case they may be purged.

4.115.5 Output Data Blocks

QHHL -- Aerodynamic matrix list - h-set

QKHL -- Aerodynamic transformation matrix between hand k sets

QHJL -- Aerodynamic transformation matrix between j and k sets

Notes:

1. QHHL, QKHL, and QHJL are matrix lists - one submatrix for each (m,k) pair.
2. If QHHL, QKHL, and QHJL are present before the module begins (APPEND on restart) and XQHHL = -1, the old data needed is read from these data blocks.

4.115.6 Parameters

NØUE -- Integer, input, no default. The number of extra points.

XQHHL - Integer, input/output, no default. If +1, the data found on appended data blocks must be discarded. If -1, it can be used. AMP sets XQHHL to -1 on exit.

GUSTAERØ - Integer, input, default = 0. If, and only if, GUSTAERØ < 0, AMP will compute QHJL.

4.115.7 Method

There are several important features which must be kept in mind.

1. In general, the input and output matrices may depend upon the aerodynamic parameters k (reduced frequency) and m (Mach number). A set of matrices (called a list) are processed in one pass through the module.
2. Special code will be introduced for restart. This is required for Doublet Lattice solutions where matrix solution time may be long. This will allow the addition (or deletion) of (m,k) pairs without redecomposing the downwash matrix.
3. An output, Q_{kh} , relating aerodynamic pressures to modal coordinates may be required for use in a data reduction module. This output will not be used in Phase 1; hence it will be purged from the calling sequence. The matrix of generalized forces, Q_{hh} , may be purged, if only data reduction is desired.

FUNCTIONAL MODULE AMP (AERODYNAMIC MATRIX PROCESSOR)

The flow chart is shown in Figure 1. The basic loop is to write out matrices for the list of (m,k) pairs found on the AERØ data block. The source of these matrices is normally the input. In the case of a restart which involves only changes in the (m,k) pairs, special code is provided to avoid recalculation; and the matrices are found on the output data block which is declared APPEND. This occurs when XQHHL = -1.

4.115.7.1 Subroutine AMPA

Put the output m,k list in core. This list is found in the second record of the AERØ data block. The index I will be used to index down this list of pairs. IMAX is the number of pairs.

Check to see if the output data blocks QKHL, QHHL, and QHJL exist and are valid. If they do, then this is a restart. These must be copied onto scratch files. Their (m,k) lists are put in core. Build a scenario file which lists the (m,k) pair, the AJJL column associated with this (m,k) pair, and the corresponding QHHL column.

4.115.7.2 Subroutine AMPB

Calculate the D_{jh} matrices. The superscript (1) is for the real part and (2) is for the imaginary part.

$$\begin{aligned} [\phi_{ai}] &= \text{Partition of } [\phi_{dh}] \quad , \\ [G_{ki}] &= [G_{ka}^T]^T [\phi_{ai}] \quad . \end{aligned} \tag{1}$$

$[G_{ki}]$ may be needed again later to calculate Q_{ih} .

$$[D_{ji}^{(1)}] = [D_{jk}^{(1)}]^T [G_{ki}] \quad . \tag{2}$$

$$[D_{ji}^{(2)}] = [D_{jk}^{(2)}]^T [G_{ki}] \quad . \tag{3}$$

$$[D_{jh}^{(1)}] = \text{Merge } [D_{ji}^{(1)}] D_{je}^{(1)} \quad . \tag{4}$$

$$[D_{jh}^{(2)}] = \text{Merge } [D_{ji}^{(2)}] D_{je}^{(2)} \quad . \tag{5}$$

If the input data blocks are purged, $D_{je}^{(1,2)}$ is zero. Start a loop with I = 0. Check the time left.

MODULE FUNCTIONAL DESCRIPTIONS

4.115.7.3 Subroutine AMPC

Calculate (or find) Q_{jh} if it is needed. It will be needed if either (a) Q_{kh} is to be output, or (b) Q_{hh} is to be output and is not found on the scratch file. The Q_{kh} and Q_{hh} are not to be output only when their output data blocks are purged. If Q_{kh} can be found on a scratch file, get it from there; otherwise, it must be calculated. First, check to see if $D_{jh}(k)$ has been calculated for the present k . If not, find it by

$$[D_{jh}] = [D_{jh}^{(1)}] + i k [D_{jh}^{(2)}] \quad , \quad (6)$$

and save for possible later use. Next, solve for Q_{jh} . The algebra included here will be theory dependent. The header record of AJJL will specify aerodynamic groups (see Section 4.115.7.5). Retrieve the submatrix $[A_{jj}]$ from AJJL. If there is more than one group, D_{jh} must be unpacked into row groups. For each group, solve for $[Q_{jh}]$, then pack the groups. For Doublet Lattice method, and the Double Lattice method with slender bodies,

$$[Q_{jh}]_{\text{group}} = [A_{jj}^T]_{\text{group}}^{-1} [D_{jh}]_{\text{group}} \quad . \quad (7)$$

For other methods,

$$[Q_{jh}]_{\text{group}} = [A_{jj}] [D_{jh}]_{\text{group}} \quad (7a)$$

4.115.7.4 Subroutine AMPD

Calculate (or find) $[Q_{hh}]$ and $[Q_{kh}]$ if they are needed. They will be needed unless the output data blocks are purged. If $[Q_{hh}]$ can be found on a scratch file, get it there, otherwise, it must be calculated. If it must be calculated $[Q_{jh}]$ will be available. To compute $[Q_{hh}]$

Where S_{kj} is a matrix list for (m,k) ,

$$[Q_{kh}] = [S_{kj}] [Q_{jh}] \quad , \quad (8)$$

$[Q_{kh}]$ is copied onto QKHL.

$$[Q_{ih}] = [G_{ki}^T] [Q_{kh}] \quad , \quad (9)$$

$$[Q_{hh}] = \text{Merge} \begin{bmatrix} Q_{ih} \\ Q_{eh} \end{bmatrix} \quad , \quad (10)$$

where $[Q_{eh}]$ is zero. Note that this requires only an update of $[Q_{ih}]$'s trailer.

FUNCTIONAL MODULE AMP (AERODYNAMIC MATRIX PROCESSOR)

Check the time. If $[Q_{jh}]$ and $[Q_{hh}]$ were calculated (rather than found), then the time per calculation can be found. If the time per calculation is known and it is not enough (with a 10% margin), no more loops should be attempted.

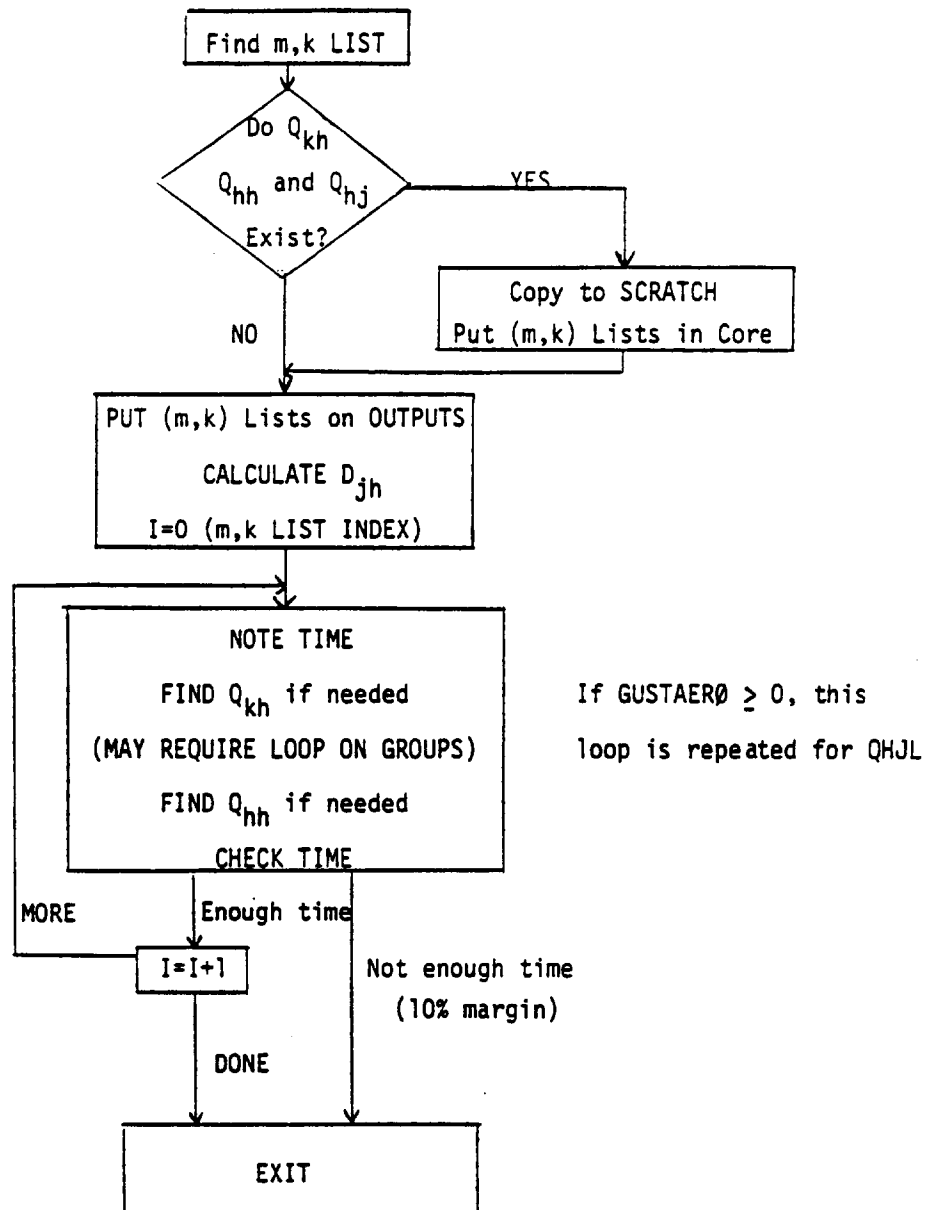


Figure 1. AMP Module Flow.

Repeat the loop (subroutines AMPC and AMPD) for additional values of I until the job is done.

MODULE FUNCTIONAL DESCRIPTIONS

If $GUSTAER0 \geq 0$, the following equations are evaluated:

Partition PHIDH (Real)

$$\begin{array}{c} \xleftarrow{h} \\ \uparrow d \\ \left[\begin{array}{c} \text{PHIDH} \end{array} \right] \\ \downarrow \end{array} \longrightarrow \begin{array}{c} \xleftarrow{h} \\ \uparrow a \\ \left[\begin{array}{c} \text{PHIAH} \end{array} \right] \\ \downarrow e \end{array} \quad (11)$$

Multiply (TRANSPØSE) (Real)

$$\begin{array}{c} \xleftarrow{k} \\ \uparrow a \\ \left[\begin{array}{c} \text{GIKA} \end{array} \right]^T \\ \downarrow \end{array} \cdot \begin{array}{c} \xleftarrow{h} \\ \uparrow a \\ \left[\begin{array}{c} \text{PHIAH} \end{array} \right] \\ \downarrow \end{array} \longrightarrow \begin{array}{c} \xleftarrow{h} \\ \uparrow k \\ \left[\begin{array}{c} \text{GKH} \end{array} \right] \\ \downarrow \end{array} \quad (12)$$

Start loop on reduced (m,k) pairs (use all).

Multiply (TRANSPØSE) (S is complex)

$$\begin{array}{c} \xleftarrow{j} \\ \uparrow k \\ \left[\begin{array}{c} \text{SKJ}(k) \end{array} \right]^T \\ \downarrow \end{array} \cdot \begin{array}{c} \xleftarrow{h} \\ \left[\begin{array}{c} \text{GKH} \end{array} \right] \end{array} \longrightarrow \begin{array}{c} \xleftarrow{h} \\ \uparrow j \\ \left[\begin{array}{c} \text{S}(k) \end{array} \right] \\ \downarrow \end{array} \quad (13)$$

Partition into Groups - (1) = Doublet Lattice, (2) = non-Doublet Lattice

$$\left[\begin{array}{c} \text{S}(k) \end{array} \right] \longrightarrow \begin{array}{c} \xleftarrow{h} \\ \uparrow j_1 \\ \left[\begin{array}{c} \text{S}(1) \\ \text{-----} \\ \text{S}(2) \end{array} \right] \\ \downarrow j_2 \end{array} \quad (14)$$

Solve each group R_{jh} :

a. Doublet Lattice group

$$A_{jj}(\text{group}) R_{jh}(1) = S(1) \quad (15)$$

b. Non-Doublet Lattice group

$$R_{jh}(2) = A_{jj}^T S(2) \quad (16)$$

FUNCTIONAL MODULE AMP (AERODYNAMIC MATRIX PROCESSOR)

Merge Results

$$\begin{bmatrix} R_{jh(1)} \\ R_{jh(2)} \end{bmatrix} \longrightarrow \begin{matrix} \xleftarrow{h} \xrightarrow{} \\ \uparrow \downarrow j \\ \begin{bmatrix} R_{jh} \end{bmatrix} \end{matrix} \quad (17)$$

Append R_{hj}

$$\left[\begin{array}{c|c} QJHL & Q_{jh(k)} \end{array} \right] \longrightarrow \left[QHJL \right] \quad (18)$$

Repeat the last five steps for (m,k) pairs.

MODULE FUNCTIONAL DESCRIPTIONS

4.115.7.5 Matrix List Data Blocks

The matrix list data block is a special data block used for NASTRAN aeroelastic calculations. It is used to store a series of matrices. The matrices in the list will depend upon two parameters. The format is similar to that of a matrix. If there are NMK matrices, each with NRØW rows and NCØL columns, then it will be stored like a matrix with NRØW rows, and NMK times NCØL columns. The matrix for the first parameter pair is stored in the first NCØL columns. The matrix for other parameter pairs is then added on at the end, one at a time.

A special header record is written which contains the following information:

<u>Word</u>	<u>Value</u>
1-2	The name
3	NCØL, for the individual matrices
4	NMK
5-(4+2NMK)	M(I), K(I), I=1, NMK
+	Other information

If a single matrix exists, it can be read as a normal NASTRAN matrix. It is possible that the matrix list was not completed by the generating module. The number of columns (found in the trailer) divided by NCØL should be an integer. This should be equal to NMK. If it is less than NMK, it is the actual number of matrices on the list. For the AJJL, there is additional information in the header.

<u>Word</u>	<u>Value</u>
(6+2NMK)	NGP, number of uncoupled aerodynamic groups
(7+2NMK) -	KT(N), NJ(N), N=1, NK(N) to NGPT, the theory
(6+2NMK+3NGP)	identifier and the number of U_j degrees of freedom associated with this group. $\sum NJ = NCØL$.

The matrix AJJL might look like (1 is the identifier for Doublet Lattice theory):

FUNCTIONAL MODULE AMP (AERODYNAMIC MATRIX PROCESSOR)

NCØL = 7

NMK = 3

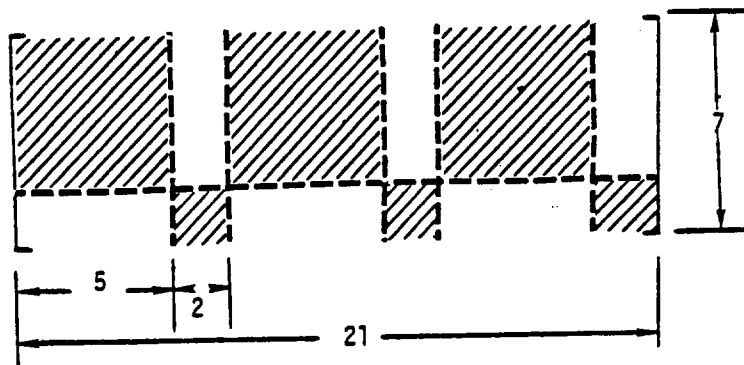
NGP = 2

KT(1) = 1

KT(2) = 1

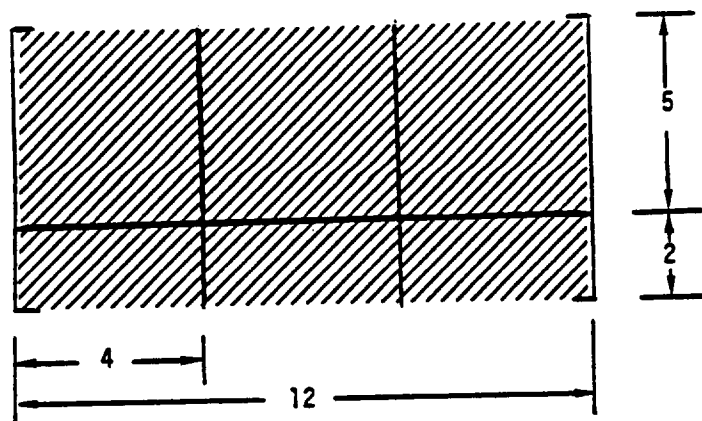
NJ(1) = 5

NJ(2) = 2



The shaded areas
may be nonzero.

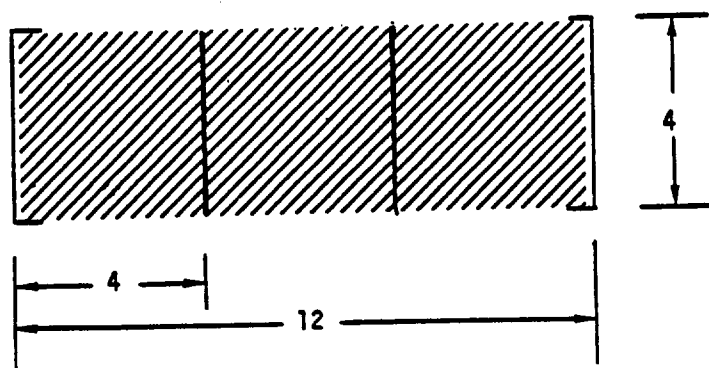
The output QKHL might look like (for 3 modes):



NCØL = 4 (number of modes
+ extra points)

NMK = 3 (the M(I), K(I) are
found from the AERØ
data block)

The output QHHL might look like:



NCØL = 4

NMK = 3

4.115.8 Subroutines

Numerous utility subroutines are used by the functional phases as shown below.

<u>AMPA</u>	<u>AMPB</u>	<u>AMPC</u>	<u>AMPD</u>	<u>AMPE</u>	<u>AMPF</u>
CYCT2B	CALCV	CYCT2B	CYCT2B	CYCT2B	CYCT2B
	SSG2B	SSG2C	SSG2B	SSG2B	SSG2B
	MERGED	CFACTR	SKPREC	SSG2A	CFACTR
	PARTN	CFBSØR		SKPREC	CFBSØR
		FILSWI			FILSWI
		TRANP1			SKPREC

4.115.8.1 Subroutine Name: AMPA

1. Entry Point: AMPA

2. Purpose: To provide a scenario for later phases and to prepare for use of the appended output files.

3. Calling Sequence: CALL AMPA (AERØ, QJHL, QHHL, AJJL, QHHLØ, QJHLØ, INDEX, IMAX, IANY)

AERØ, QJHL, QHHL, and AJJL are the GINØ file numbers of their respective data blocks.

QHHLØ and QJHLØ are the GINØ file numbers of two scratch files to hold valid submatrices from QHHL and QJHL on restart.

INDEX is the GINØ file number of the scenario data block. Its contents are as follows:

<u>Record No.</u>	<u>Word</u>	<u>Contents</u>
0	1	Header
1	1	M column number
	2	K column number
	3	AJJL column number
	4	QHHLØ column number (0 implies recompute)
	⋮	
	⋮	
	IMAX	

FUNCTIONAL MODUME AMP (AERODYNAMIC MATRIX PROCESSOR)

IMAX is the total number of (m,k) pairs on output.

IANY is the flag for necessity to compute at least 1 QJH or QHH (0 implies compute -- 1 implies all retrieved modes).

4. Common Blocks

/AMPCØM/NCØLJ,NSUB,XM,XK,AJJCØL,QHHCØL,NGP,NGPD(2,30),MCBQHH(7),MCBQKH(7),NCØLH,
IDJH,MCBRJH(7)

NCØLJ - Number of columns in a submatrix of AJJL

NSUB - Number of submatrices in AJJL

XM - Current M value

XK - Current K value

AJJCØL - Current column number of AJJL

QHHCØL - Current column number in QHHLØ (a zero value implies compute a new QHH)

NPG - Number of groups

NGPD - Two words for each group - Theory ID (1 = D.L.) - Number of columns of AJJ
belonging to this group

MCBQHH - Matrix control block for QHHL

MCBQKH - Matrix control block for QJHL

NCØLH - Number of H points

IDJH - Flag for change in k value in (m,k) pair

MCBRJH - Matrix control block for QJHL

4.115.8.2 Subroutine Name: AMPB

1. Entry Point: AMPB

2. Purpose: To compute GKI and the DJH1 and DJH2.

3. Calling Sequence: CALL AMPB (PHIDH,GTKA,D1JK,D2JK,D1JE,D2JE,USETD,DJH1,DJH2,GKI,SCR1,
SCR2,SCR3)

All inputs are GINØ file numbers.

MODULE FUNCTIONAL DESCRIPTIONS

4.115.8.3 Subroutine Name: AMPB1

1. Entry Point: AMPB1
2. Purpose: To build a partitioning vector which will add a given number of columns to another matrix.
3. Calling Sequence: CALL AMPB1 (IPVECT,NCØL1,NCØL2)

FUNCTIONAL MODULE AMP (AERODYNAMIC MATRIX PROCESSOR)

THIS PAGE HAS BEEN LEFT BLANK INTENTIONALLY.

MODULE FUNCTIONAL DESCRIPTIONS

IPVECT - the GINØ file number on which the partitioning matrix will be built.

NCØL1 - the number of columns in first matrix.

NCØL2 - the number of columns in second matrix (to add onto the first matrix).

4.115.8.4 Subroutine Name: AMPB2

1. Entry Point: AMPB2

2. Purpose: This routine is a general driver for PARTN.

3. Calling Sequence: CALL AMPB2 (A,A11,A12,A21,A22,RP,CP,N1,N2)

A, A11, A12, A21, A22, RP, and CP are the GINØ file numbers of the matrices supplied to PARTN.

$$[A] = \left\{ \begin{matrix} CP \end{matrix} \right\} \begin{bmatrix} A11 & A12 \\ A21 & A22 \end{bmatrix}$$

If any partition is not desired, set its file name to zero.

N1 and N2 are the number of rows of RP and CP respectively. These are used only if RP or CP = 0. A, RP and CP must have matrix trailers. Trailers will be written on all existing outputs.

4.115.8.5 Subroutine Name: AMPC

1. Entry Point: AMPC

2. Purpose: To compute (or retrieve) QJH and to form QJHL (if not purged).

3. Calling Sequence: CALL AMPC (DJH1,DJH2,DJH,AJJL,QJHL,QJHØ,QJHUA,SCR1,SCR2,SCR3,SCR4,SCR5,SCR6)

DJH1,DJH2,DJH,AJJL and QJHL are the GINØ file numbers (GFN) of their respective data blocks

QJHØ is the GFN of a data block containing old QJH's from restart

QJHUA is the GFN of a data block containing the current QJH

SCR1 - SCR6 are the GINØ file numbers of six scratch files

4.115.8.6 Subroutine Name: AMPC1

1. Entry Point: AMPC1

MODULE FUNCTIONAL DESCRIPTIONS

2. Purpose: To copy columns of one open matrix to another matrix.

3. Calling Sequence: CALL AMPC1 (INPUT,ØUTPUT,NCØL,IZ,MCB)

INPUT = GINØ file number of the input matrix

ØUTPUT = GINØ file number of the output matrix

NCØL = the number of columns to copy

IZ = open core

MCB = matrix control block for ØUTPUT.

/UNPAKX/ and /PACKX/ control AMPC1.

4. Design Requirements: Both matrices must be opened and properly positioned. No trailers are written. Both matrices are left open.

4.115.8.7 Subroutine Name: AMPC2

1. Entry Point: AMPC2

2. Purpose: To copy each column of the INPUT file onto the bottom of each column of the ØUTPUT file.

3. Calling Sequence: CALL AMPC2 (INPUT,ØUTPUT,SCR1)

INPUT, ØUTPUT, and SCR1 are the GINØ file numbers of their respective data blocks.

4. Method: On the first entry, INPUT and ØUTPUT are switched. On subsequent entries ØUTPUT and SCR1 are switched and read together, one column at a time to produce ØUTPUT.

4.115.8.8 Subroutine Name: AMPD

1. Entry Point: AMPD

2. Purpose: To compute or retrieve QHH and to write QHHL.

3. Calling Sequence: CALL AMPD (QJHUA,QHHØ,SKJ,GKI,QHHL,SCR1,SCR3,SCR4)

All inputs are the GINØ file numbers of their respective data blocks.

FUNCTIONAL MODULE AMP (AERODYNAMIC MATRIX PROCESSOR)

4.115.8.9 Subroutine Name: AMPE

1. Entry Point: AMPE
2. Purpose: To compute GKH
3. Calling Sequence: CALL AMPE(PHIDH,GTGA,GKH,SCR1,SCR2,USETA)

All inputs are the GINØ file numbers of their respective data blocks.

4. Method: AMPE calls CALCV and SSG2A to partition PHIDH. It then calls SSG2B to compute GKH.

4.115.8.10 Subroutine Name: AMPF

1. Entry Point: AMPF
2. Purpose: To solve for QHJL
3. Calling Sequence: CALL AMPF(SKJ,GKH,AJJL,QHJL,PLAN,IMAX,SCR1,SCR2,SCR3,SCR4,SCR5,SCR6,
SCR7,SCR8,SCR9,SCR10)

SKH, GKH, AJJL, QHJL, PLAN, SCR1-SCR10 are GINØ file numbers of their respective data blocks.

IMAX is the number of m,k pairs.

4.115.9 Design Requirements

1. AMP requires 14 scratch files. These files are used as follows:

MODULE FUNCTIONAL DESCRIPTIONS

THIS PAGE HAS BEEN LEFT BLANK INTENTIONALLY.

FUNCTIONAL MODULE AMP (AERODYNAMIC MATRIX PROCESSOR)

<u>NAME</u>	<u>DATA BLOCK</u>	<u>COMPUTED BY</u>	<u>USED BY</u>
SCR1	Old QHHL (QHHØ)	AMPA	A,D
SCR2	Old QKHL (QJHØ)	AMPA	A,C
SCR3	Index of work to do (INDEX)	AMPA	A,DRIVER
SCR4	DJH1	AMPB	B,C
SCR5	DJH2	AMPB	B,C
SCR6	GKI	AMPB	B,D
SCR7	DJH	AMPB	C,C
SCR8	QJHUA	AMPC	C,D
SCR9	Scratch File		B,C,D
SCR10	Scratch File		B,C,D
SCR11	Scratch File		B,C,D
SCR12	Scratch File		C,D
SCR13	Scratch File		C
SCR14	Scratch File		C

2. Open Core:

<u>ROUTINE</u>	<u>OPEN CORE</u>	<u>FUNCTION</u>
AMP	AMPB2X	Buffer
AMPA	AMPA1X	See layout
AMPB	AMPB2X	CALCV
AMPB1	AMPB1X	Buffer
AMPB2	AMPB2X	PARTN
AMPC	AMPC1X	Buffer, CYCT2B, AMPC1
AMPC2	AMPC1X	Buffer
AMPD	AMPD1X	Buffer, CYCT2B
AMPE	AMPEX	Buffer, CYCT2B
AMPF	AMPFX	Buffer, CYCT2B, AMPC1

FUNCTIONAL MODULE AMP (AERODYNAMIC MATRIX PROCESSOR)

Open core AMPA1X is laid out as follows:

<u>Contents</u>	<u>Length</u>	<u>Pointer</u>
NCØLJ	1	
NSUB	1	
m-k pairs from AJJL Header	2*NSUB	IAJJL
m-k pairs from AERØ	2*IMAX	IAERØ
NCØLH	1	
NØH	1	
m-k pairs from old QHH	2*NQHH	IQHH
2 Buffers		

4.115.10 Diagnostic Messages

The following messages may occur: 3045, 3001, 3002, 3003, 3008, 3008

3007 occurs when a theory is used which AMP does not understand; 3045 occurs when insufficient time remains to compute another m-k pair.

MODULE FUNCTIONAL DESCRIPTIONS

The loop between AMPC and AMPD would require much overlaying. Thus, AMP currently is a single overlay chain.

4.115.10 Diagnostic Messages

The following messages may occur: 3045, 3001, 3002, 3003, 3008, 3007

3007 occurs when a theory is used which AMP does not understand; 3045 occurs when insufficient time remains to compute another m-k pair.

FUNCTIONAL MODULE FA1 (FLUTTER ANALYSIS - PHASE 1)

4.116 FUNCTIONAL MODULE FA1 (FLUTTER ANALYSIS - PHASE 1)

4.116.1 Entry Point: FA1

4.116.2 Purpose

To prepare the modal matrices MXHH, BXHH and KXHH for the K method of eigenvalue analysis or to do the eigenvalue analysis for the KE or PK method.

4.116.3 DMAP Calling Sequence

FA1 KHH,BHH,MHH,QHHL,CASECC,FLIST/FSAVE,KXHH,BXHH,MXHH/V,N,FLØØP/
 V,N,TSTART/C,N,NØCEAD \$

4.116.4 Input Data Blocks

KHH - Modal stiffness matrix - h-set
BHH - Modal damping matrix - h-set
MHH - Modal mass matrix - h-set
QHHL - Aerodynamic matrix list - h-set
CASECC - Case control data block
FLIST - Flutter control table

Note: BHH may be purged for the K and PK methods. BHH must be purged for the KE method.

4.116.5 Output Data Blocks

FSAVE - Flutter storage save table or answer table
KXHH - Total modal stiffness matrix - h-set or eigenvector matrix
BXHH - Total modal damping matrix - h-set or eigenvalue table
MXHH - Total modal mass matrix - h-set

Note: If BHH is purged, BXHH may be purged for the K method. For the KE and PK methods, BXHH may not be purged.

4.116.6 Parameters

FLØØP - Input/Output - integer. Zero upon initial entry; Loop count at exit except for the last loop when FLØØP = -1.
TSTART - Output - integer. CPU clock time at entry to module.
NØCEAD - Output - integer - flag whether to perform CEAD or not. 1 = yes, -1 = no.

4.116.7 Method

Module FA1 is broken into two parts. Part one is the initial entry into the module, where the flutter data cards are read and records 0, 1, 2 and 3 of FSAVE are built. Part two builds the output matrices for eigenvalue extraction depending upon the flutter method selected.

The flow for part one is as follows: Case Control is read into core and the Set ID for the selected flutter data card is found. The table FLIST is opened and the AERØ data card is read and saved, all the FLFACT data cards are read into core, and the proper FLUTTER data card is found and saved.

Then the flutter analysis method selected is determined and the matrix interpolation method is determined. Record 0 of FSAVE is then written.

The selected FLFACT lists for Mach number (m), reduced frequency (k) and density ratios (ρ) are found, and record 1 of FSAVE is written by varying ρ then k then m. The total number of triplets is the number of loops to perform. For the PK method, the triplets are ρ , m and v, where v is a velocity.

Then for each list matrix on QHHL, an(m,k) pair is written on FSAVE and its trailer is initialized. CASECC is written onto FSAVE and the loop on method is started.

4.116.7.1 K Method

For part two, FLØØP is incremented by one, TSTART is set by a call to KLØCK and a branch on method is taken. After the matrices are built, FLØØP is set to -1 if this is the last loop.

The following matrices are built each time through the loop:

$$[M_{hh}^x] = (k^2/b^2)[M_{hh}] + (\rho_{ref} \rho/2)[Q_{hh}(m,k)] \quad (1)$$

$$[K_{hh}^x] = [K_{hh}] \quad (2)$$

$$[B_{hh}^x] = k/b[B_{hh}] \quad (3)$$

where k - reduced frequency (varies with loop)

b - reference length (from AERØ card)

ρ_{ref} - referencing density (from AERØ card)

ρ - density (varies with loop)

$[Q_{hh}(m,k)]$ - interpolated aerodynamic matrix at (m,k) (varies with loop)

FUNCTIONAL MODULE FA1 (FLUTTER ANALYSIS - PHASE 1)

Note: For this method BHH may be purged.

Most of the work for the K method is done while building $[Q_{hh}(m,k)]$ (see Figure 1) FA1 calls FA1K to build $[Q_{hh}(m,k)]$ and then calls SSG2C to perform the add. FA1K will call MINTRP when it is necessary to build a new interpolated matrix.

4.116.7.2 KE Method

All the loops are done within FA1, the eigenvalue problem is solved and the answers are sent directly to FA2.

For each m,k pair subroutine FA1K is called to build $Q_{hh}(m,k)$, then FA1KE is called to solve the eigenvalue equation:

$$\left[k^2/b^2 [M_{hh}] + \rho_{ref} \rho/2 [Q_{hh} \ m, k] \right] \lambda^2 + [K_{hh}] = 0 \quad (4)$$

M_{hh} and K_{hh} are put in core on the first loop. Then on each loop subroutine INCØRE is used to return the eigenvector given the sum of $M_{hh} + Q_{hh}$ and $-K_{hh}$. Subroutine ALLMAT is called to give the eigenvalues. The eigenvalues are written on BXHH for every loop. On the last loop the eigenvalues are copied from BXHH to FSAVE.

4.116.7.3 PK Method

All the loops are done within FA1, the eigenvalue problem is solved, and the answers are sent directly to FA2.

For each new mach number FA1PKI is called to build $[A]^{-1}$ and $[Q]$ in core.

For each m,k pair FA1PKE is called to solve the eigenvalue problem.

$$[M]p^2 + [B]p + [K] \phi = 0 \quad (5)$$

where:

$$[M] = -[M_{hh}]$$

$$[B] = -[B_{hh}] + \rho_{ref} \rho/2 \ b \ v \ [Q^I]$$

$$[K] = -[K_{hh}] + \rho_{ref} \rho/2 \ v^2 \ [Q^R]$$

MODULE FUNCTIONAL DESCRIPTIONS

and

$$\begin{bmatrix} Q^R \\ Q^I \end{bmatrix} = [Q][C] \quad (6)$$

where:

$$[C] = [A^{-1}](p_v)$$

$$(p_v) = \left(\frac{|k_{est} - k_j|^3 + |k_{est} + k_j|^3}{1} \right) \quad 1 \leq j \leq \begin{matrix} \text{number of frequencies at a mach} \\ \text{number} \end{matrix} \quad (7)$$

The \bar{A} matrix is then formed and the eigenvalue equation is solved by subroutine FAIPKA.

$$[\bar{A} - pI] \Psi = 0 \quad (8)$$

where:

$$[A] = \left[\begin{array}{c|c} 0 & I \\ \hline -M^{-1}K & -M^{-1}B \end{array} \right]$$

The eigenvalues are accepted only if

b/v times imaginary part of $p = k_{est}$

where:

k_{est} starts at zero and is equal to b/v I_{mp} for first root not accepted.

This requires an iterative procedure until the number of roots desired is reached. The roots saved are put on FSAVE for each MR pair and if eigenvectors are requested they are printed by FAIPK. At the end of the loop the eigenvalues are put on BXHH and the eigenvectors on KXHH.

FUNCTIONAL MODULE FA1 (FLUTTER ANALYSIS - PHASE 1)

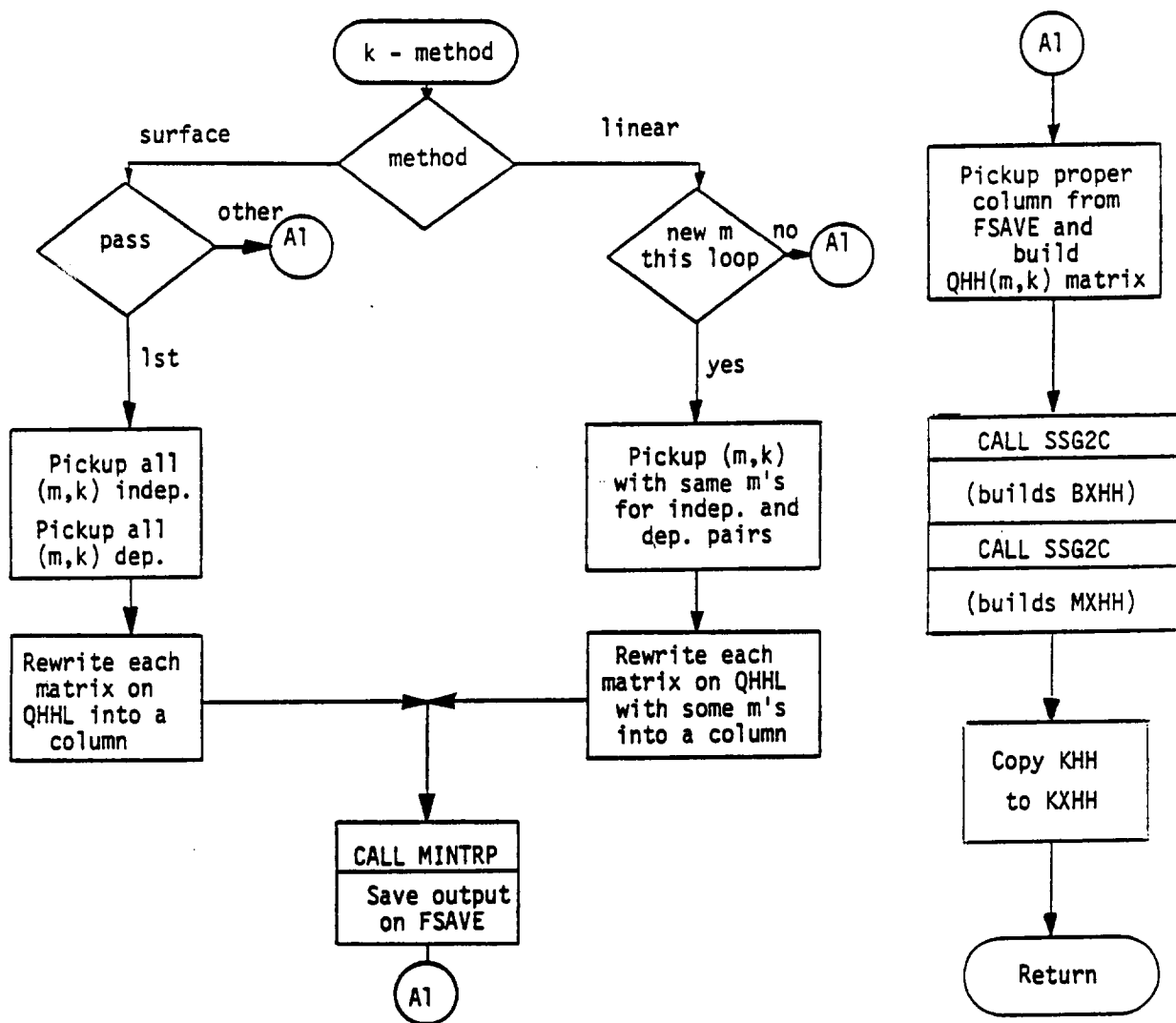


Figure 1. Flow Chart for k - method

4.116.8.1 Subroutine Name: FA1

1. Entry Point: FA1
2. Purpose: Module driver (see method Section 4.116.7)
3. Calling Sequence: CALL FA1

4.116.8.2 Subroutine Name: FA1K

1. Entry Point: FA1K
2. Purpose: Build QHH(m,k) (see method Section 4.116.7)
3. Calling Sequence: CALL FA1K(SMETH,K,RH0,OUTFIL)

MODULE FUNCTIONAL DESCRIPTIONS

where:

- SMETH - 1 = surface spline interpolation; 2 = linear spline interpolation (input-integer)
- k - Reduced frequency for this loop (real-output)
- RHØ - density (ρ) for this loop (real-output)
- ØUTFIL- GINØ file number for QHH(m,k) (integer-input)

4.116.8.3 Subroutine Name: MINTRP

1. Entry Point: MINTRP
2. Purpose: Produce an interpolated matrix.
3. Calling Sequence: CALL MINTRP (NI,XI,ND,XD,TYPE,SYMM1,SYMK1,DZ,INFILE,ØUTFIL, SCR,SCR1, G,NCØRE,NØGGØ,IPRES)

where:

- NI - Number of independent (m,k) pairs.
- XI - Pairs of m and k (m(I), k(I) I = 1, NI).
- ND - Number of dependent (m,k) pairs.
- XD - Pairs of m and k (m(I), k(I) I = 1, ND)
- TYPE - Type of interpolation and type of output; positive means interpolation for (m,k), negative means interpolation for k only.
 - ±1 = complex interpolation for k only
 - ±2 = real part of interpolated matrix
 - ±3 = imaginary part of interpolated matrix
 - ±4 = complex interpolation matrix plus slopes
 - ±5 = real part of interpolated matrix plus slopes
 - ±6 = imaginary part of interpolated matrix plus slopes
- SYMK1 - Symmetry flag about k; 0 = no symmetry, 1 = symmetry, -1 = antisymmetry.
- SYMM1 - Symmetry flag about m; 0 = no symmetry, 1 = symmetry, -1 = antisymmetry.
- DZ - Attachment flexibility.
- INFILE - GINØ file which contains complex matrix to be interpolated.
- ØUTFIL - GINØ file for the resulting matrix.
- SCR, SCR1 - Scratch files for MINTRP.
- G - First location of open core for MINTRP.
- NCØRE - Amount of open core from G.
- NØGGØ - 1 is singular matrix, 0 otherwise.
- IPRES - Precision of multiply operation.

4.116.8.4 Subroutine Name: FA1KE

1. Entry Point: FA1KE
2. Purpose: Solve the eigenvalue problem for the KE method.
3. Calling Sequence: CALL FA1KE(OUT,K,B,RH0,RREF,FL00P,NL00P)

where:

OUT - GINØ file number for QHH(M,K)
 K - Reduced frequency for this loop
 B - Reference lengths
 RHØ - Density for this loop
 RREF - Reference density
 FLØØP - Loop count
 NLØØP - Total number of loops to do

4.116.8.5 Subroutine Name: FA1PKI

1. Entry Point: FA1PKI
2. Purpose: Build an interpolated inverse matrix and Q matrix in core
3. Calling Sequence: CALL FA1PKI(FSAVE, QHHL)

where: FSAVE and QHHL are GINØ file numbers

4. Method

For all values of K(nk) at mach m build a matrix of order NK+1 and INVERT

$$A_{ij} = \begin{bmatrix} |K_i - K_j|^3 + |K_i + K_j|^3 & \text{for } 1 \leq i \text{ or } j \leq NK \\ 1 & \text{for } i = NK + 1 \text{ or } j = NK + 1 \\ 0 & \text{for } i = j = NK + 1 \end{bmatrix}$$

and build Q from QHHL - divide imaginary points by K.

4.116.8.6 Subroutine Name: FA1PKE

1. Entry Point: FA1PKE

MODULE FUNCTIONAL DESCRIPTIONS

2. Purpose: Solve the iterative eigenvalue problem for the PK method.

3. Calling Sequence: CALL FA1PKE(KHH,BHH,MHH,BXHH,FSAVE,NLØØP,B,RREF,NEIW,EPS)

where: KHH,BHH,MHH,BXHH and FSAVE are GINØ file numbers

NLØØP - Number of loops to do

B - Reference length

RREF - Reference density

NEIW - Number of roots wanted

EPS - Epsilon for acceptance testing

4. Method

MHH, BHH, and KHH are placed in core for each new mach number. The inverse of MHH is taken and then the loop for solutions starts with $k=0$. The partitions of the \bar{A} matrix are built and FA1PKA is called to build \bar{A} and solve. Only the roots with positive imaginary parts are used in epsilon tests. The roots are tested and saved. When the number of roots wanted is blocked a record of FSAVE is written and a new loop starts. Eigenvectors are printed for each root when the velocity of their loop is flagged negative by the user. FA1PKV is called to print this eigenvector.

4.116.8.7 Subroutine Name: FA1PKA

1. Entry Point: FA1PKA

2. Purpose: Solve for eigenvalues

3. Calling Sequence: CALL FA1PKA(A,M1K,M1B,EIV,NCØRE,N)

where:

A - Space for \bar{A} matrix

M1K - Address where $M^{-1}K$ is stored

M1B - Address where $M^{-1}B$ is stored

EIV - Space for answers

NCØRE - Amount of core available at EIV

N - Size of problem to be solved

FUNCTIONAL MODULE FA1 (FLUTTER ANALYSIS - PHASE 1)

4. Method

\bar{A} is built in core and NSBG is called to decompose \bar{A} , then ATEIG is called to get the eigenvalues. Diag. 39 will print the time taken by HSBG and ATEIG.

4.116.8.8 Subroutine Name: FA1PKV

1. Entry Point: FA1PKV
2. Purpose: Print eigenvector for PK method
3. Calling Sequence: CALL FA1PKV(A,M1K,M1B,N,EIV,CZ,BREF,PI,VEL,BUF)

where: A, M1K, M1B, N, EIV are the same as FA1PKA

CZ - Scratch storage

BREF - Reference lengths

PI - π

VEL - Velocity for the loop

BUF - GINØ buffer

4. Method

A matrix is built from the eigenvalue in question and EGNVCT is called to give the eigenvector.

$$[AZ] = [EIG [-M1B] - [M1K]] EIG. [I] \quad (9)$$

The vectors are saved on SCR1.

4.116.8.9 Subroutine Name: RSØRT

1. Entry Point: RSØRT
2. Purpose: Sort real arrays in core
3. Calling Sequence: CALL RSØRT(N,I,Z,J)

where:

N - Number of words in a group

I - Word in group to sort on

MODULE FUNCTIONAL DESCRIPTIONS

Z - Anywhere groups are stored

J - Length of Z

Note: If I .LT. 0 sort by absolute value of Ith word.

4.116.9 Design Requirements

1. FA1 has an open core common block FA1XX.
2. One scratch file is used.
3. The K and KE methods use open core at FA1KXX.
4. The PK method uses open core at FA1PKX and a communication common block FA1PKC to pass pointers into FA1PKX.
5. FA1 uses six scratch files.

4.116.10 Diagnostic Messages

System fatal messages 3001, 3002, 3003, 3007, 3008, and 3061 can be produced as well as user fatal messages 2266 - 2271. Singular matrix from INVERS may also be produced.

FUNCTIONAL MODULE FA2 (FLUTTER ANALYSIS - PHASE 2)

4.117 FUNCTIONAL MODULE FA2 (FLUTTER ANALYSIS - PHASE 2)

4.117.1 Entry Point: FA2

4.117.2 Purpose

To collect data for reduction and presentation for each loop through the configuration parameters.

4.117.3 DMAP Calling Sequence

FA2 PHIH,CLAMA,FSAVE / PHIHL,CLAMAL,CASEYY,ØVG / V,N,TSTART / C,Y,VREF=1.0 / C,Y,PRINT=YES \$

4.117.4 Input Data Blocks

PHIH - Complex eigenvectors - h set, modal formulations.

CLAMA - Complex eigenvalue output table.

FSAVE - Flutter storage save table.

Note: No input data block may be purged.

4.117.5 Output Data Blocks

PHIHL - Appended complex mode shapes - h set.

CLAMAL - Appended complex eigenvalue output table.

CASEYY - Appended case control data table.

ØVG - Output aeroelastic curve requests (V-g or V-f).

Notes:

1. No output data block may be purged.
2. All output data blocks are read (DMAP attribute APPEND) on subsequent calls (FLØØP from FSAVE ≠ 1 if the method is K).

4.117.6 Parameters

TSTART - Integer-input/output-no default value. On input TSTART is the CPU time at the start of the DMAP flutter loop. On output TSTART will be -1 if there is insufficient time for another DMAP loop.

VREF - Real-user input; no default. V_{out} will be scaled by VREF:

$$V_{out} = V/V_{ref} .$$

PRINT - BCD-user input-default = YES. If PRINT = NØ, no flutter summary will be printed.

MODULE FUNCTIONAL DESCRIPTIONS

4.117.7 Method

The primary purpose of module FA2 is to gather data for reduction and presentation. The header record of FSAVE will contain the METHOD. The actions of FA2 are method dependent.

4.117.7.1 K-Method

This module is near the end of a DMAP loop. Its output files PHIHL, CLAMAL, CASEYY and ØVG are appended for each entry. On the first pass, special code must be executed to initiate the files.

The complex eigenvalues λ have been found by module CEAD. These should have been sorted by $\text{Im}(\lambda)$ increasing. Only use the first "NVALUE" modes. The quantities that need to be computed are:

$$\begin{aligned} V_{\text{out}} &= \text{Im}(\lambda)/V_{\text{ref}} \quad , \\ g &= \begin{cases} (2.0) \text{Re}(\lambda)/\text{Im}(\lambda) & \text{if } \text{Im}(\lambda) \neq 0 \\ 0 & \text{if } \text{Im}(\lambda) = 0 \end{cases} \quad , \\ f &= k \text{Im}(\lambda)/2\pi b_{\text{ref}} \quad , \\ V_{\text{Mach}} &= V_{\text{sound}}^m/V_{\text{ref}} \quad . \end{aligned}$$

The values of the parameter FLØØP, m, k, b_{ref} and NVALUE are found in the file FSAVE. A printer output is prepared.

The PHIHL, CASEYY and CLAMAL data blocks are created by appending the PHIH, CASEYY and CLAMA data blocks.

The CASEYY data block is for modules SDR2 and PLØT. It must keep in step with the append vectors. m, k, ρ and FLØØP will be added to the LABEL.

The ØVG data block is appended each time through the LØØP. This will be used to create V-g or V-f plots. m, k, ρ and FLØØP will be added to the LABEL.

4.117.7.2 PK-Method

The values for λ , I, and G are supplied by FA1 on FSAVE for all eigenvalues and all MACH number RHØ pairs. FA2 collects all k values together and outputs each collection of N such eigenvalues on a curve for V-g plotting. CASEYY, PHIHL and CLAMAL are not written.

4.117.7.3 KE-Method

Existing FSAVE records contain records of length $2 \times N$ where $2 = \text{Real}, \text{Imag} = V$, $N = \text{number of modes}$.

FUNCTIONAL MODULE FA2 (FLUTTER ANALYSIS - PHASE 2)

The records are sorted by (see Figure 1)

m = mach number

k = reduced frequency

ρ = density.

The output should be sorted by

m = mach number

ρ = density

n = mode number.

The records will be used for \emptyset VG, and for formatted print.

A special "sorting" algorithm will be used to order the roots. For the first k value in each loop, the roots are accepted in the order of ALLMAT. Define

the ith eigenvalue for the

P_{in} = nth reduced frequency k.

In the above, i = 1, 2, 3, ... (number of modes)

n = 1, 2, 3, ... (number of k values)

Define the extrapolated value based upon previous n's to be

$$P_{i,2}^e = P_{i,1}$$

$$P_{i,n}^e = P_{i,(n-1)} + (k_n - k_{n-1})(P_{i,(n-1)} - P_{i,(n-2)}) / (k_{n-1} - k_{n-2}) \quad n \geq 3$$

where $P_{i,0}$ will be chosen equal to $P_{i,1}$. Then, select for $P_{1,n}$ the root found in the nth loop closest to $P_{1,n}^e$. Delete that root and let $P_{2,n}$ be the one of the remaining roots closest to $P_{2,n}^e$. Continue until all roots are exhausted. The measure of "closeness" of the complex numbers is the square of the magnitude of the difference. If P_1 and P_2 are two roots,

$$P_1 = \text{Re } P_1 + i \text{ Im } P_1,$$

then the square is,

$$[\text{Re}(P_2 - P_1)]^2 + [\text{Im}(P_2 - P_1)]^2.$$

All eigenvalues are put on the FSAVE data block to be passed to FA2 module. CASEYY, PHIHL and CLAMAL are not written.

MODULE FUNCTIONAL DESCRIPTIONS

In Summary

Flutter Summary (skip if PRINT ≠ YES)		Complex Eigenvectors
K - method (FA2 in loop)	Output in order received Point = (m,k,p) triplet # of entries = # of modes	<u>Always</u> Come in loop and PHIH, CLAMA slots (No change)
K - method (no loop)	Sorting required Point = (m,p,mode) triplet # of entries = # of k's	<u>None</u>
PK - method (no loop)	Transpose required Point = (m,p,mode) triplet # of entries = # of V's	<u>None</u>

Note: All must have ØVG

Figure 1.

4.117.8 Subroutines

Utility routine CYCT2B is called.

4.117.9 Design Requirements

Open core for FA2 is at /FA2X/ .

FA2 uses no scratch files.

4.117.10 Diagnostic Messages

The following messages may occur: 3001, 3002, 3003, 3007, 3008 and 3045. Only 3045 is a user message. It indicates that the DMAP loop was not completed by exhausting the configuration parameters but rather by a time-to-go failure.

EXECUTIVE DMAP MODULE PARAML (PARAMETERS FROM A LIST)

4.118 EXECUTIVE DMAP MODULE PARAML (PARAMETERS FROM A LIST)

4.118.1 Entry Point: PARAML

4.118.2 Purpose

To select parameters from a user input matrix or table.

4.118.3 DMAP Calling Sequence

PARAML INPUT // C,N,op / V,N,RECNØ / V,N,WØRDN / V,N,REAL1 / V,N,INTEG /
V,N,REAL2 / V,N,BCD \$

4.118.4 Input Data Blocks

INPUT - Any matrix or table

4.118.5 Output Data Blocks

None.

4.118.6 Parameters

- op - Input, BCD, no default. op must take on one of the values "DMI", "DTI", "NULL", or "PRESENCE".
- RECNØ - Input, integer, default = 1
- WØRDN - Input, integer, default = 1
- REAL1 - Output, real, default = 1
- INTEG - Output, integer, default = 0
- REAL2 - Output, real, default = 1.0
- BCD - Output, BCD, default = blanks

Notes:

1. REAL, INTEG, REAL2 and BCD will be set by the module only if they are "V" type parameters.
2. PARAML does its own SAVE

4.118.7 Method

PARAML interrogates INPUT and sets its parameters according to the value of op.

MODULE FUNCTIONAL DESCRIPTIONS

op = PRESENCE

INTEG will be -1 if INPUT is purged.

op = NULL

INTEG will be -1 if INPUT is a null matrix.

op = DMI

The value in column = RECNØ and row = WØRDN will be placed in REAL1. If the matrix is complex REAL2 will contain the imaginary part of the term.

op = DTI

The values in record = RECNØ and word = WØRDN will be placed in REAL, REAL2, INTEG and BCD. In general, only one interpretation will make sense.

PARAML uses subroutine FNDPAR (see Section 4.119.8.1) to locate the parameters in the ØSCAR entry.

4.118.8 Subroutines

PARAML calls FNDPAR (see Section 4.119.8.1).

4.118.9 Design Requirements

Common blocks /XVPS/, /SEM/, /ØSCENT/, and /SEM/ must be available to PARAML. Open core is defined at /PARMLX/.

4.118.10 Diagnostic Messages

Messages 3002, 3003 and 3007 may be issued for this module.

EXECUTIVE DMAP MODULE PARAMR (FLOATING POINT PARAMETER PROCESSOR)

4.119 EXECUTIVE DMAP MODULE PARAMR (FLOATING POINT PARAMETER PROCESSOR)

4.119.1 Entry Point: QPARAMR

4.119.2 Purpose

To perform specified arithmetic and tests on DMAP parameters.

4.119.3 DMAP Calling Sequence

PARAMR // C,N,op / V,N,ØUTR / V,N,INR1 / V,N,INR2 / V,N,ØUTC / V,N,INC1 /
V,N,INC2 / V,N,FLAG \$

where the following operations (op) are available:

<u>ØP</u>	<u>OUTPUT VALUE</u>
ADD	ØUTR = INR1 + INR2
SUB	ØUTR = INR1 - INR2
MPY	ØUTR = INR1 * INR2
DIV	ØUTR = INR1/INR2
NØP	no change
SQRT	ØUTR = $\sqrt{\text{INR1}}$
SIN	ØUTR = SIN(INR1)
CØS	ØUTR = CØS (INR1)
ABS	ØUTR = INR1
EXP	ØUTR = exp (INR1)
TAN	ØUTR = TAN (INR1)
NØRM	ØUTR = ØUTC
PØWER	ØUTR = INR1 ** INR2
ADDC	ØUTC = INC1 + INC2
SUBC	ØUTC = INC1 - INC2
MPYC	ØUTC = INC1 * INC2
DIVC	ØUTC = INC1 / INC2
CSQRT	ØUTC = $\sqrt{\text{INC1}}$
CØMPLEX	ØUTC = INR1 + i INR2

MODULE FUNCTIONAL DESCRIPTIONS

CØNJ	$\emptyset UTC = \overline{INC1}$
REAL	$INR1 = \text{Re}(\emptyset UTC)$ $INR2 = \text{Im}(\emptyset UTC)$
EQ	FLAG = -1 if $INR1 = INR2$
GT	FLAG = -1 if $INR1 > INR2$
LT	FLAG = -1 if $INR1 < INR2$
LE	FLAG = -1 if $INR1 \leq INR2$
GE	FLAG = -1 if $INR1 \geq INR2$
NE	FLAG = -1 if $INR1 \neq INR2$
LØG	$\emptyset UTR = \text{Log}_{10}(INR1)$
LN	$\emptyset UTR = \text{Ln}(INR1)$
FIX	FLAG = FIX($\emptyset UTR$)
FLØAT	$\emptyset UTR = \text{FLØAT}(\text{FLAG})$

Notes:

1. Output parameters must be a "V" type parameter if it is computed by op.
2. For op = DIV or op = DIVC the answer is zero if the denominator is zero.
3. PARAMR does its own SAVE.
4. For op=SIN or op=CØS or op=TAN, the input must be in radians.

4.119.4 Input Data Blocks

None.

4.119.5 Output Data Blocks

None.

4.119.6 Parameters

- op - Input, BCD, no default. op must be chosen from the above table.
- $\emptyset UTR$ - Output/Input-real-default = 0.0.
- INR1 - Output/Input-real-default = 0.0.
- INR2 - Output/Input-real-default = 0.0.
- $\emptyset UTC$ - Input/output-complex-default = (0.0,0.0)
- INC1 - Input-complex-default = (0.0, 0.0)
- INC2 - Input-complex-default = (0.0, 0.0)
- I - Output/Input-Integer-default = 0

4.119.7 Method

QPARMR searches on op code table for op and branches to the appropriate computation routine. QPARMR then calls subroutine FNDPAR to locate to output parameter in /ØSCENT/ and then in /XVPS/. A direct store into /XVPS/ is made for the output parameter(s).

4.119.8 Subroutines

4.119.8.1 Subroutine Name: FNDPAR

1. Entry Point: FNDPAR
2. Purpose: To search the ØSCAR for a specified parameter and to return its index into the /XVPS/.
3. Calling Sequence: CALL FNDPAR (IP,IL)
/ØSCENT/ ØSCAR(1)

where:

- IP = The parameter number desired. If IP > 0, a fatal error will occur if the IP's parameter is not a variable. If IP < 0 and the |IP|th parameter is not a variable
IL = -1
- IL = -1 (see IP)
or the index of the IPth parameter in the /XVPS/.

4.119.9 Design Requirements

Common blocks /XVPS/, /ØSCENT/, /SEM/, and /SYSTEM/ must be available to this module.

4.119.10 Diagnostic Messages

Messages 3007, 3123 and 3124 may be issued for this module.

FUNCTIONAL MODULE ØPTPR1

4.120 FUNCTIONAL MODULE ØPTPR1 (Fully Stressed Design Phase I)

4.120.1 Entry Point: ØPTPR1

4.120.2 Purpose:

To create a list of elements and properties which may be fully stressed along with the pointers and property limits.

4.120.3 DMAP Calling Sequence:

ØPTPR1 MPT,EPT,ECT,DIT,EST / ØPTP1 / V,N,PRINT / V,N,TSTART / V,N,CØUNT \$

4.120.4 Input Data Blocks:

MPT - Material Property Table - material, PØPT and PLIMIT data.

EPT - Element Property Table - property data.

ECT - Element Connection Table - property ID's for elements.

DIT - Direct Input Table - temperature dependent materials.

EST - Element Summary Table - element temperature.

Notes: The MPT may be purged or no PØPT card may exist in which ØPTPR1 returns.
All other input data blocks must exist.

4.120.5 Output Data Blocks:

ØPTP1 - Property Optimization Table.

Note: If ØPTP1 is purged, ØPTPR1 immediately returns.

4.120.6 Parameters:

PRINT - Iteration print control parameter, output, integer = 1.

TSTART - Ending time of ØPTPR1, output, integer.

CØUNT - Iteration counter, output integer. Set to zero or if no elements may be optimized it is set to -1.

4.120.7 Method

4.120.7.1 Overview of the Method

This module creates a data block that may be updated by ØPTPR2 with each iteration. Only elements and their associated properties that meet the optimization criteria will be output.

MODULE FUNCTIONAL DESCRIPTIONS

The optimization criteria (assuming the PØPT bulk data card exists) is as follows:

1. The element must have a property card.
2. The material stress limit must exist for either the primary or secondary property that may be optimized. For example, the cross-sectional area of a rod needs tension and/or compression stress limits.
3. Sufficient open core exists for the data.
4. At least one element may be optimized.
5. The element is listed in internal tables. See Section 4.120.9.4 for a list of the elements.

4.120.7.2 Program Method

Subroutine ØPTPR1 controls all file open/close operations, subroutine calls, and exit conditions. The following steps are followed:

1. It is determined if the output file exists and if the PØPT card is present on file MPT. If not, CØUNT = -1, PRINT = 1 and TSTART is set before exiting.
2. If PLIMIT (property change limit) cards exist on file MPT, they are processed by ØPTPRX onto scratch file 1.
3. The start of open core is loaded with material data. Then ØPTPIA is called to load the remaining open core with as many elements that meet the optimization criteria. This is done from file EST, one element type at a time. Five words per element are reserved as follows:
 - Word 1 - integer - element identification number
 - Word 2-4 - real - temperature dependent stress limits for tension, compression and shear
 - Word 5 - integer - pointer to variable to optimize and the appropriate stress limit.
4. The remaining open core is filled with the corresponding property. Property ID's not used (or previously loaded) are not output. The fifth word of the element section is placed in the second word of this property section and the first property section word is set to the property ID. The fifth word of each element then points to the location of its property. This is done one element at a time.

5. File EPT is used in ØPTP1C to complete the property section data as follows:

Word 1 - integer - property ID.

Word 2 - integer - file EST property pointer and corresponding stress limit.

Word 3 - real - original property value.

Word 4 - real - last value of property (same as word 3).

Word 5 - real - change factor (set to -1.0).

Word 6 - integer - property limit pointer (set to zero).

6. Word 6 of the property section is set in ØPTP1D as the property limits are read into the remaining open core. If no limit exists for the property, word 6 remains zero.

7. File ØPTP1 is created direction from the data in core. Count is set to 0 and parameter TSTART is set before exiting.

4.120.8 Subroutines

4.120.8.1 Subroutine Name: ØPTPX

1. Entry Point: ØPTPX

2. Purpose: To analyze the PLIMIT bulk data cards.

3. Calling Sequence: CALL ØPTPX

CØMMØN //	}	See Section 4.120.9.3
CØMMØN / ØPTPW1 /		
CØMMØN / XXØPT1 /		
CØMMØN / NAMES /		See Section 2.5.1.8
CØMMØN / SYSTEM /		See Section 2.4.1.8
CØMMØN / GPTA1 /		See Section 2.5.2.1

4. Method: A preliminary pass to determine which element types are specified on the PLIMIT card or if ALL is specified. Illegal types are rejected and an error flag is set.

If any legal types exist, the file is re-read for the types specified and the predetermined number of entries. References to ALL are included in the resulting table. The table is sorted and checked for duplicate entries (which are error flagged). Unless the error flag is set, the scratch file is written as follows:

MODULE FUNCTIONAL DESCRIPTIONS

Word	Symbol	Type	Description	
1	IDE	Integer	Internal pointer to element type	} repeated for all specified ranges } one record per element type
2	N	Integer	Number of four word entries follow	
3 to 4N+2	PID1	Integer	Property ID - lower range	
	PID2	Integer	Property ID - upper range	
	PMIN	Real	Minimum property ratio change	
	PMAX	Real	Maximum property ratio change	

5. Additional Subroutines: ØPTPX1, EJECT, READ, BCKREC, FREAD, SØRT, WRITE, EØF, MESSAGE.

4.120.8.2 Subroutine Name: ØPTPX1

1. Entry Point: ØPTPX1
2. Purpose: To decode the PLIMIT bulk data card and store four word entries in core.
3. Calling Sequence:

CALL ØPTPX1 (*,STØR,NØGØ,NEN,LØC1)

CØMMØN //

CØMMØN / XXOPT1 /

CØMMØN / SYSTEM /

} See Section 4.12.9.3

} See Section 2.4.1.8

where

* - nonstandard return if insufficient core.

STØR - input - PLIMIT bulk data card words to decode, output - decoded values.

NØGØ - output - error counter.

NEN - input, output - number of words stored in core for this element type.

LØC1 - start of this element type in core.

4. Additional Subroutines: EJECT, SØRT, BISHEL, MESSAGE

4.120.8.3 Subroutine Name: ØPTP1A

1. Entry Point: ØPTP1A
2. Purpose: To place in core elements that may be optimized and initialize pointers.

3. Calling Sequence:

CALL ØPTP1A (ELT,ELØP,ELE)

COMMON //

COMMON / ØPTPW1 /

COMMON / GPTA1 /

COMMON / SYSTEM /

COMMON / MATIN /

COMMON / MATØUT /

COMMON / NAMES /

See Section 4.120.9.3

See Section 2.5.2.1

See Section 2.4.1.8

See Section 3.4.36

See Section 2.5.1.8

where

ELT - output - element type optimization pointer table - zero means element cannot be optimized. Non-zero is a pointer to ELØP array.

ELØP - output - two dimensional array defining the start of each element type (and of the corresponding property) in core.

ELE - output - element section of core.

4. Additional Subroutines: EJECT, READ, MAT, FREAD, MESSAGE.

4.120.8.4 Subroutine Name: ØPTP1B

1. Entry Point: ØPTP1B
2. Purpose: To allocate core for each unique property ID segregated by element type. The first two words of the property are set as well as pointers to it.

3. Calling Sequence:

CALL ØPTP1B (ELT,ELØP,ELE,PR)

COMMON //

COMMON / XXØPT1 /

COMMON / ØPTPW1 /

COMMON / GPTA1 /

COMMON / SYSTEM /

COMMON / NAMES /

See Section 4.120.9.3

See Section 2.5.2.1

See Section 2.4.1.8

See Section 2.5.1.8

MODULE FUNCTIONAL DESCRIPTIONS

where

ELT - input - element type optimization pointer table.

ELØP - input, output - array defining the start of each element type and of the corresponding property in core. Only the property section is changed in ØPTP1B.

ELE - input, output - element section of core.

PR - output - property section of core.

4. Method: For each element type that has element entries in core, a sequential match of the element ID on the ECT file is made. The property ID from the ECT and the EST/Stress Limit variable are placed in the property section of core. The element section of core is set to point to the location (relative to the element type's property starting location) of its property.

5. Additional Subroutines: BISHEL, BISLØC, LØCATE, READ, FREAD, EJECT, MESSAGE.

4.120.8.5 Subroutine Name: ØPTP1C

1. Entry Point: ØPTP1C

2. Purpose: To initialize the property section using file EPT.

3. Calling Sequence:

CALL ØPTP1C (ELT, ELØP, PR)

CØMMØN //

CØMMØN / ØPTPW1 /

CØMMØN / XXØPT1 /

CØMMØN / SYSTEM /

CØMMØN / GPTA1 /

CØMMØN / NAMES /

} See Section 4.120.9.3

See Section 2.4.1.8

See Section 2.5.2.1

See Section 2.5.1.8

where

ELT - input - element type optimization pointer table.

ELØP - input - element type and property core location pointers.

PR - input, output - property section of core.

FUNCTIONAL MODULE ØPTPR1

4. Method: File EPT and the property section of core are assumed to be sequential.
This may not be the case for two different properties on one Bulk Data Deck card.
5. Additional Subroutines: LØCATE, READ, FREAD, EJECT, MESSAGE.

4.120.8.6 Subroutine Name: ØPTP1D

1. Entry Point: ØPTP1D
2. Purpose: To store the maximum and minimum property change limit values in core.
If a property has a property limit, the sixth word of the property is set to point to the property limit.
3. Calling Sequence:
CALL ØPTP1D (ELØP,PR,PL)

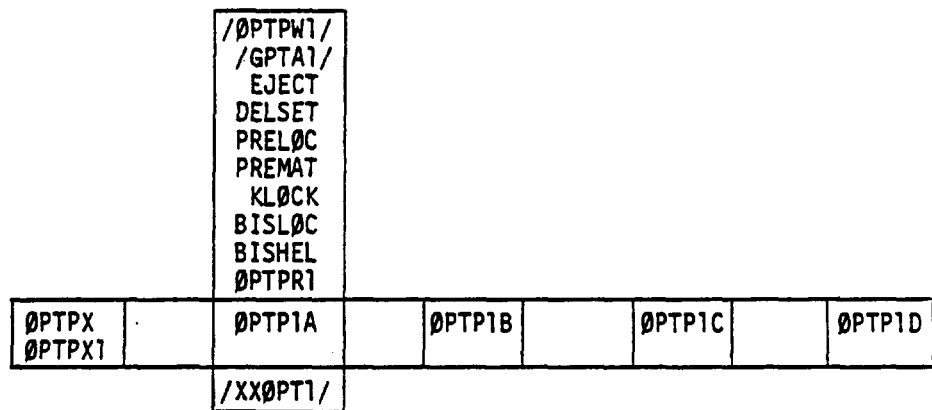
CØMMØN //	}	See Section 4.120.9.3	
CØMMØN / ØPTPW1 /			
CØMMØN / SYSTEM /			See Section 2.4.1.8
CØMMØN / NAMES /			See Section 2.5.1.8
4. Method: The scratch file containing interpreted property limits is read to determine the element type. If elements of this type exist the four word entries are then read. A sequential search of the properties ID's in core is made to see if the property ID falls within the "thru range" of the limit (the first two words). If it does, the sixth word of the property is set to point to the property list. Only unique property limits are loaded into core.
5. Additional Subroutines: READ, FREAD, EJECT, MESSAGE.

MODULE FUNCTIONAL DESCRIPTIONS

4.120.9 Design Requirements

4.120.9.1 Overlay Design

The module may in the future have the following overlay. However, it does not help to overlay if module ØPTR2 has less core available, (ØPTR2 does not have the material data section of core). Normal NASTRAN utilities are not shown.



FUNCTIONAL MODULE ØPTPR1

4.120.9.2 Open Core Design (Common Block XXØPT1)

X(1)	PØPT(6) - Data from the PØPT bulk data card
Y(1)	Material Data
Y(PCØR1)	ELT(NTYPES) - element type pointers in /GPTA1/ sequence. Contains zeros for elements to skip and an increasing integer sequence for types to optimize (NPØW types are non-zero).
Y(PCØR2)	ELØP(2,NPØW+1) - Element type and element property starting locations in ELE and PR arrays. ELØP(1,I) for elements and ELØP(2,I) for properties where I is the ELT value. The difference between succeeding entries is the number of words used for the element type.
Y(ECØR1)	<p>ELE(NELW) - Element section of core where NELW is the number of words. Each entry is as follows:</p> <p>ELE(1) - Element ID</p> <p>ELE(2) - σ_T</p> <p>ELE(3) - σ_C stress limits from material card</p> <p>ELE(4) - σ_S</p> <p>ELE(5) - Property card pointer relation to ELØP(2,I)</p>
Y(PRCØR1)	<p>PR(NPRW) - Property section of core where NPRW is the number of words. Each entry is as follows:</p> <p>PR(1) - Property ID</p> <p>PR(2) - EST word and element stress limit to optimize</p> <p>PR(3) - Original property</p> <p>PR(4) - Last property value - initially PR(3) = PR(4)</p> <p>PR(5) - Property change ratio</p> <p>PR(6) - Property change limit pointer relative to PL(1)</p>
Y(KCØR1)	<p>PL(NKLW) - Property change limits where NKLW is the number of words. Each entry is as follows:</p> <p>PL(1) - minimum property $\left(\frac{\text{NEW}}{\text{ORIGINAL}} \right)$</p> <p>PL(2) - maximum property $\left(\frac{\text{NEW}}{\text{ORIGINAL}} \right)$</p>

MODULE FUNCTIONAL DESCRIPTIONS

4.120.9.3 Block Data Interface

Values following description are the original values.

1. COMMON // PRINT,TSTART,COUNT, 'SKP(2)',YCOR,BIP1,NPØW,NELW,NWDSE,NPRW,NWDSP,NKLW

PRINT - Parameter to print, positive, or not print negative (1).

TSTART - Parameter ending time of this module.

COUNT - Parameter to execute ØPTPR2, zero, or not to execute ØPTPR2, negative (0).

YCØR - Open core available relative to COMMON / XXØPT1 / X(1)

BIP1 - Location of a GINØ buffer with 1 extra word.

NPØW - Number of element types that may be optimized (13).

NELW - Total number of words in element section of core (0).

NWDSE - Number of words for each element entry (5).

NPRW - Total number of words for each property entry (0).

NWDSP - Number of words for each property entry (6).

NKLW - Total number of words in property change ratio section of core (0).

2. COMMON / ØPTPW1 / ZCØR,Z(100)

ZCØR - length of array Z(100)

Z - fixed length array that each module may use as scratch space.

3. COMMON / XXØPT1 / X(6),Y(1)

X - Data from PØPT bulk data card.

Y - Open core.

4.120.9.4 Element Optimization Table

Element Type	Primary Property				Secondary property			
	Entry	Stress Limits	Related Change	Stress	Entry	Stress Limits	Related Change	Stress
RØD	A	σ_t, σ_c	J	σ_A	J	σ_s	A	τ
TUBE	O.D.	σ_t, σ_c		σ_A				
SHEAR	t	σ_s		τ_M				
TRIA1	t_1	$\sigma_t, \sigma_c, \sigma_s$	I, t_2	σ_p, τ_M	I	$\sigma_t, \sigma_c, \sigma_s$	t_1, t_2	σ_p, τ_M
TRBSC	I	$\sigma_t, \sigma_c, \sigma_s$	t_2	σ_p, τ_M				
TRPLT	I	$\sigma_t, \sigma_c, \sigma_s$	t_2	σ_p, τ_M				
TRMEM	t_1	$\sigma_t, \sigma_c, \sigma_s$		σ_p, τ_M				
QDPLT	t_1	$\sigma_t, \sigma_c, \sigma_s$	t_2	σ_p, τ_M				
QDMEM	t_1	$\sigma_t, \sigma_c, \sigma_s$		σ_p, τ_M				
TRIA2	t	$\sigma_t, \sigma_c, \sigma_s$		σ_p, τ_M				
QUAD2	t	$\sigma_t, \sigma_c, \sigma_s$		σ_p, τ_M				
QUAD1	t_1	$\sigma_t, \sigma_c, \sigma_s$	I, t_2	σ_p, τ_M	I	$\sigma_t, \sigma_c, \sigma_s$	t_1, t_2	σ_p, τ_M
BAR	A	σ_t, σ_c	J, I1, I2, I12	σ_M	J	σ_t, σ_c	A, I1, I2, I12	σ_M

where:

A = axial area
 O.D. = outside diameter
 t = thickness
 t_1 = membrane thickness
 t_2 = shear thickness
 I = bending inertia
 σ_M = maximum/minimum stress

 σ_p = principle stresses τ_M = maximum shear σ_A = axial stress τ = shear stress σ_t = tension stress σ_c = compression stress σ_s = shear stressmaterial
limits4.120.10 Diagnostic Messages

Messages are written on the output file directly and each subroutine returns to ØPTRP1 which causes a fatal error if needed. Messages that may occur are 2288 thru 2301. Queued messages are 3001, 3002, 3003, 3007 and 3061.

FUNCTIONAL MODULE DSCHK (DIFFERENTIAL STIFFNESS CHECK)

4.121 FUNCTIONAL MODULE DSCHK (DIFFERENTIAL STIFFNESS CHECK)

4.121.1 Entry Point: DSCHK

4.121.2 Purpose

Module DSCHK performs convergence checks and makes strategy decisions for use in the iterative Differential Stiffness calculations.

4.121.3 DMAP Calling Sequence

DSCHK PG1,PGI1,UBGV // C,Y,EPsi0 / V,N,DSEPSI / C,Y,NT / V,N,T0 / V,N,TI / V,N,DONE /
V,N,SHIFT / V,N,COUNT / C,N,BETAD \$

4.121.4 Input Data Blocks

PG1 - Matrix of load vectors from previous iterations - g set.

PGI1 - Matrix of load vectors from current iterations - g set.

UBGV - Matrix of solution vectors from current iteration - g set.

Note: No input data blocks may be purged.

4.121.5 Output Data Blocks

None.

4.121.6 Parameters

EPsi0 - Input by user - real - DMAP default = 1.0E-5. EPsi0 (ϵ_0) defines the acceptable ratio of energy error to total energy.

DSEPSI - Input/Output - real - no default. On entering DSCHK, DSEPSI is the last ratio of energy error to total energy. On leaving DSCHK, DSEPSI is the current value of this quantity.

NT - Input by user - integer - DMAP default = 10. Total number of iterations allowed.

T0 - Input - integer - no default. T0 is the number of CPU seconds which had elapsed when the last outer (recomputation of $[k_{gg}^d]$) DMAP loop was initiated.

MODULE FUNCTIONAL DESCRIPTIONS

- TI - Input - integer - no default. TI is the number of CPU seconds which had elapsed when the last inner (load iterations) DMAP loop was initiated.
- DØNE - Output - integer - no default. DØNE controls the final iterations. If DØNE = -1, the DMAP will proceed to data recovery. If DØNE = +N, N is the estimated number of other iterations which will be required to converge.
- SHIFT - Input/Output - integer - no default. If SHIFT is -1 on input, this indicates the first entry from the outer loop. On output, SHIFT = +1 if inner loop iterations are to continue; -1 if a return to the outer is to be scheduled.
- CØUNT - Input/Output - integer - no default. CØUNT is the number of iterations used to date. CØUNT should be set to zero initially. DSCHK will increment CØUNT by one for each entry.
- BETAD - Input by user - integer - DMAP default = 4. BETAD (β) is the shift decision factor. Small values of BETAD promote shifting; large values discourage shifting.

4.121.7 Method

Module DSCHK (Differential Stiffness Check) performs the following calculations.

The error ratio ϵ_i is:

$$\epsilon_i = \frac{|\{u^{i+1}\}^T \{p_g^{i+1} - p_g^i\}|}{|\{u^{i+1}\}^T \{p_g^i\}|} \quad (1)$$

(If $\epsilon_i \leq \epsilon_0$, a user supplied parameter, convergence has been achieved and exit mode 1 is initiated.)

The speed of convergence is the value λ , where

$$\lambda = \left| \frac{\epsilon_{i-1}}{\epsilon_i} \right| \quad (2)$$

If $\lambda < 1.0$ and $i > 1$, exit mode 2 is initiated.

FUNCTIONAL MODULE DSCHK (DIFFERENTIAL STIFFNESS CHECK)

The estimated number of iterations to convergence is:

$$N_f = \log \frac{\epsilon_i}{\epsilon_0} / \log(\lambda) \quad (3)$$

If $N_f > (N_t - N_i)$ the "shift" flag is set, where N_t is the user-specified limit and N_i is the number of times this module has been executed. If $(N_t - N_i) \leq 0$, the module will exit in mode 3.

The time remaining, T_r , is compared with the time to execute the inner loop, T_i , and the time to execute the outer loop, T_o .

If $T_i \cdot N_f > T_o + \beta T_i$, the "shift" flag is set. (β is a user-defined parameter, the default is 4.) If the module estimates it cannot finish the problem with either strategy, it will exit with mode 4.

The module will exit in mode 4 if:

$$T_r > T_o + \beta T_i \text{ and the shift flag is "on"}$$

or

$$T_r < T_i N_f \text{ and the shift flag is "off"}.$$

This flow is diagrammed in Figure 1.

4.121.8 Subroutines

Utility subroutine SSG2B is used. See subroutine descriptions, Section 3.

4.121.9 Design Requirements

DSCHK requires three scratch files. Open core for DSCHK is defined at /DSCHKX/.

4.121.10 Diagnostic Messages

Each entry into DSCHK generates message 7019.

Reasons for termination (mentioned in 7019) are as follows:

Reason Number	Meaning
0	Continue iteration
1	Converged to user supplied EPSIØ
2	Iterations are diverging
3	Insufficient time for convergence
4	User supplied iteration limit

MODULE FUNCTIONAL DESCRIPTIONS

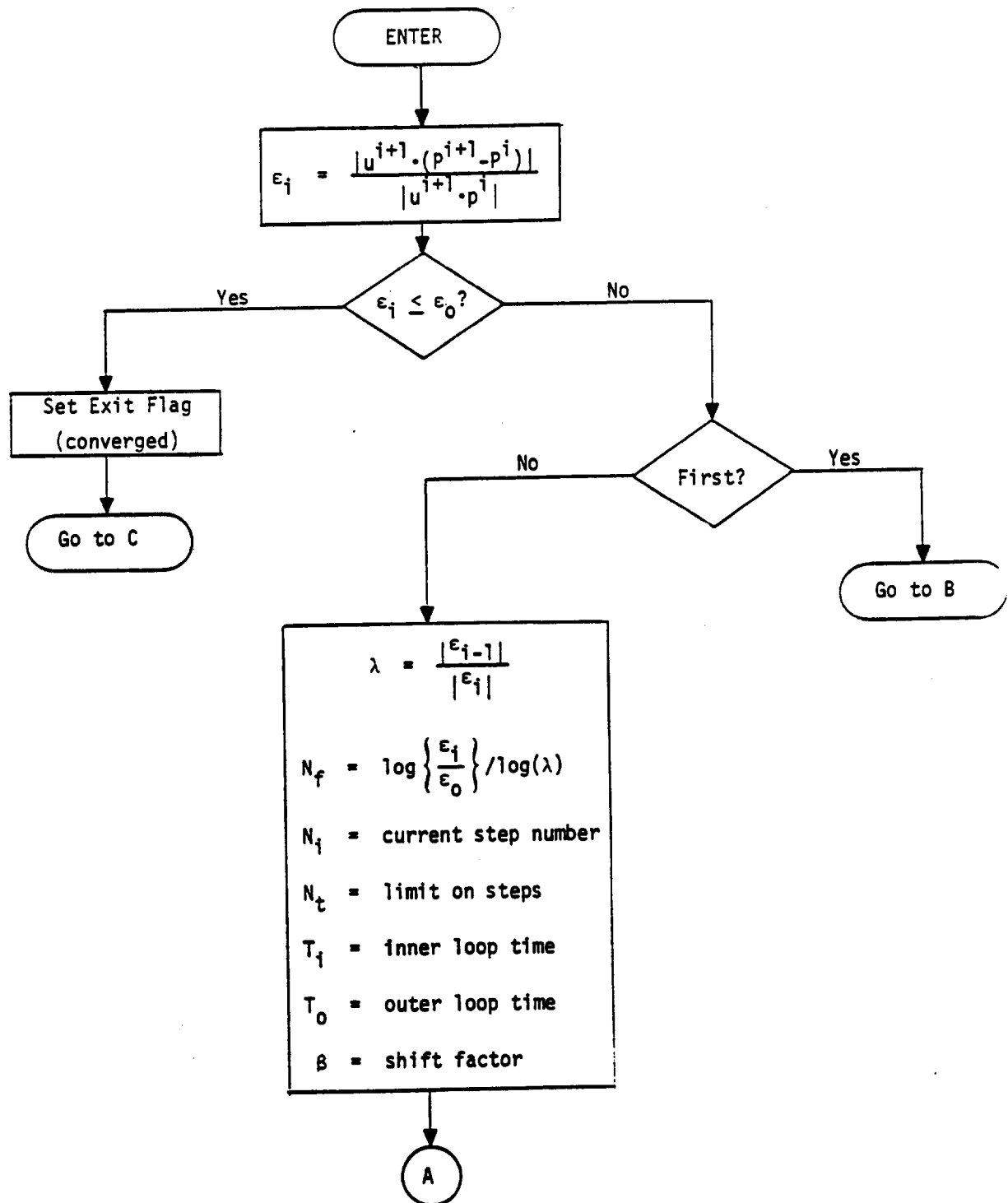


Figure 1. Flow Diagram for Module DSCHK

FUNCTIONAL MODULE DSCHK (DIFFERENTIAL STIFFNESS CHECK)

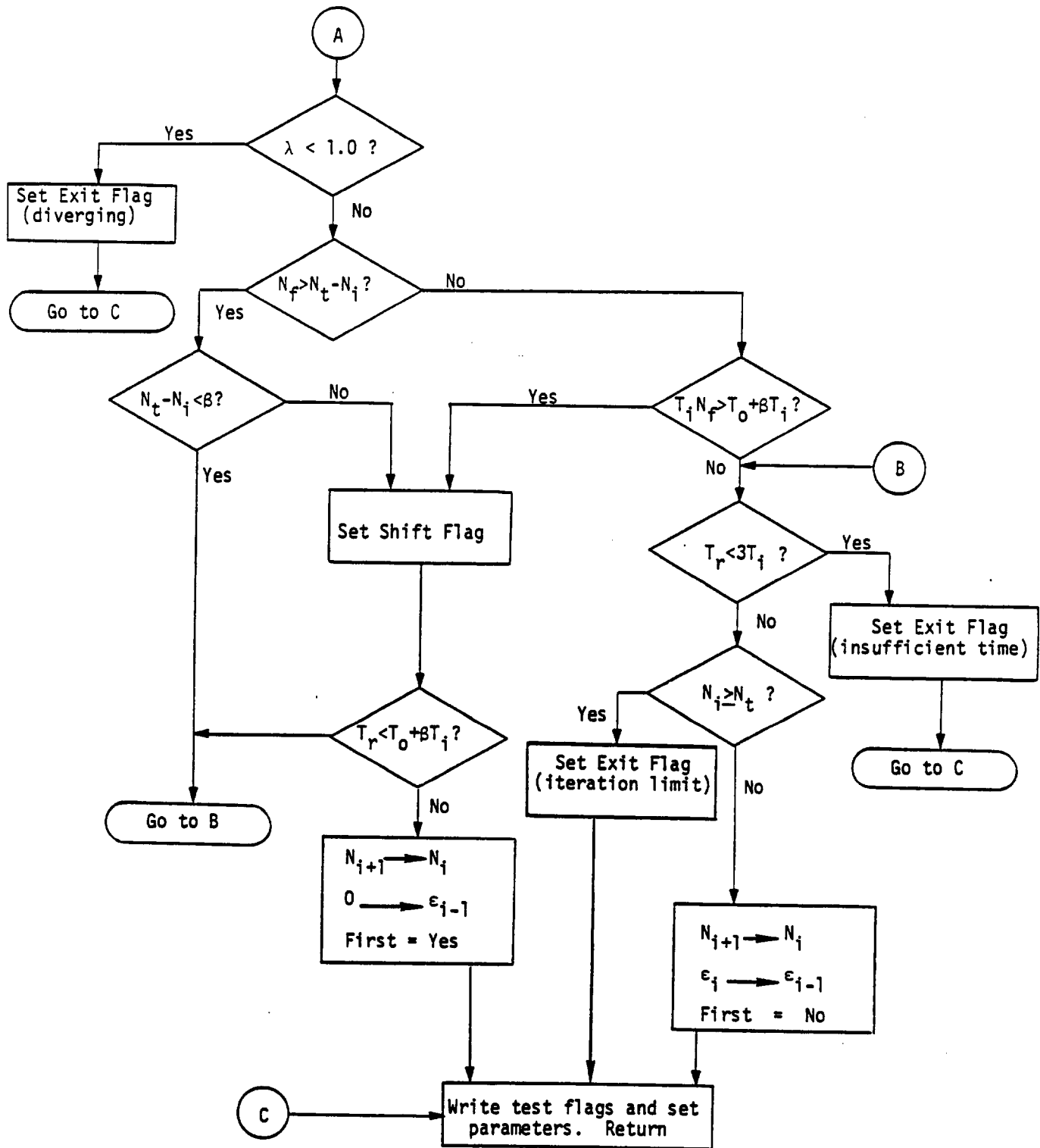


Figure 1. Flow Diagram for Module DSCHK (continued).

FUNCTIONAL MODULE TABPCH (TABLE PUNCH)

4.122 FUNCTIONAL MODULE TABPCH (TABLE PUNCH)

4.122.1 Entry Point: TABPCH

4.122.2 Purpose

To punch NASTRAN tables onto DTI cards for user post processing.

4.122.3 DMAP Calling Sequence

TABPCH TAB1,TAB2,TAB3,TAB4,TAB5 // V,N,A1 / V,N,A2 / V,N,A3 / V,N,A4 / V,N,A5 \$

4.122.4 Input Data Blocks

TAB1 - TAB5 -- Any NASTRAN tables.

Note: Any or all tables may be purged.

4.122.5 Output Data Blocks

None -- Output is punched onto DTI cards.

4.122.6 Parameters

A1 - A5 -- Input,BCD, defaults are 'AA', 'AB', 'AC', 'AD' and 'AE'. These parameters are used to form the first two characters (columns 74 and 75) of the continuation field for each table respectively.

4.122.7 Method

TABPCH reads a record at a time into open core. Each word is identified as to real, integer or BCD using the same process as used by TABPT. These values are punched into DTI cards. Integer and BCD values will be punched onto single field cards. A real number will be punched into a double field card. The formats are I8, 2A4, and E16.9 respectively.

4.122.8 Subroutines

None.

MODULE FUNCTIONAL DESCRIPTIONS

4.122.9 Design Requirements

1. Open core must be at /TABPHX/.
2. Up to 99,999 cards may be punched per table.
3. Twice any entire record in the table must fit into open core.
4. Tables with 1 word BCD values (ELSETS) will not be punched correctly.

4.122.10 Diagnostic Messages

Messages 3008 and 4015 may occur from this module.

4.123 FUNCTIONAL MODULE EMA (ELEMENT MATRIX ASSEMBLER)

4.123.1 Entry Point: EMA

4.123.2 Purpose

To superimpose matrices corresponding to elements into a structural matrix corresponding to all degrees of freedom at all grid points.

4.123.3 DMAP Calling Sequence

EMA GPECT,XEMD,XMAT / XGG,GPST / C,N,NØK4 / C,N,WTMASS \$

4.123.4 Input Data Blocks

GPECT - Grid Point Element Connection Table

XEMD - X-matrix Element Matrix Dictionaries (X = K, M, B, or KD)

XMAT - Element matrices

4.123.5 Output Data Blocks

XGG - Structural Matrix (X = K, M, B, KD, or K2)

GPST - Grid Point Singularity Table

4.123.6 Parameters

NØK4 - Input-integer-no default. Flag which specifies whether damping factor is to be used in assembling matrix (-1 ignores factor).

WTMASS - Input-floating point-default = 1.0. Constant by which all element matrix terms are multiplied.

4.123.7 Method

EMA is divided into two phases. The first phase merges the connection (GPECT) and matrix dictionary table (XEMD) data blocks. The result is a list, in grid point order, of all parameters and element matrix pointers necessary to assemble the final matrix. The second phase performs the assembly of the structural matrix itself by utilizing the list built during the first phase and a new direct access capability in GINØ to read the element matrices.

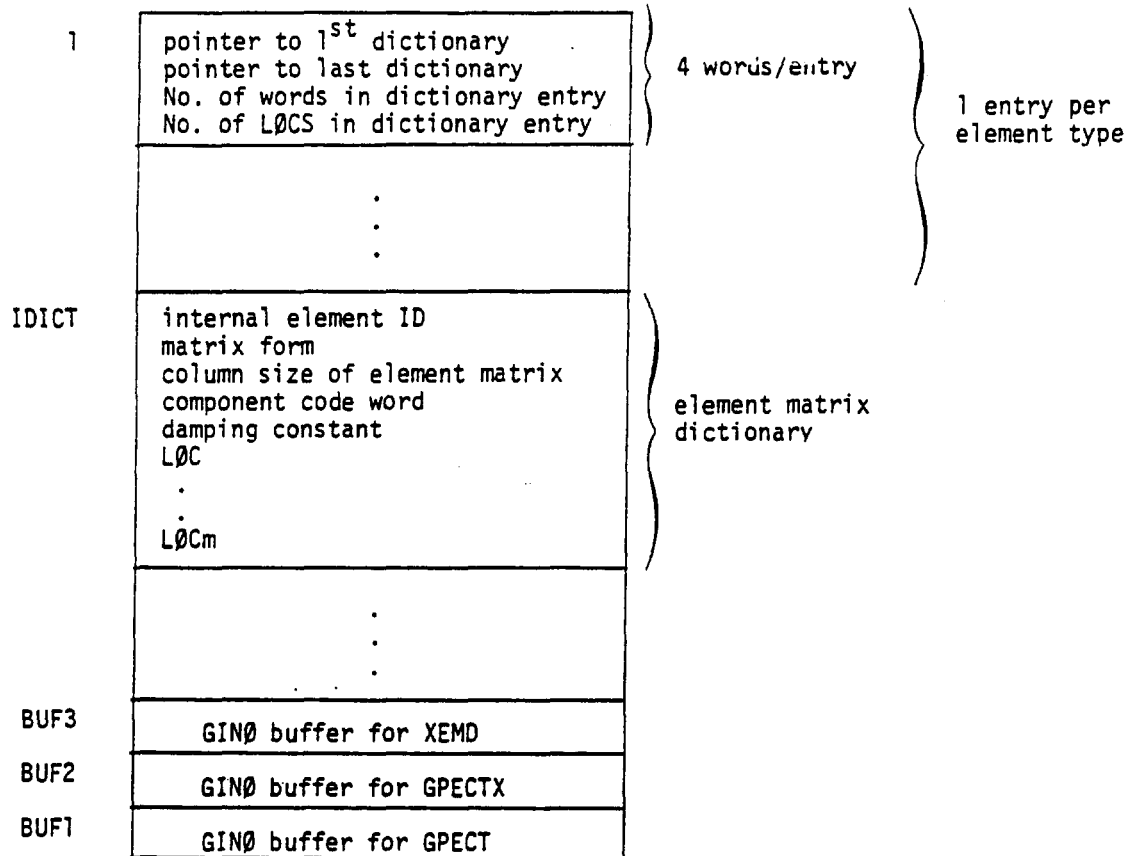
MODULE FUNCTIONAL DESCRIPTIONS

The first phase proceeds as follows:

1. At the top of open core a table of four words per entry one entry per element type (as determined from /GPTA1/) is allocated and set to zero.
2. Element matrix dictionaries are read from XEMD until either core is filled or the data block is exhausted. Pointers to the first and last dictionary for each element type are noted in table described above.
3. The GPECT (or partially completed GPECTX) is read one entry at a time. Four conditions are possible:
 - a. The dictionary for the element has already been attached on a previous pass. In this case the entry is copied directly to the new GPECTX.
 - b. The dictionary associated with the element has not yet been read into core. In this case the uncompleted entry is copied to GPECTX (it will be completed on a subsequent pass).
 - c. The dictionary associated with the element does not exist. For example, mass matrices are not defined for every element. In this case the entry is dropped.
 - d. The dictionary associated with the element is in core. In this case, the dictionary is attached and the completed entry is written on GPECTX.
4. When the last entry on GPECT has been read and the XEMD data block has been read in its entirety, phase two is initiated. Otherwise, the files for GPECT and GPECTX are switched and control returns to step (2) above.

Allocation of open core for the first phase is as follows:

FUNCTIONAL MODULE EMA (ELEMENT MATRIX ASSEMBLER)



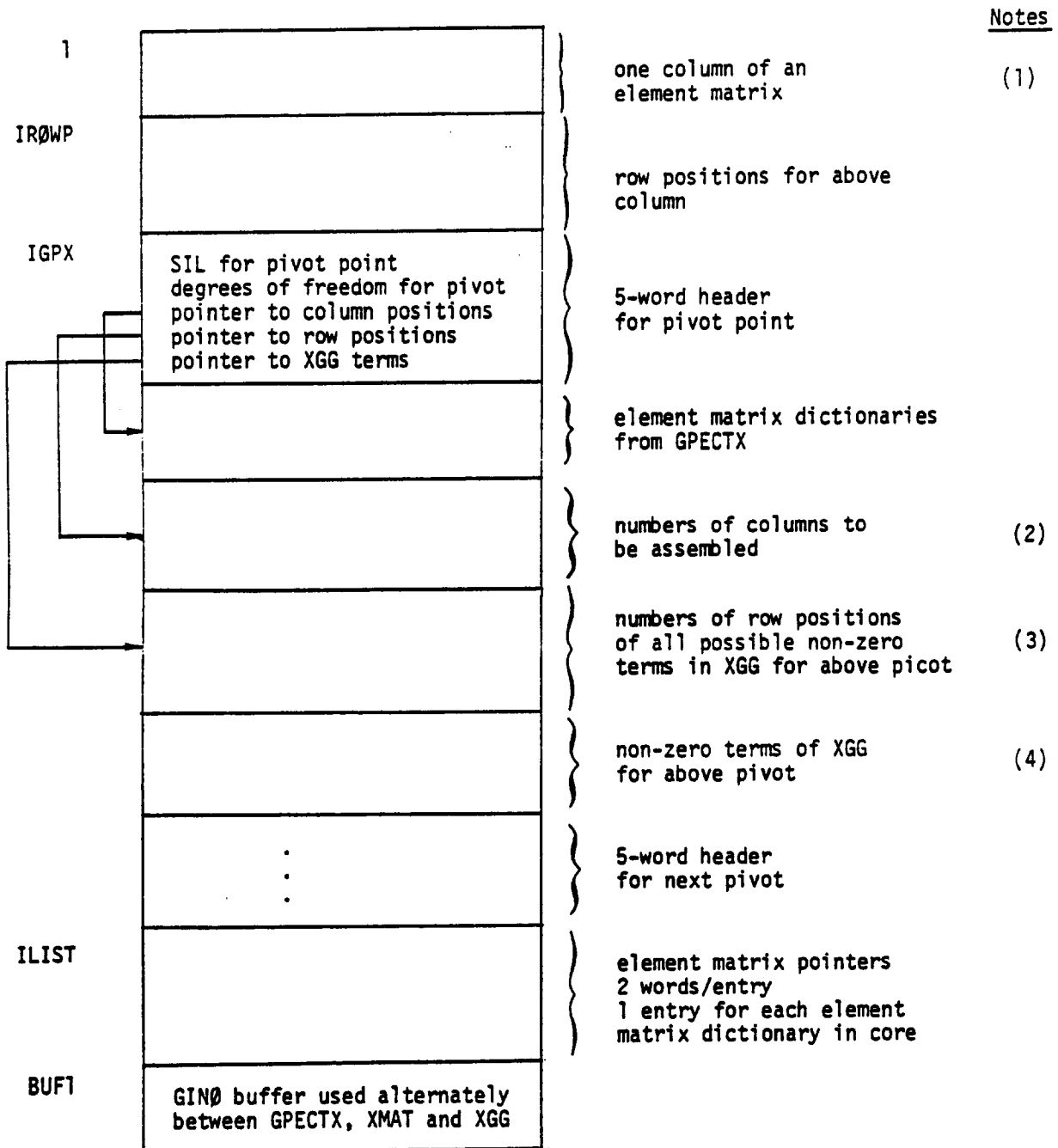
Entries on GPECTX are in the same order as GPECT, i.e., one logical record for each grid or scalar point and within each logical record one entry for each element attached to the grid or scalar point. The contents of each entry in GPECTX is as follows:

<u>Word No.</u>	<u>Description</u>
0	No. of words in entry (not including this word)
1	internal element ID
2	form
3	column size
4	component code
5	damping constant
6	LØC of element matrix for current pivot
7	SILi
.	.
.	.
.	.
n	SILm

MODULE FUNCTIONAL DESCRIPTIONS

During the second phase of EMA, XGG is assembled. GPECTX is passed (read) once. Open core is used to hold dictionary data, associated parameters and non-zero terms of XGG. Each pass during this phase involves assembling as many columns of XGG as may be accommodated by the size of open core.

The allocation of open core for this phase is as follows:



FUNCTIONAL MODULE EMA (ELEMENT MATRIX ASSEMBLER)

Notes

- (1) The size of this area is the maximum column size for any element matrix as determined during the first phase.
- (2) The number of columns to be assembled corresponds to the number of non-zero bit positions in the union of all component code words for elements connected to the pivot.
- (3) The number of row positions is determined by forming a unique list of row positions for all elements connected to the pivot.
- (4) The size of the array of non-zero terms for XGG is the product of the number of columns and number of rows as determined by notes (2) and (3) above.

The assembly phase proceeds as follows:

1. A record from GPECTX is read into open core. The record consists of a 5-word header and an arbitrary number of entries (there may be no entries if no elements connected to the pivot).
2. The list of entries for the pivot is scanned and the union of all component code words is formed. Additionally, a 2-word entry for each element entry (one word is the LØC and the other a pointer to the element entry) is formed and stored in a list in lower core.
3. From the union component word, a list of non-zero column numbers is formed.
4. The list of entries is now scanned a second time and a list of unique non-zero row numbers is formed.
5. Storage for the non-zero terms of XGG is now allocated.
6. If there is sufficient core for another pivot point, control is returned to step (1) above. Otherwise, the actual assembly begins.
7. The list of element pointers and LØCs is sorted by LØC in order to optimize the reading of element matrices from the direct access storage file.
8. The processing is now directed by passing the sorted list of element pointers.
For each 2-word entry:
 - a. The component code word for the element is decoded and a list of row numbers for the element matrix column is formed.

MODULE FUNCTIONAL DESCRIPTIONS

- b. This list is compared with that of the pivot to form correspondances between the two.
 - c. An element matrix column is read.
 - d. Multiplication by damping constant or weight mass factor (or both) is performed on each element matrix term.
 - e. The element matrix terms are added into their respective positions in XGG.
9. When all 2-word entries have been processed, the columns of XGG are packed onto the output data block.
 10. A test is made to determine if the GPST is to be generated. If so, then for each pivot in core, parameter information plus the XGG terms associated with the pivot are written on a scratch file.
 11. When all columns of XGG are complete, EMA is complete unless the GPST is to be generated. In the latter case, the scratch file generated during the assembly phase is read and DETCK is called to analyze the singularities and write the entries on the GPST data block.

4.123.8 Subroutines

4.123.8.1 Subroutine Name: DETCKX

1. Entry Point: DETCKX
2. Purpose: To generate the GPST by examining the 3x3 "translational" and 3x3 "rotational" diagonal submatrices of XGG.
3. Calling Sequence:

CALL DETCKX (JARG,IFGPST,NPVT)

where:

JARG	{	= 0 if the pivot point has connected elements
		= -1 if the pivot point is a scalar point with no connected elements
		= 1 if the pivot point is a grid point with no connected elements

IFGPST = GINØ reference name for the GPST data block

NPVT = SIL number for the pivot point

4. Method:

Let the pivot point be a grid point with scalar index p in the following discussion. Let $[Q]$ be the "translational" or "rotational" 3×3 symmetric submatrix along the diagonal of the stiffness matrix, $[K_{gg}^x]$, i.e., the rows and columns of the "translational" $[Q]$ matrix would correspond to scalar index numbers $p, p+1$, and $p+2$; and the rows and columns of the "rotational" $[Q]$ matrix would correspond to scalar index numbers $p+3, p+4$, and $p+5$.

The following steps comprise the algorithm for determining the presence or absence of grid point singularities. The discussion assumes $[Q]$ is the "translational" 3×3 matrix but the same algorithm holds for the "rotational" $[Q]$.

- a. The matrix $[Q]$ is scaled by the magnitude of the largest term, Q_{\max} :

$$[B] = \frac{[Q]}{Q_{\max}} \quad (1)$$

If the largest term is non-positive, the singularity is of order 3, and the scalar index numbers $p, p+1$ and $p+2$ are written on the GPST.

- b. The vector magnitudes of 3×1 columns (rows) are calculated:

$$b_1 = \sqrt{B_{11}^2 + B_{12}^2 + B_{13}^2} \quad (2)$$

$$b_2 = \sqrt{B_{21}^2 + B_{22}^2 + B_{23}^2} \quad (3)$$

$$b_3 = \sqrt{B_{31}^2 + B_{32}^2 + B_{33}^2} \quad (4)$$

- c. For each $b_i = 0$, the singularity order counter $IORDER$ is increased by one.
- d. If two b_i are zero, the order of the singularity is two, and the scalar index numbers j and k corresponding to these two rows of $[B]$ are written on the GPST.
- e. If one b_i is zero, and i is the row such that $b_i = 0$, define j and k as the other rows of $[B]$ and calculate:

$$m = \det \begin{bmatrix} B_{jj} & B_{jk} \\ B_{kj} & B_{kk} \end{bmatrix} \quad (5)$$

$$R = (B_{jj}^2 + B_{kj}^2) (B_{jk}^2 + B_{kk}^2) \quad (6)$$

If $\frac{m}{R} < TOL^*$, the order of the singularity is 2 and the GPST contains the paired scalar index values for i, j, k in the order (1) (i,j) if $B_{kk} > 0$ or (2) (i,k) if $B_{jj} > 0$. If $\frac{m}{R} \geq TOL$, the order of the singularity is one and only the SIL value for i is written on the GPST.

f. If all $b_i > 0$, we calculate

$$D = \det [B] \quad (7)$$

If $D > .05 \times TOL \times (b_1 b_2 b_3)$, there are no singularities, and DETCK returns if [Q] is the "rotational" matrix, or, if [Q] is the "translational" matrix, the "rotational" [Q] is input to the algorithm.

g. If $D > .05 \times TOL \times (b_1 b_2 b_3)$, one or more singularities exist. The following terms are calculated:

$$m_1 = \det \begin{bmatrix} B_{22} & B_{23} \\ B_{32} & B_{33} \end{bmatrix}, \quad (8)$$

$$m_2 = \det \begin{bmatrix} B_{11} & B_{13} \\ B_{31} & B_{33} \end{bmatrix}, \quad (9)$$

$$m_3 = \det \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}, \quad (10)$$

$$R_1 = \sqrt{(B_{22}^2 + B_{23}^2) (B_{32}^2 + B_{33}^2)}, \quad (11)$$

$$R_2 = \sqrt{(B_{11}^2 + B_{13}^2) (B_{31}^2 + B_{33}^2)}, \quad (12)$$

$$R_3 = \sqrt{(B_{11}^2 + B_{12}^2) (B_{21}^2 + B_{22}^2)}. \quad (13)$$

*The tolerance may be supplied by the user on the NASTRAN card by specifying the desired real value for SYSTEM(70) or STST. The default is .01.

h. Determine i, j, k such that:

$$\frac{m_i}{R_i} > \frac{m_j}{R_j} > \frac{m_k}{R_k} \quad (14)$$

- i. If $\frac{m_i}{R_i} < T\emptyset L$, the singularity is of order 2. Redefine i, j, k such that $B_{ii} < B_{jj} < B_{kk}$. The SIL values for the paired indexes (j,k), (i,k) and (i,j) are written on GPST only if the corresponding B is greater than zero. For instance if B_{kk} is zero, the SIL pair (i,j) is not written on the GPST.
- j. If $\frac{m_i}{R_i} \geq T\emptyset L$, (see step h) the singularity is of order 1. The SIL values are written on the GPST in the order

$$i \text{ since } \frac{m_i}{R_i} > T\emptyset L \quad (15)$$

$$j \text{ if } \frac{m_j}{R_j} > T\emptyset L \quad (16)$$

$$k \text{ if } \frac{m_k}{R_k} > T\emptyset L \quad (17)$$

4.123.9 Design Requirements

The first phase of EMA (GPECTX generation) is open ended. The second (assembly) phase requires that at least one complete pivot point be allocated (see core picture in 4.123.7). A fatal message No. 3008 is generated if this is not the case.

The design of EMA is open ended with respect to the number of degrees of freedom for a grid point. However, the algorithm used for determination of grid point singularities (subroutine DETCK) is valid only for grid points with 6 degrees of freedom. Consequently, this subroutine is called only for grid points with 6 degrees of freedom.

4.123.10 Diagnostic Messages

3001, 3002, 3003, 3008, and 3102.LØGIC ERRØR EMA-nnnn

In the latter message, nnnn refers to a FØRTRAN statement number in the code. Essentially this message refers to a "fail safe" check by EMA and most likely indicates either a program logic or obscure data error. A printout of the information which follows the message will be useful to the maintenance programming staff.

FUNCTIONAL MODULE EMG (ELEMENT MATRIX GENERATOR)

4.124 FUNCTIONAL MODULE EMG (ELEMENT GENERATOR)

4.124.1 Entry Point: EMG

4.124.2 Purpose

This module will compute and output stiffness, mass, and/or damping matrices for individual elements. The module also produces auxiliary description and location data; i.e., dictionary information, with respect to the above element stiffness, mass, and/or damping matrices. It should be noted this module does not assemble these element matrices with regard to some element linkage specification. The assembly process is performed by module EMA.

4.124.3 DMAP Calling Sequence

```
EMG    EST,CSTM,MPT,DIT,GEØM2,/{HKELM} {HKDICT} {HBELM} {BDICT} /V,N,NØK/V,N,NØM/  
      {KELM } {KDICT } {BELM } {HBDICT}  
  
      V,N,NØB/V,N,NØK4GG/V,N,NØKDGG/V,N,CONMASS/ $
```

4.124.4 Input Data Blocks

EST - Element Summary Table.
CSTM - Coordinate System Transformation Table.
MPT - Material Properties Table.
DIT - Direct Input Table.
GEØM2 - Geometry Table 2.

Note: 1. The CSTM may be purged. The MPT may be purged only if elements which do not reference any material data are used. The DIT may be purged only if the material properties are not temperature dependent.

4.124.5 Output Data Blocks

KELM - Element stiffness matrix partitions.
KDICT - Dictionary table for element stiffness matrix partitions.
MELM - Element mass matrix partitions.
MDICT - Dictionary table for element mass matrix partitions.
BELM - Element damping matrix partitions.
BDICT - Dictionary table for element damping matrix partitions.
HKELM - Element conductivity matrix partitions.
HKDICT - Dictionary table for element conductivity matrix partitions.
HBELM - Element capacity matrix partitions.
HBDICT - Dictionary table for element capacity matrix partitions.

MODULE FUNCTIONAL DESCRIPTIONS

Note

1. If either of a matrix-dictionary data block pair is purged, that particular data block pair will not be formed.

4.124.6 Parameters

- NØK - Input - integer - no default. ≤ 0 implies do not form stiffness matrix and dictionary data blocks. > 0 implies form stiffness matrix and dictionary data blocks.
- NØM - Same as NØK, but for mass matrices.
- NØB - Same as NØK, but for damping matrices.
- NØK4GG - ØUTPUT - default = -1. =1 implies nonzero damping constant detected. (= -1 otherwise)
- NØKDGG - currently not used.
- CØNMASS - Integer - input - no default. ≤ 0 implies do not form consistent mass matrices. > 0 implies form consistent mass matrices.

4.124.7 Method

Subroutine EMG is a small driver which calls serially the following main routines:

1. EMGTAB prepares material property, coordinate system transformation, direct input, and any other pertinent data tables in core.
2. EMGCNG prepares a small table in core which reflects the existence of any element congruencies.
3. EMGCØR allocates core and buffers as required, considering the outputs the module is to form on this execution.
4. EMGPRØ passes the EST, processing all elements and forming the tables of element matrices and dictionaries requested by the input parameters and problem make up.
5. EMGFIN wraps up all operations, closes data blocks, and writes trailers.

Each of these five routines are discussed in more detail below.

4.124.8 Subroutines

The utility routines PRETRD, PRETRS, and PREMAT are called by EMGTAB for initialization purposes, so that the structural element subroutines may call the entry points TRANSD, TRANSS, MAT, and HMAT to obtain coordinate system transformation matrices (CSTM) data, and material properties data, respectively. GMMATD, GMMATS, INVERD, INVERS, SAXB, DAXB, SADØTB, and DADØTB are for basic in-core matrix and vector computations. The descriptions for these routines may be found in Section 3.2 of the Programmer's Manual.

FUNCTIONAL MODULE EMG (ELEMENT MATRIX GENERATOR)

For purposes of communication between subroutines of the EMG module, the following common blocks are significant.

<u>Block</u>	<u>Variables in Order</u>
/EMGPRM/	<p>ICØRE - First word of open core.</p> <p>JCØRE - Next available location in open core.</p> <p>NCØRE - Current last available location in open core.</p> <p>ICSTM - First location in open core used for CSTM data.</p> <p>NCSTM - Last location of CSTM data.</p> <p>IMAT - First location of MPT data.</p> <p>NMAT - Last location of MPT data.</p> <p>IHMAT - Same as IMAT in "heat" formulation.</p> <p>NHMAT - Same as NMAT in "heat" formulation.</p> <p>IDIT - First location of DIT data.</p> <p>NDIT - Last location of DIT data.</p> <p>ICØNG - First location of congruency table.</p> <p>NCØNG - Last location of congruency table.</p> <p>LCØNG - Length of congruency table.</p> <p>ANYCØN - .TRUE. if congruency table exists. .FALSE. if congruency table does not exist.</p> <p>KMAT - .TRUE. if stiffness matrices will be computed, and output .FALSE. otherwise.</p> <p>MMAT - Same as KMAT for mass matrices.</p> <p>BMAT - Same as KMAT for damping matrices.</p> <p>PRECIS - 1 for module computations in single precision; 2 for module computations in double precision.</p> <p>ERRØR - .TRUE. when a fatal error has been flagged; .FALSE. otherwise.</p> <p>HEAT - .TRUE. when "heat" formulation is indicated; .FALSE. otherwise.</p>

MODULE FUNCTIONAL DESCRIPTIONS

/EMGEST/	ESTBUF(100) - Element entries, as read from the EST data block, are passed to element matrix computation routes via this buffer.
/EMGDIC/	ELTYPE - Current element type being operated upon. LDICT - Net length of element dictionary required for element type ELTYPE. NLØCS - Maximum number of element connection partitions to be formed. ELID - Current element identification as obtained from an EST element entry. ESTID - The serial element entry position in the EST. A pseudo identification is placed in the first word of every dictionary entry output.
/EMGZZZ/	Z - Open core for the module begins at point this block resides in core.
/EMGFIL/	EST - CSTM - MPT - DIT - GEØM2 - KMAT - MMAT - BMAT - KDICT - MDICT - BDICT -

GINØ file numbers for the files as listed.

4.124.8.1 Subroutine Name: EMGTAB

1. Entry Point: EMGTAB
2. Purpose: To establish in open core the coordinate system transformation data (if any) and material property data.
3. Calling Sequence: CALL EMGTAB.

FUNCTIONAL MODULE EMG (ELEMENT MATRIX GENERATOR)

4. Method: One GINØ buffer is allocated; the CSTM is then opened. If present, the CSTM data (Record 1) is read into open core. A call is made to PRETRS and PRETRD, which merely sends to these routines, for future calls to TRANSS and TRANSD, the address in open core of the CSTM data. Material property data is then entered into open core via a call to PREMAT, or if a "HEAT" formulation, a call to HMAT.

4.124.8.2 Subroutine Name: EMGCNG

1. Entry Point: EMGCNG
2. Purpose: To build in open core a table of element identifications of those elements specified as being congruent to each other via CONGRNT bulk data cards.
3. Calling Sequence: CALL EMGCNG.
4. Method: CONGRNT bulk data cards, if present, are found in data block GEØM2. If none are found, the routine returns. If CONGRNT cards are found, they are read, a table is constructed and sorted on the element ID. This table has the following design.

EXAMPLE

CORE LOCATION I.E.	TABLE COLUMN 1	TABLE COLUMN 2
Z (77)	ID ₁ =1	89
Z (79)	ID ₃ =2	0
Z (81)	ID ₂ =3	89
Z (83)	ID ₇ =4	79
Z (85)	ID ₆ =5	79
Z (87)	ID ₄ =6	79
Z (89)	ID ₅ =7	0

/EMGPRM/ Example values

ICØNG	= 77
NCØNG	= 90
LCØNG	= 14
ANYCØN	= .TRUE.

I.e., the above table resulted from the existence of two CONGRNT cards, the first of which contained ID₃, ID₄, ID₆, ID₇, and the second of which contained ID₅, ID₁, and ID₂.

Note the values in column 2 of the table are the open-core indices to the first ID which appeared in a congruency group. The zero in column 2 will later be replaced by a negated open-core index pointing to dictionary information.

4.124.8.3 Subroutine Name: EMGCØR

1. Entry Point: EMGCØR.
2. Purpose: To determine the type of matrix data blocks to be formed and after determination of these data blocks, to allocate sufficient GINØ buffers in the bottom of open core (highest addresses) for the operations which will be performed.
3. Calling Sequence: CALL EMGCØR
4. Method: DMAP parameters are noted, data blocks are opened (if possible), and flags are set to indicate operations to be performed.

4.124.8.4 Subroutine Name: EMGPRØ

1. Entry Point: EMGPRØ.
2. Purpose: To pass the EST data block once, and while making this pass, call the appropriate element matrix computation routines required to prepare data needed for the data blocks flagged to be formed.
3. Calling Sequence: CALL EMGPRØ.
4. Method: The EST contains one record for each element type the user has included in the problem being solved. Thus, for each record, the element type is determined, various parameters for the element type set, and a loop initiated to process all of the elements contained within the one type.

After each element's data entry is read from the EST record into /EMGEST/, a binary search is made (if any element congruencies exist) in the congruency table in open core for the element ID of the entry. If the ID is found, a further check is made in the table to determine if the "primary" ID of the congruency group has a negative pointer to dictionary information. If the pointer is negative, its value is the location of the dictionary information, which is output directly to the dictionary's data blocks being formed. No call is made to the element computation routine.

FUNCTIONAL MODULE EMG (ELEMENT MATRIX GENERATOR)

Should no dictionary information exist, or for that matter, no indication of a congruency, the appropriate element computation routine is then called. (A description of the placement of dictionary information in open core with respect to elements predisposed as being congruent may be found in the subroutine description of EMGØUT; see Section 4.122.8.6).

The element computation routine will then, based on the matrix flags set, compute and output to EMGØUT the element matrices and dictionaries.

Any dictionary information in open core is essentially purged between element types by the resetting of the dictionary data addresses associated with the primary ID's in the congruency table to zeros.

4.124.8.5 Subroutine Name: EMGFIN

1. Entry Point: . EMGFIN.
2. Purpose: EMGFIN merely closes any open data block and outputs appropriate trailers for these data blocks.
3. Calling Sequence: CALL EMGFIN.

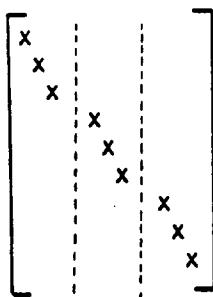
4.124.8.6 Subroutine Name: EMGØUT

1. Entry Point: EMGØUT.
2. Purpose: To receive element matrix data from element computation routines, and then output this data to appropriate matrix data blocks in table form. Dictionaries are also output, and when required by existence of an element congruency, saved in open core.
3. Calling Sequence: CALL EMGØUT (BUF,BUF,LSIZE,EØE,DICTIONARY,FILTYP,INPREC)

BUF - One or more, or all of the row-stored vertical partitions of the full element matrix. (If diagonal, then only diagonal terms of the partitions are present.)

MODULE FUNCTIONAL DESCRIPTIONS

- LSIZE** - The actual number of terms being sent. (Not the number of computer words.) Note if, for example, EMGOUT was being called with one partition of a diagonal 9x9 element matrix for which the element in question has three connection points, then LSIZE would be 3.



One partition of this same matrix, if square, would imply an LSIZE = 27.

- EØE** - If not all data has been sent on completion of this call for one element matrix, EØE should be set = 0. If this call supplies all, or the remaining one or more partitions, EØE should be set = 1.
- DICT** - An array of length LDICT found in /EMGPRM/ plus one location for each connected grid. (These extra locations will be set by EMGOUT, with GINØ direct access indexing information for later location of the matrix partitions.) For additional information on the contents of this array, see Section 6.8 concerning the programming and addition of a NASTRAN element matrix computation routine.
- FILTY** - Equals 1, if matrix data are stiffness type.
 Equals 2, if matrix data are mass type.
 Equals 3, if matrix data are damping type.
 Equals 4, if undefined future.
 Equals 5, if undefined future.
 Equals 6, if undefined future.
- INPREC** - Equals 1, if matrix data are single precision.
 Equals 2, if matrix data are double precision.
- Note: EMGOUT will accept data in single or double precision, but will output this data (making any necessary conversion) in the precision that the output data blocks are to be formed. However, INPREC must be set to 1 or 2 to match the data.

4. Method: EMGOUT maintains counts of the number of partitions received for each type of data block being formed. The calling routine may, if it desires, output matrix partitions alternately to all the data block types being formed before sending the final partitions or partitions of each data block type. This is with respect to any one element ID.

After checking the amount of data sent to EMGOUT, the element matrix data are then written on the appropriate matrix file, a partition at a time. (Each vertical partition becomes a record in the matrix partitions data block.) Simultaneously, the GINØ

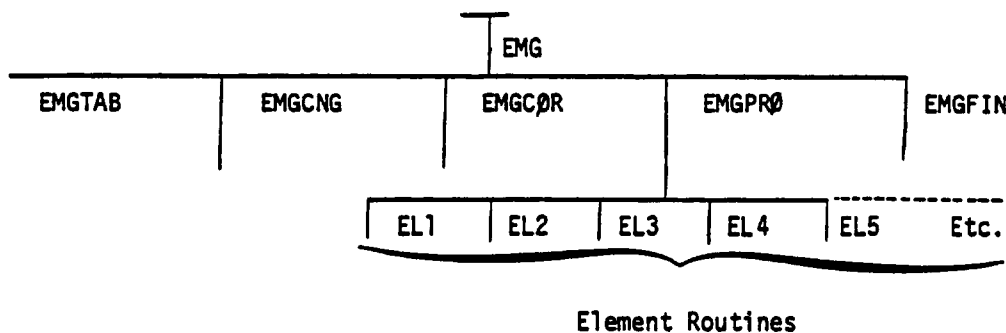
pointers for each record written are obtained via the GINØ entry point SAVPØS and stored in the dictionary sent on this call to EMGØUT.

After all partitions available are output to the element matrix partitions data block, a return is made if EØE = 0. If EØE is 1, then a check is made to determine if a correct and reasonable number of partitions have been sent and output. The dictionary entry for the element is then output to the dictionary data block type specified by FILTYP. If a congruent table exists, a binary search is made for the existence in the congruency table of the element ID of this dictionary. If the ID is not found, a return is made. If found, the primary ID of the congruency group is next located. When the primary element ID is found, it will have either a negative address or a zero associated with it.

If the address is zero, it implies no dictionary tables for the element exist. In this case, a table of length equal to the maximum number of element matrix data block types is added to open core. Its negated address is stored with the primary ID of the congruency group. This small table is set to zero when first formed. Thus, when an element dictionary entry is added to core, its core address is placed in this small table in position FILTYP. (Note, if at any time a dictionary or small table can not be placed in core due to insufficient core, the congruency is ignored and an information message output to the print file.) Return is then made.

4.124.9 Design Requirements

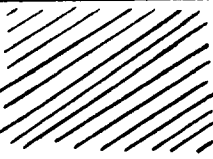
4.124.9.1 Basic Overlay for All Machines



Open core is placed under the longest of the element routines. (The above overlay diagram is not of complete detail.)

MODULE FUNCTIONAL DESCRIPTIONS

OPEN CORE

CSTM Data
MPT Data
MPT Data
DIT Data
Congruency Data

Congruency Dictionary Data
GINØ Buffer n
.
.
.
GINØ Buffer 3
GINØ Buffer 2
GINØ Buffer 1

Z(ICSTM)

Z(NCST:I)

Z(IMAT)

Z(NMAT)

Z(IHMAT)

Z(NHMAT)

Z(IDIT)

Z(NDIT)

Z(ICØNG)

Z(NCØNG)

Z(JCØRE)

Z(NCØRE)

This area is available to any subroutine on an as needed basis.

(Buffers are allocated and reassigned as needed for each set of problem requirements. n will vary.)

4.124.9.2 Precision of Computation and Element Routines

The matrix output routine EMGOUT accepts matrix partitions in single or double precision and outputs these in the precision, as specified by the module precision flag. As the module is making only one pass of the EST, a "ping pong" loader problem does not exist when each element type computation routine is in its own overlay limb. However, a "ping pong" loader problem will appear if the element routine is in various pieces for computation of stiffness, mass, and damping matrices. Thus, it is necessary to place stiffness, mass, and damping computation routines for a given element type in the same limb.

4.124.10 Diagnostic Messages

Messages 3103 thru 3112, 3301 and 3302 may be issued.

THERE IS NO SECT. 4.125
(available for future use)

4.126 FUNCTIONAL MODULE MØDACC (VECTOR SELECTION MODULE)

4.126.1 Entry Point: MØDACC

4.126.2 Purpose

To select vectors for further data reduction (such as Mode Acceleration or Stress Data Output).

4.126.3 DMAP Calling Sequence

R. F. 11

MØDACC CASEXX,PPF,UHVF,PPF,PDF,PSF / TØL1,UHV1F,PP3,PDF3,PSF3 / V,N,ØP=FREQ \$

R.F. 12

MØDACC CASEXX,PPT,UHVT,PPT,PDT,PST / FØL1,UHVT1,PP3,PDT3,PST3 / V,N,ØP=TRAN \$

R. F. 10

MØDACC CASEYY,CLAMAL,PHIHL,CASECC,, / CLAMAL1,CPHIH1,CASEZZ,, / C,N,CEIGN \$

4.126.4 Input Data Blocks

CASEXX - Case control - dynamics

CASEYY - Case control - aerodynamics

PPF
PPT
CLAMAL

- Source of list to be compared to ØFREQ (or ØTIME) set

UHVF
UHVT
PHIHL

- Vectors to be reduced

PPF
PPT

- Load file 1 to be reduced

PDF
PDT

- Load file 2 to be reduced

PSF
PST

- Load file 3 to be reduced

CASECC - Case control data table

Notes:

1. CASEXX or CASEYY may not be purged.
2. Output list (Input #2) may not be purged.

3. Vectors to be reduced may not be purged.
4. Any or all load files to be reduced may be purged.
5. CASECC may not be purged.

4.126.5 Output Data Blocks

FØL1
 TØL1
 CLAMA1

} - Output label files.

UHVT1
 UHV1F
 CPHIH1

} - Reduced vectors.

PP3 - Reduced loads - P size.

PDF3
 PDT3

} - Reduced loads - D size.

PSF3
 PST3

} - Reduced loads - S size.

CASEZZ - Case Control Data Block adjusted for reduction.

Notes:

1. Output file 1 may not be purged.
2. Output file 2 may not be purged.
3. Output files 3, 4 and 5 may be purged if their respective inputs are purged.
4. If ØP = CEIGN, output file 3 may not be purged.

4.126.6 Parameters

ØP - BCD-input-no default.

ØP specifies the type of processing to be performed; TRAN implies transient, CEIGN implies complex eigenvalues, FREQ implies frequency response.

4.126.7 Method

MØDACC determines the type of processing to be performed; TRAN, CEIGN or FREQ. It then builds a list of real numbers from INPUT number 2 (IN2). If TYPE = TRAN or FREQ, the list is from the header record of IN2; if TYPE = CEIGN, the list is the imaginary part of the complex

eigenvalue. It then marks selected members of the list for output based on the set selected by ØFREQ, and the logic used by SDR2. (One output will exist for every member of the ØFREQ set, it will be the nearest value to ØFREQ value.)

The reduced output list (if ØFREQ = ALL, the entire list) is written on the first output (TØL1). If TYPE = CEIGN this is a reduced CLAMAL and OUTPUT #3 is a corresponding CASEYY if enough records are present. The selected columns of the vector input is copied to the vector output file, 3 per selection if TYPE = TRAN, 1 per selection if TYPE ≠ TRAN. The load inputs are appropriately copied to the load outputs if present. Load output 1 has the reduced output list added to its header record.

4.126.8 Subroutines

Utility Subroutine CYCT2B is called.

4.126.8.1 MØDAC1

1. Entry Point: MØDAC1
2. Purpose: To reduce the number of entries in the output list to the set selected in CASECC.
3. Calling Sequence: CALL MØDAC1 (CASECC,TØL,TØL1,CASEZZ) where CASECC, TØL, TØL1 and CASEZZ are GINØ file numbers for Input Case Control, Input Output List, Output Output List and Output Case Control.

/MØDAC3/ NFØ,NFN,NZ,ID

NFØ - Length of old output list

NFN - Length of new output list

NZ - Length of open core

ID - 1 Frequency Response

2 Complex Eigenvalue

3 Transient

4.126.8.2 MØDAC2

1. Entry Point: MØDAC2
2. Purpose: To copy selected columns
3. Calling Sequence: CALL MØDAC2 (NV,INP1,IØUT)

MODULE FUNCTIONAL DESCRIPTIONS

NV - Number of columns to skip after copy. If $NV < 0$ put output list in IOUT header.
INP1, IOUT - GINØ file numbers of the INPUT and OUTPUT matrices.

/MØDAC3/ See 4.126.6.1

4.126.9 Design Requirements

No scratch files are used.

Open Core /MØDACX/ is as follows

New Times	New Times	}	NFN
	Keep/ not flags		
	Matrix Column	}	NRØW
	BUF1		
	BUF2	}	2*SYSBUF

4.126.10 Diagnostic Messages

Fatal Error Messages 3001, 3002, 3003 and 3008 may occur.

EXECUTIVE MODULE ASDMAP (ASSEMBLE SUBSTRUCTURE DMAP)

4.127 EXECUTIVE MODULE ASDMAP (ASSEMBLE SUBSTRUCTURE DMAP)

4.127.1 Entry Point: ASDMAP

4.127.2 Purpose

The ASDMAP module is used to process the NASTRAN Substructure Control Deck. These data are included in the user data deck following the Executive Control Deck and preceding the Case Control Deck. The functions of the module are to read and interpret the user commands, set up the control block for the SØF files, generate case-control type data for each command, and assemble a DMAP alter file which reflects the user input. The NASTRAN User's Manual describes the actual format and contents of the substructure commands in Section 2.7. The DMAP ALTER data to be modified and inserted are described in Section 5.5.

4.127.3 Calling Sequence

CALL ASDMAP

ASDMAP, a preface module is called only by subroutine SEMINT and only when word 69 of /SYSTEM/ is non-zero (see XCSA, Section 4.2).

4.127.4 Input Data

The input to the module consists of the Substructure Control Deck and the following files on the NPTP (Problem Tape) unit:

XALTER - User DMAP ALTER data (may not exist)

XCSA - Execution control file

4.127.5 Output Data Blocks

XALTER - Merged user and generated DMAP ALTER data on problem tape file.

CASECC - Decoded substructure commands are placed on this file to be followed by the normal case control data generated by IFP1. Module SGEN is used to split this data for normal processing.

4.127.6 Method

The step by step operations performed by ASDMAP are:

1. The page header "NASTRAN SUBSTRUCTURE DECK ECHO" is placed in the /ØUTPUT/ block, and other initial output operations are performed.

MODULE FUNCTION DESCRIPTIONS

2. The length of open core is established and three GINØ buffers are reserved.
3. Input cards are read until a SUBSTRUCTURE card is encountered. A fatal message is written for any illegal cards and NØGØ is set to 1. If a TITLE card is encountered before a SUBSTRUCTURE card NØGØ is set to 3. Otherwise, the substructure phase is placed in variable PHASE and the alter flag ALTER is set to .TRUE. The DMAP for the SUBSTRUCTURE command will be generated later.
4. The APP flag (SYSTEM(21)) and the ISUBS flag (SYSTEM(69)) are analyzed. If APP ≠ 2 (DISP) or ISUBS/10 ≠ 1, no substructure alters will be generated. In this case, ALTER is set to .FALSE. and step 5 is bypassed.
5. If ALTER = .TRUE., the NPTP (problem tape) is opened and analyzed. The SØL value (rigid format number) is obtained from file XCSA. If file XALTER exists, the NPTP is positioned there and the first two words (DMAP alter numbers) of record 1 are read into array ALTS; the flag ALTFL is set to .TRUE. If file XALTER does not exist, the NPTP is positioned to the beginning of file XCSA, and ALTFL is set to .FALSE. A record is kept of the NPTP file positions of the XALTER and XCSA files.
6. If PHASE = 1 or 2, CASECC is opened and the work "CASESS" is written onto it as record 0.
7. Various parameters are initialized. In particular, the variable ØBITS, whose bits correspond to the various matrix options (i.e., stiffness, mass, etc.), is assigned a default value depending on the rigid format number. ALTER is set to .FALSE. if APP = HEAT.
8. If ALTER = .TRUE., the scratch file SCRT (GINØ 301) is opened and the word "XALTER" is written onto it as record 0. The user-defined alters and automatically generated alters will be merged and temporarily stored on SCRT.
9. The next input card is read. If it is not an SØF or PASSWØRD card, control is passed to the command card processing algorithm (see step 10). If it is an SØF card, the SØF name and length are placed in the appropriate entries in the /SØFCØM/ block. If it is a PASSWØRD, the two-word password is also placed in the /SØFCØM/ block.
10. Processing then begins on the last command card read (initially the SUBSTRUCTURE command card). In the first step of this process a subroutine, ASCMii, is called, "ii" depending upon the command. ASCMii assembles the appropriate raw DMAP and working data into the /ASDBD/ block. The various ASCMii subroutines and the commands associated with each are as follows:

EXECUTIVE MODULE ASDMAP (ASSEMBLE SUBSTRUCTURE DMAP)

ASCM01 - SUBSTRUCTURE
ASCM02 - RUN, wrap-up DMAP (ENDS)
ASCM03 - COMBINE
ASCM04 - REDUCE
ASCM05 - SOLVE (statics and modal)
ASCM06 - RECOVER, MRECOVER
ASCM07 - BRECOVER
ASCM08 - SOLVE (dynamics)
ASCM09 - MREDUCE
ASCM10 - DESTROY, EDIT, EQUIV, SDFPRINT, DELETE, RENAME
ASCM11 - SDFIN, SDFOUT, RESTORE, DUMP, CHECK, COMPRESS, APPEND
ASCM12 - PLOT
ASCM13 - CREduce

The raw DMAP is then copied into array DMAP.

11. Input cards are read until the next command card is encountered. Each intermediate card is tested against keywords found in the /ASDBD/ block to see if it is a subcommand associated with the command being processed. The data for each subcommand is stored as three words in array EXTRA. Subcommands OUTPUT, RANGE, and RUN require special processing. Any illegal cards result in a fatal message being printed and N0G0 being set to 1.
 12. If an OPTIONS card is encountered, the specified matrices cause corresponding bits in the variable NEWBT to be turned on. These options are then saved and will be utilized for any subsequent command processed.
 13. If the command to be processed next is a RECOVER command, and PHASE = 3, it is internally converted to a BRECOVER command.
 14. Array VAR is then filled with various data needed for further processing of the current command. In general, its contents are as follows:
 - (1) Rigid format alter information obtained from the /ASDBD/ block.
 - (2) A copy of array EXTRA.
 - (3) Other option and rigid-format-dependent data.
- The data is stored as three words per entry as follows:
- (1) Keyword
 - (2) and (3) Associated data

MODULE FUNCTION DESCRIPTIONS

15. If ALTER = .TRUE., subroutine ASPRØ is called to make appropriate alterations to the raw DMAP based on the contents of array VAR and the working data produced by the ASCMii routine.
16. The modified DMAP is merged with XALTER data (if any) and written onto file SCRT. If overlapping of the DMAP alters occurs, a fatal error occurs.
17. If PHASE = 1 or 2, the command name keyword (one word), the length of array VAR (one word), and array VAR is written as one logical record onto file CASESS (CASECC). The next command is then processed.
18. It is seen to that a PHASE 2 RECOVER command always follows a PHASE 2 SOLVE command, whether or not the user has specified it in the Substructure Control Deck.
19. After all commands have been processed, a check is made of the SØF and PASSWØRD data. The SØF lengths are then called and an ENDSUBS card is written onto the system output.
20. If ALTER = .TRUE., the wrap-up DMAP is processed and written onto the end of the alter file.
21. The rest of the NPTP (problem tape) is copied onto SCRT. According to user specification, the complete set of DMAP alters is then printed (DIAG 23) and/or punched (DIAG 24). Finally, SCRT is written onto NPTP so that NPTP now contains the complete set of alters.

4.127.7 ASCMii Subroutines

4.127.7.1 Entry Point: ASCMii, ii = 1,...,13

4.127.7.2 Purpose

Subroutines ASCM01 through ASCM13 generate the working data for the ASDMAP and ASPRØ routines.

4.127.7.3 Calling Sequence

CALL ASCMii (CNAME,PHASE,SØL,NØGØ)

COMMON/ASDBD/IRDM,NRDM,IXTRA,NXTRA,

* IØCT,NØCT,IPTBS,NPTBS,IPH,NPH,IDAT(n)

CNAME - Command name keyword - BCD - input.

PHASE - Substructure phase number - integer - input.

SØL - Rigid format number - integer - input.

NØGØ - Error flag - integer - output
 IRDM - Pointer to first word of RDMAP table
 NRDM - Length of RDMAP table in words
 IXTRA - Pointer to first word of XTRA table
 NXTRA - Length of XTRA table in words
 IØCT - Pointer to first word of ØCT table
 NØCT - Length of ØCT table in words
 IPTBS - Pointer to first word of PTBS table
 NPTBS - Length of PTBS table in words
 IPH - Pointer to first word of PH table
 NPH - Length of PH table in words
 IDAT - Array containing tables of working data

4.127.7.4 Method

Five variable length tables, some of which may be of zero length, are constructed from internal data and assembled back-to-back into array IDAT. Along with pointers (into IDAT) and the table lengths, IDAT is passed through the /ASDBD/ block to be used by the ASDMAP and ASPRØ routines.

4.127.7.5 Description of Data Tables

The five data tables are described as follows.

RDMAP table (18 words per entry) - contains the raw DMAP card images corresponding to the current command. These images are later edited, depending on user input, and placed on the XALTER file.

XTRA table (1 word per entry) - contains keywords for allowable subcommand input for the current command.

ØCT table (3 words per entry) - contains data specifying the removal of certain raw DMAP card images, depending upon rigid format number and selected matrix options. Each table entry contains:

Word 1: The raw DMAP card number.

Word 2: Delete code. If an on bit in this word matches an on bit in the matrix option word, ØBITS, or the rigid format bit, the entire card is removed from the raw DMAP.

MODULE FUNCTION DESCRIPTIONS

Note:

- Bit 1 = Stiffness (K) matrix
- Bit 2 = Mass (M) matrix
- Bit 3 = Load (P) matrix
- Bit 4 = Load append (PA) matrix
- Bit 5 = Damping (B) matrix
- Bit 6 = Structural damping (K4) matrix
- Bit 17 = Rigid Format 1
- Bit 18 = Rigid Format 2
- Bit 19 = Rigid Format 3
- Bit 20 = Rigid Format 8
- Bit 21 = Rigid Format 9

Word 3: Keep code. The raw DMAP card image is rejected unless an on bit in this word matches an on bit in ØBITS. If this word is zero, it is ignored. Note: the delete code overrides the keep code.

PTBS table (7 words per entry) - contains data controlling the variable data blocks and parameters to be placed in the DMAP card images. The contents for each entry are:

- Word 1: The raw DMAP card number.
- Word 2: The column number, in the card image, of the first character in the variable string.
- Word 3: The column number of the first character to be inserted in the DMAP.
- Word 4: The number of characters to be inserted (0 to 8).
- Word 5: The "key" word of the variable to be inserted in the card image. The variables, given in the VAR array, are generated from user input and ASDMAP code. If a BCD word is used, the VAR array is searched for a match. If an integer, j, is used, the variable is an ALTER N1, N2 where N1 and N2 are given in the jth entry of the VAR array.
- Word 6: Option flag. The data block is removed if the bit corresponding to the current Rigid Format is on. Otherwise, it is kept only if ØBITS has at least one corresponding on bit, or if bits 1-6 are all off.

EXECUTIVE MODULE ASDMAP (ASSEMBLE SUBSTRUCTURE DMAP)

Word 7: Data block control. Certain DMAP language restrictions require this input. If the word is positive, the data block is input and must have been defined previously. (The list of active data blocks is searched and, if not found, the data block is removed.) If the word is negative, the data block is output and may not have been previously output. (It is added to the active data block list if it has not been previously output.)

PH table (2 words per entry) - contains pairs of numbers indicating Rigid Format DMAP alterations.

4.127.8 Subroutine Name ASPRØ (Assembler Processor)

4.127.8.1 Entry Point: ASPRØ

4.127.8.2 Purpose

Subroutine ASPRØ performs the basic assembly of the DMAP data for each set of substructure command inputs. Starting with the "raw" DMAP data, this routine fills in the variable data block and parameter entries and deletes selected DMAP statements based on user input.

4.127.8.3 Calling Sequence

CALL ASPRØ (DMAP,VAR,NVAR,ØBITS,SØL)

The arguments, which depend on each command type, are:

DMAP - DMAP card images (18 x NRM) - BCD - input/output

VAR - Description of variables and user options - input

Each entry contains:

1. Key word (BCD)
2. } BCD or integer data
3. }

NVAR - Number of input variables - integer - input

ØBITS - Matrix option flag word. Each bit corresponds to a matrix type which may be "on" or "off" - integer - input

Bit 1 = K (Stiffness)

Bit 2 = M (Mass)

Bit 3 = P (Loads)

MODULE FUNCTION DESCRIPTIONS

Bit 4 = PA (Appended Loads)

Bit 5 = B (Damping)

Bit 6 = K4 (Structural Damping)

SØL - Rigid Format number

4.127.8.4 Method

The basic computation steps are given below. For details of the working data, see Section 4.127.7.5.

1. The ØCT data is copied from the /ASDBD/ block into array ØCT. The given set of ØCT table entries are processed, one at a time, in order to remove extraneous DMAP cards based upon the Rigid Format number or the user-selected matrix options. If a data card is to be removed, the value, -1, is placed in the first word of the DMAP card image.
2. The PTBS data is copied from the /ASDBD/ block into array PTBS. The variable positions in the DMAP card images are then processed via the PTBS table.
 - a. A search is made in the VAR table to find a match between the key PTBS word and the variable name. (ALTER cards are set by matching the VAR entry number and the key number.) If a variable is not found, the RMV (remove) flag is set.
 - b. If the variable is found, its value (word 2 or words 2 and 3 of the VAR entry) is placed in the DMAP card image using subroutine PUSH.
 - c. If the option flag is on, the whole data block name is removed if it is not a selected matrix option or if the bit corresponding to the Rigid Format number is on.
 - d. If the variable affects an input or output data block which may conflict with the output of a previous module, it must be removed. A list is maintained of all output data blocks for the purpose of checking the validity of a data block. If a conflict exists, the data block is removed from the DMAP card image and replaced by blanks.
3. When a data block or variable field is "removed," blanks are inserted. If all variable positions are removed from one logical DMAP command, the DMAP card and its continuation cards are removed. The flag, -1, is placed in the first word of the DMAP card (or cards) image for this purpose.

EXECUTIVE MODULE ASDMAP (ASSEMBLE SUBSTRUCTURE DMAP)

4. After all DMAP entries have been processed, the DMAP card images flagged for removal are deleted from the array and the remaining card images are moved to compress the DMAP array. The number of remaining cards is placed in the /ASDBD/ block word NRDM.

MODULE FUNCTIONAL DESCRIPTIONS

4.127.9 Subroutine Name CØMBØ

4.127.9.1 Entry Point: CØMBØ

The CØMBINE command in the Substructure Control deck may contain a large amount of user data. This subroutine is used to process this data, set up the array of variables (VAR), determine the input substructure numbers, and then identify the resulting substructure.

4.127.9.3 Calling Sequence:

CALL CØMBØ (CDATA,NX,EXTRA,NNAM,NAME,NN,VAR,IER)

CDATA - Decoded image of CØMBINE command card as processed by subroutine XRCARD -
integer - input

NX - Number of entries in EXTRA array - integer - input

EXTRA - Array containing three words per data card following the command card - input.
The three words are:

1. Key name of EXTRA (TØLE, CØNN, etc.)
2. BCD word which follows key name or -1 if integer was input
3. Second BCD word or integer

NNAME - Number of substructure names in NAME table - integer - input/output

NAME - Table containing 2 BCD words per active substructure referenced in substructure
deck - integer - input/output

NN - Number of substructures to be combined - integer - output

VAR - Array defining all variables to be used in combine operation - output

IER - Error flag - If IER ≠ 0, a fatal error has occurred - integer - output

4.127.9.4 Method:

The primary steps in the execution are:

1. Set defaults on the ØPTION and SØRT requests
2. Decode the CDATA array. An example user input card is:

CØMBINE (AUTØ,XY) SUB1,SUB2,SUB3

XRCARD will produce the following array in CDATA:

EXECUTIVE MODULE ASDMAP (ASSEMBLE SUBSTRUCTURE DMAP)

<u>Word</u>	<u>Contents</u>
1	7 = Number of following BCD entries
2, 3	CØMB, INE
4, 5	Blank
6, 7	AUTØ (= 'ØPT')
8, 9	XY (= 'SORT')
10, 11	SUB1 (= 'NAMS')
12, 13	SUB2 (= 'NAMS')
14, 15	SUB3 (= 'NAMS')

3. The number corresponding to each of the substructure names is searched for in the NAME table. If it is found, that entry number is the substructure number. If it is not found, the name is added to the table and the entry number is stored.
4. The EXTRA table is moved to the VAR array.
5. The substructure numbers are placed in the VAR array (identified by 'N1', 'N2', etc.)
6. The name of the resulting substructure is added to the NAME array. The error flag is set if it already exists. Its number is placed in the VAR array after the key word 'NCNO'.

MODULE FUNCTIONAL DESCRIPTIONS

4.127.10 Subroutine Name REDU

4.127.10.1 Entry Point: REDU

4.127.10.2 Purpose:

This routine processes the user input for the REDUCE, MREDUCE, and CREduce substructure commands. The input variables are processed and placed in the VAR array.

4.127.10.3 Calling Sequence:

CALL REDU (CDATA,NX,EXTRA,NNAM,NAMES,NVAR,VAR,IPRE,IER)

CDATA - Image of REDUCE input card as decoded by subroutine XRCARD - BCD array - input

NX - Number of EXTRA entries - integer - input

EXTRA - Array of three words per extra input card - BCD - input

NNAM - Number of active substructure names - integer - input

NAMES - Array of 2 BCD words for each active substructure name - input

NVAR - Number of entries in VAR array - integer - output

VAR - Array of variables containing three words per entry - output

IPRE - Flag denoting word precision for matrix operations (1 = single, 2 = double) - integer - input

IER - Flags for errors encountered (If IER = 0, no errors) - integer - output

4.127.10.4 Method:

The name of the input substructure, 'NAMA', is found on the CDATA card image. The name of the resulting reduced structure, 'NAMB', is defined by the EXTRA input 'NAME'. The NAMES table is built and maintained to contain only the names encountered on each substructure command of the current set. For each execution, a new table is created. This NAMES table is searched to find the number corresponding to an existing name. If not found, the name is added to the NAMES table, and the location corresponds to the number. If 'NAMB' already exists, the error flag is set.

EXECUTIVE MODULE ASDMAP (ASSEMBLE SUBSTRUCTURE DMAP)

4.127.11 Subroutine PUSH

4.127.11.1 Entry Point: PUSH

4.127.11.2 Purpose:

Subroutine PUSH is used to insert a BCD string or an integer into a larger BCD string, replacing the existing characters. The first character and the number of characters is a variable.

4.127.11.3 Calling Sequence:

CALL PUSH (IN,BCDA,ICH,NCH,FLAG)

IN - Characters to be inserted in BCDA - BCD array or integer - input

BCDA - String of BCD data (A4) to be modified - BCD array - input/output

ICH - Index of first character in BCD to be modified. The ICH character of BCDA is replaced by the first IN character - integer - input

NCH - Number of characters to be inserted - integer - input

FLAG - Denotes type of IN data (0 = BCD, 1 = integer) - integer - input

4.127.11.4 Method:

1. If FLAG = 1, the integer IN is converted to BCD characters, left adjusted.
2. The first and last words of the BCDA array to be modified are found. Each word is treated as two parts. The right side of the first BCDA word will be replaced by the left side of the first IN word. The left side of the second BCDA word will be replaced by the right side of the first IN word, etc. The last word is a special case, whereby only the characters on the left side of the BCDA word are replaced by the remaining IN characters.

4.127.12 Subroutine PULL

4.127.12.1 Entry Point: PULL

4.127.12.2 Purpose:

Subroutine PULL is used to extract a BCD string from the middle of a larger BCD string. The location of the first character and the number of characters to be extracted are variables.

4.127.12.3 Calling Sequence:

CALL PULL (BCDA,OUT,ICH,NCH,FLAG)

MODULE FUNCTIONAL DESCRIPTIONS

The variables in PULL are the same as in subroutine PUSH (4.127.12) except that BCDA is only input and ØUT is the BCD string of output characters. (FLAG is not used.)

4.127.12.4 Method:

The method is similar to subroutine PUSH except that no integer conversion is performed and the operation is reversed.

2.127.13 Design Requirements

1. Uses one scratch file (301)
2. Open core at ASDZZZ will contain the list of active data blocks (2 BCD words per data block). Three buffers are located at the bottom.
3. Interfaces with /SYSTEM/

<u>Word</u>	<u>Description</u>
1	Buffer size
2	Print file number
3	NØGØ flag
4	Input file
21	Approach code
39	Number of bits per character
40	Number of bits per word
41	Number of characters per word
55	Matrix precision flag
69	Control flag for ALTER output

4.127.14 Diagnostic Messages

ASDMAP produces Messages 6001-6011, 6015, 6016, 6232, 6301-6304, and 6510. The NØGØ flag is set for any fatal error, but the execution continues. Only if the preface has been misprogrammed to cause a bad file setup, will the module exit.

FUNCTIONAL MODULE CØMB1 (SUBSTRUCTURE COMBINATION, STEP 1)

4.128 FUNCTIONAL MODULE CØMB1 (SUBSTRUCTURE COMBINATION, STEP 1)

4.128.1 Entry Point: CØMB1

4.128.2 Purpose

To combine an arbitrary set of basic or pseudostructures to form a new pseudostructure. User flexibility is enhanced by allowing for either manual or automatic linking of substructures, or a combination of both. Maximum versatility is assured by the ability to link individual component degrees of freedom between substructures enabling adequate representation of even the most complex of structural connectivities.

4.128.3 DMAP Calling Sequence

CØMB1 CASECC,GEØM4//C,N,STEP/V,N,DRY \$

4.128.4 Input Data

4.128.4.1 GINØ Data Blocks

CASECC - Case Control for substructuring

GEØM4 - Substructuring bulk data images

4.128.4.2 SØF Items

EQSS - Substructure equivalence tables

BGSS - Grid point table

LØDS - Load set data

CSTM - Coordinate system transformation matrices

PLTS - Plot set data

These items refer to the pseudostructures being combined.

4.128.5 Output Data

4.128.5.1 GINØ Data Blocks

None

4.128.5.2 SØF Items

EQSS, BGSS, CSTM, LØDS, PLTS - These are described above (See 4.128.4.2)

HØRG - H transformation matrix relating the degrees of freedom of the pseudostructure to the combined structure.

MODULE FUNCTIONAL DESCRIPTIONS

These items pertain to the newly created combined structure.

4.128.6 Parameters

- STEP - Input, integer. Step number of substructure command.
- DRY - Output, integer. Controls execution status of operation.

4.128.7 Method

This section discusses the logical flow of the CØMB1 module so that a clear overview of the substructure combination process may be obtained. Each step represents a separate routine, the details of which will be found in Sec. 4.128.8.

The communication of data among the routines has been facilitated by specially defined auxiliary files. The generation of the general file structures is included herein.

The following processing sequence will be maintained when combining substructures:

1. New grid point definitions specified on GNEW bulk data cards are added to the substructure data files.
2. CØNCT bulk data cards are processed. Connections given by this data will augment and, if conflicting, will override automatic connections.
3. If the automatic option is elected, matching grid points are found and connection definitions are generated internally.
4. The CØNCT1 data is processed. This data has overall priority and specifications here will supercede any connection resulting from 2, 3 or 4 above. In particular, degrees of freedom connected by CØNCT1 data may not be disconnected through the use of a RELES card.
5. The RELES data is applied to the combined set of connections defined in 2 and 4 above.

Two terms which pervade the logic of this module and are intrinsic to an understanding of the handling of connection data will now be defined:

1. I_p - Internal Point - Number

The internal point number is the key to accessing data for a given substructure from the SØF. Every basic substructure or pseudostructure on the SØF has a set of internal point numbers which are essentially used to identify degrees of freedom within that structure. In the case of basic substructures the I_p will refer to a grid point, whereas, in the case of a

FUNCTIONAL MODULE COMB1 (SUBSTRUCTURE COMBINATION, STEP 1)

pseudostructure, an I_p may correspond to some subset of the component degrees of freedom of a physical grid point. (The latter implies that several I_p may represent one grid point.)

2. Connection Entry

A connection entry is a set of words that defines precisely the connectivity relations between particular I_p numbers for connected substructures. The form of this variable length record is:

$$C/CC/I_{p1}/I_{p2}/\dots/I_{p7}$$

where

- C Is the component degree of freedom code defining those components that are to be connected.
- CC Is the connection code where connected substructures have been ordered as they appear on the COMBINE instruction of the Case Control Deck. The command COMBINE (OPTIONS) A, B, ..., G defines a seven bit word from the following correspondence:

A ↔ 1	B ↔ 2	C ↔ 4
D ↔ 8	E ↔ 16	F ↔ 32
G ↔ 64		

Then, each set of connected substructures is uniquely identified by the bit pattern created by adding the codes for the component substructures, i.e.,

A + B ↔ 3	0000011	
B + E ↔ 18	0010010	etc.

- I_{pi} The internal point number of the i^{th} substructure of a set of points to be connected through degrees of freedom given in C. The order of appearance of the I_{pi} in the connection entry follows the same order as the substructures were given on the COMBINE instruction.

Figure 1 illustrates the flow of the module. The small figures in the corner of a block refer to the step in the following outline. Below the block is the reference to the section detailing its computational flow.

The steps in the COMB1 module are:

1. Initialization of both GIN0 and S0F files and I0 buffers.

MODULE FUNCTIONAL DESCRIPTIONS

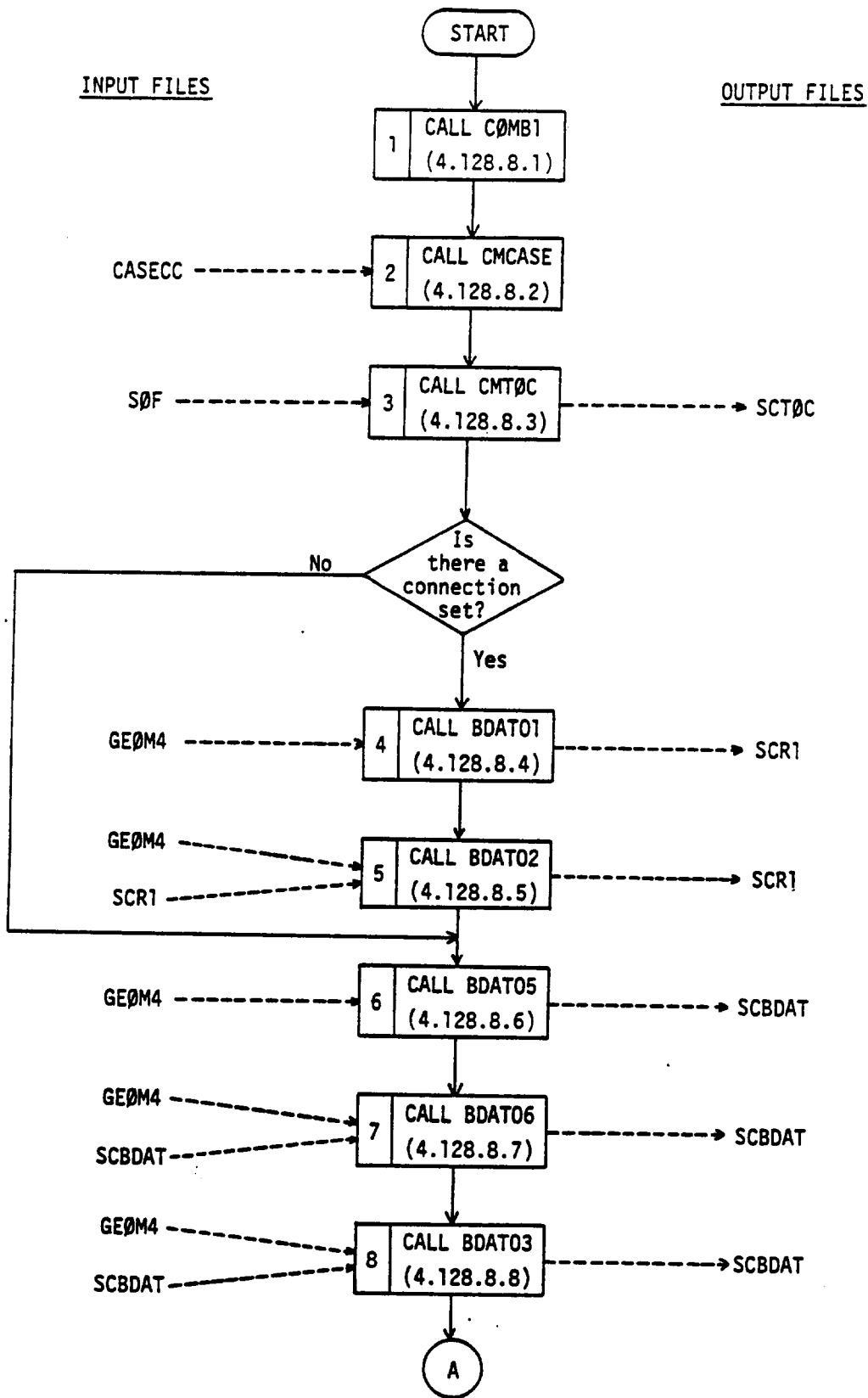


Figure 1a. Logical flow of Module COMB1.

FUNCTIONAL MODULE CØMB1 (SUBSTRUCTURE COMBINATION, STEP 1)

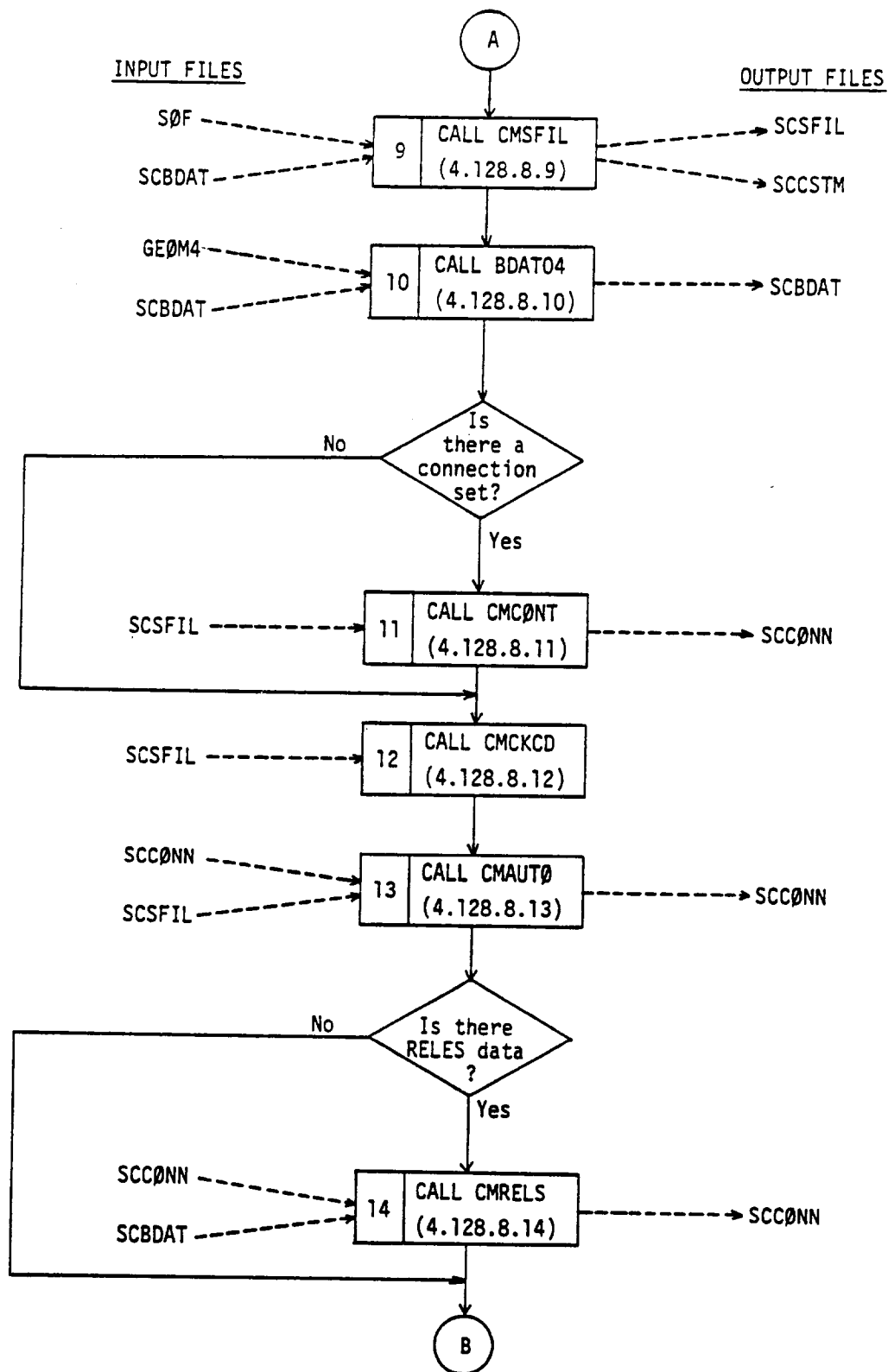


Figure 1b. Logical flow of Module CØMB1 (Cont'd)

MODULE FUNCTIONAL DESCRIPTIONS

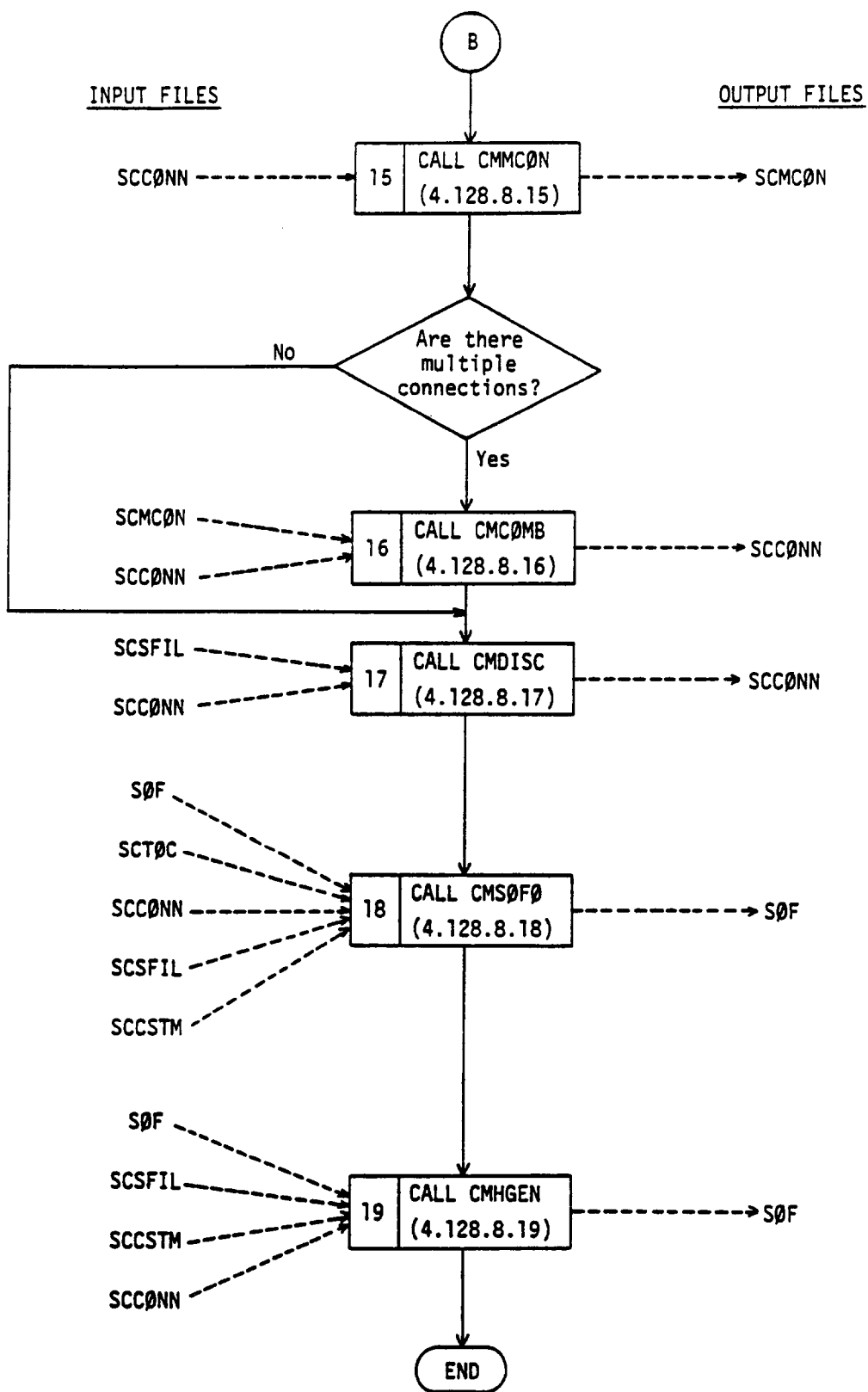


Figure 1c. Logical flow of Module C0MB1 (Concluded)

FUNCTIONAL MODULE CØMB1 (SUBSTRUCTURE COMBINATION, STEP 1)

2. Obtains all relevent user-supplied Case Control data specifying the current combine operation.
3. Generates a table of contents of all pseudostructure names that will be combined and their component substructure names. (Placed on file SCTØC.)
4. Processes CØNCT1 Bulk Data defining connections between three or more basic substructures in terms of local grid point identification numbers. (Placed on file SCR1.)
5. Processes CØNCT Bulk Data defining pairwise connections between basic substructures in terms of local grid point identification numbers. (Placed on file SCR1.)
6. Processes GTRAN Bulk Data defining transformations to a selected grid point in a given basic substructure. (Placed in the first logical record of file SCBDAT.)
7. Processes GNEW Bulk Data defining new grid points at any stage of substructuring. (Placed in the second logical record of file SCBDAT.)
8. Processes TRANS Bulk Data defining coordinate transformations to be applied to given pseudostructures and ID's of TRANS sets referenced by GTRAN data. (Placed in the third logical record of file SCBDAT.)
9. Generates a copy of the SØF data items EQSS and BGSS for the particular pseudostructures being combined. The coordinates are transformed to the overall system, as are the component degrees of freedom. Also, in the case of combining pseudostructures that are themselves a result of an earlier combination, codes are added to locate grid points that may be represented by multiple I_p numbers. (Processed data is written on file SCSFIL.)
10. Processes the RELES Bulk Data which prevents specified degrees of freedom from being connected. Data is written out in terms of I_p numbers rather than gridpoint ID's. (Placed in the fourth logical record of file SCBDAT.)
11. Processes the connection entries defined in steps 4 and 5 above translating the grid-point identification numbers as given by the user into the I_p numbers used by this module. (Processed data is written on file SCCØNN.)
12. If any combination has been specified by either CØNCT1 or CØNCT Bulk Data, the geometric consistency of the requested connection is checked and grid points not within the specified tolerance range are flagged as errors.

MODULE FUNCTIONAL DESCRIPTIONS

13. Generates connection entries when the automatic combine option has been specified. These connection entries are found in terms of I_p numbers. (Processed data is merged with data already on file SCCØNN.)
14. Enforces any RELES data that may be given and updates the connection entries. (Places updates back on file SCCØNN.)
15. Determines whether multiple connection, i.e., two or more connection entries defining connections involving the same I_p number, exist due to use of the automatic option or through redundant specifications during a manual combine. (List of multiple connections is written on file SCMCØN.)
16. If multiple connections exist from step 15, they are processed and combined in order to eliminate the redundancy of I_p numbers. (Final list of connection entries is written on file SCCØNN.)
17. Determines the disconnected degrees of freedom generating "Disconnection Entries" and merges these with the connection entries. This combined set is then the final definition of the degrees of freedom in the combined structure. (Final list is written on file SCCØNN.)
18. The data for the resultant structure is processed and a new SØF entry created. All SØF items are then generated and written.
19. The [H] transformation matrices relating the degrees of freedom of each component pseudostructure to the resultant structure are assembled and written on the SØF.

4.128.8 Subroutines

4.128.8.1 Subroutine Name: CØMB1

1. Entry Point: CØMB1
2. Purpose: CØMB1 module driver and initializer of GINØ and SØF files and IØ buffers.
3. Calling Sequence: CALL CØMB1

4.128.8.2 Subroutine Name: CMCASE

1. Entry Point: CMCASE
2. Purpose: To read the user-supplied case control data and assemble the parameters used to control the CØMBINE operation.
3. Calling Sequence: CALL CMCASE

CØMMØN/CMBØØ1/

CØMMØN/CMBØØ2/

CØMMØN/CMBØØ3/

CØMMØN/CMBØØ4/

See Sec. 4.128.9 for description of labeled common blocks

4. Method: There are eleven acceptable case control mnemonics for the CØMB1 module, the processing of each is described:

- a. ØPTS - If the specified option is AUTØ, the logical flag IAUTØ is set to .TRUE., otherwise, it is .FALSE.
- b. SØRT - Preferential sort direction for AUTØ option, for SØRT = X, Y or Z, ISØRT is set to 1, 2 or 3 respectively, the default option is X.
- c. NAMC - The names of the structures being combined are stored in the array CØMBØ (See note below).
- d. NAMS - The name of the resultant combined structure is put in the two word array CNAM.
- e. TØLE - The value of the geometric tolerance on connections is placed in the variable TØLER.
- f. CØNN - The connection set ID is stored in the variable CØNSET; also CØNECT is set to .TRUE.

MODULE FUNCTIONAL DESCRIPTIONS

- g. CØMP - The component names will be searched against the NAMC and input will be stored in CØMBØ (See note below).
- h. TRAN - A specified TRANS set ID. The logical variable TRAN is set to .TRUE.
- i. SRCH - Control on the substructure to be searched during any AUTØ connect. The array RESTCT(KK,L) is set to 1 if KK is to be searched against L, otherwise, 0.
- j. SYMT - A symmetric reflection of a structure about a given axis or set of axes. Given by the variable SYMT defining a bit pattern of the reflection to be made, i.e., 001, about X; 101 about X and Z; etc.
- k. ØUTP - A user control on selected output during the CØMBINE process. A logical bitwise "OR" function is performed between all ØUTP values in case control.

Note: The array CØMBØ (7,5) contains controls for each pseudostructure (to a maximum of seven). The first two words in each row are the pseudostructure names (NAMC). For each of these names that is also a specified component (CØMP), the third and fourth words are the TRAN and SYMT values, if given. The fifth word in each row is set in subroutine CMTØC (See Sec. 4.128.8.3) to the number of component substructures contained in the pseudostructure (NAMC).

4.128.8.3 Subroutine Name: CMTØC

- 1. Entry Point: CMTØC
- 2. Purpose: To generate a table of contents of the pseudostructures being combined and their respective component substructures.
- 3. Calling Sequence: CALL CMTØC

CØMMØN/CMBØØ1/

CØMMØN/CMBØØ2/

CØMMØN/CMBØØ3/

CØMMØN/CMBØØ4/

See Sec 4.128.9 for description of labeled common blocks.

- 4. Method: Each of the pseudostructures that have been specified on the CØMBINE card may have many component basic substructures all of which may be referenced in the bulk data deck. Each of these names is accessed on the SØF, and a table of contents of component names is generated and written on scratch file SCTØC. The organization of this file is:

FUNCTIONAL MODULE CØMB1 (SUBSTRUCTURE COMBINATION, STEP 1)

<u>Word</u>	<u>Description</u>
1, 2	Pseudostructure name
3	Number of components
4, 5	Name - component 1
6, 7	Name - component 2
....
....
....

EØR

A record of this format is written for each pseudostructure and then an end-of-file is written.

4.128.8.4 Subroutine Name: BDATØ1

1. Entry Point: BDATØ1

2. Purpose: To process the CØNCT1 bulk data and define connection entries from this data in terms of grid point identification numbers.

3. Calling Sequence: CALL BDATØ1

CØMMØN/CMBØØ1/

CØMMØN/CMBØØ2/

CØMMØN/CMBØØ3/

CØMMØN/CMBØØ4/

See Sec. 4.128.9 for description of labeled common blocks.

4. Method: The CØNCT1 bulk data is accessed from the GEØM4 input data block and processed in the following manner:

a. Each set of CØNCT1 data that references the connection set identification number given in case control is translated into a group of connection entries thusly;

(1) The component basic substructure names that appear in the data are found in the table of contents by the subroutine FINDER (See Sec. 4.128.8.21). The pseudostructure ID number and component ID number are returned.

(2) The degree of freedom code is transformed by subroutine ENCØDE to the bit pattern used by substructuring (See Sec. 4.128.8.19).

(3) Using the pseudostructure ID numbers, the connection code (defined in Sec.

MODULE FUNCTIONAL DESCRIPTIONS

4.128.7) is computed and multiplied by -1 so that in the later processing of RELES data these entries will be unaffected.

(4) The grid point ID numbers are keyed to the basic substructure to which they belong by:

$$\text{CODED GRID ID} = \text{COMPONENT NO.} * 10^6 + \text{GRID ID}$$

b. The connection entries are generated and written on scratch file SCR1, one per logical record. The flag TDAT(1) is set to .TRUE. signifying the existence of CØNCT1 data in the current analysis.

4.128.8.5 Subroutine Name: BDAT02

1. Entry Point: BDAT02

2. Purpose: To process the CØNCT bulk data and define connection entries in terms of grid point identification numbers.

3. Calling Sequence: CALL BDAT02

CØMMØN/CMB001/

CØMMØN/CMB002/

See Sec. 4.128.9 for description of labeled common blocks.

CØMMØN/CMB003/

CØMMØN/CMB004/

4. Method: With only one exception, the processing of this data is identical with that of the CØNCT1 data (See Sec. 4.128.8.4). For this data, the connection code is NOT multiplied by -1 in order that RELES may be applied to them.

4.128.8.6 Subroutine Name: BDAT05

1. Entry Point: BDAT05

2. Purpose: To process the GNEW bulk data.

3. Calling Sequence: CALL BDAT05

CØMMØN/CMB001/

CØMMØN/CMB002/

See Sec. 4.128.9 for definition of labeled common blocks.

CØMMØN/CMB003/

CØMMØN/CMB004/

4. Method: The GNEW data is processed as follows:

- a. The component basic substructure names are treated as for BDAT01 (See Sec. 4.128.8.4).
- b. The degree of freedom number is transformed to a bit pattern.
- c. The grid point ID numbers are keyed to their component basic substructure number.
- d. The above data is processed for each GNEW card referencing the input connection set identification number and written on a temporary scratch file. When all the data has been processed, it is sorted on the pseudostructure number and written into one logical record in the first file of SCBDAT. (Note that for GNEW, GTRAN, TRANS, and RELES bulk data, if there is no data of the given type, an end-of-file mark is written on SCBDAT. Therefore, SCBDAT will always contain four files.) An end-of-file mark is then written.

4.128.8.7 Subroutine Name: BDAT06

1. Entry Point: BDAT06
2. Purpose: To process the GTRAN bulk data.
3. Calling Sequence: CALL BDAT06

CØMMØN/CMB001/

CØMMØN/CMB002/

See Sec. 4.128.9 for definition of labeled common blocks.

CØMMØN/CMB003/

CØMMØN/CMB004/

4. Method: GTRAN data is processed thusly:

- a. The basic substructure name is processed as for BDAT01 (See Sec. 4.128.8.4).
- b. The reference TRANS set ID is unchanged.
- c. The grid point ID number is coded to the component substructure number.
- d. The GTRAN TRANS set ID is unchanged but a list of ID numbers is placed in CØMMØN to be passed to the TRANS bulk data routine.
- e. The above steps are carried out for each GTRAN card that references any of the TRANS set ID's given in case control. The data is sorted on the pseudostructure number and written in one logical record in the second file of SCBDAT and an end-of-file mark is written.

MODULE FUNCTIONAL DESCRIPTIONS

4.128.8.8 Subroutine Name: BDAT03

1. Entry Point: BDAT03
2. Purpose: To process TRANS bulk data cards and generate geometric transformation matrices.
3. Calling Sequence: CALL BDAT03

CØMMØN/CMBØØ1/

CØMMØN/CMBØØ2/

See Sec. 4.128.9 for definition of labeled common blocks.

CØMMØN/CMBØØ3/

CØMMØN/CMBØØ4/

CØMMØN/CMBZZZ/

Open core common block containing TRANS set ID numbers for GTRAN input.

4. Method:

- a. For each TRANS ID specified in case control or referenced by a GTRAN data card, the thru points defining the coordinate system are read from the GEØM4 input data block.
- b. The transformation matrix, T, is computed, and along with the coordinates of the new origin are written into the third file of SCBDAT in the same form as the NASTRAN data block CSTM. The form of this block is:

<u>Record</u>	<u>Word</u>	<u>Item</u>	
0		Header Record	
1	1	TRANS system ID number	} Repeated for each coordinate system
	2	1.0	
	3-5	x_0, y_0, z_0 (Coordinates of origin)	
	6-14	$T_{11}, T_{12}, T_{13}, T_{21}, T_{22}, T_{23}, T_{31}, T_{32}, T_{33}$	
2		End-of-file	

4.128.8.9 Subroutine Name: CMSFIL

1. Entry Point: CMSFIL
2. Purpose: To generate an updated file, SCSFIL, containing the SØF items EQSS and BGSS for the pseudostructures being combined in the current operation after all required geometric and local coordinate system transformations have been applied. Also, to generate new local coordinate system transformation matrices and the [H] matrix used to relate the final combined degrees of freedom to the degrees of freedom in each pseudostructure.

3. Calling Sequence: CALL CMSFIL

CØMMØN/CMBØØ1/

CØMMØN/CMBØØ2/ See Sec. 4.128.9 for the definition of labeled common blocks.

CØMMØN/CMBØØ3/

4. Method: The logical flow of subroutine CMSFIL is given in Figure 2. Steps that are lettered in the flowchart are further explained below:

a. Two arrays, each of length equal to the number of internal points in the I^{th} pseudostructure, will be used to store, respectively, the new CID's after they are renumbered internally, and the pointer to the [H] matrix for the given internal point of that pseudostructure.

b. A call to subroutine GTMAT1 (Sec. 4.128.8.28) locates the requested TRANS geometric transformation matrix, $[TT]_{3 \times 3}$, computes the symmetric reflection matrix, $[SYM]_{6 \times 6}$, and returns the final transformation matrix defined by:

$$[TT6]_{6 \times 6} = \begin{bmatrix} [TT] & \\ & [TT] \end{bmatrix} [SYM]$$

The values of the geometric coordinates $\{x,y,z\}$ are transformed by

$$\begin{Bmatrix} x \\ y \\ z \end{Bmatrix}_{\text{NEW}} = [TT]_{3 \times 3} \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}_{\text{OLD}} + \begin{Bmatrix} x_0 \\ y_0 \\ z_0 \end{Bmatrix}$$

where the coordinates $\{x_0, y_0, z_0\}$ specify the origin of the old pseudostructure in the basic coordinate system of the new pseudostructure.

c. The transformation matrix to be applied to the component degrees of freedom depends on the possible combinations of transformations specified by the user. These include the geometric transformations specified by TRANS, $[TT]$, pseudostructure local transformations given by $[T_G]$, and the possibility of substructure local grid point coordinate systems (CSTM), $[T_C]$. The following table defines the overall application of these transformations and their effect on local coordinate system identification:

MODULE FUNCTIONAL DESCRIPTIONS

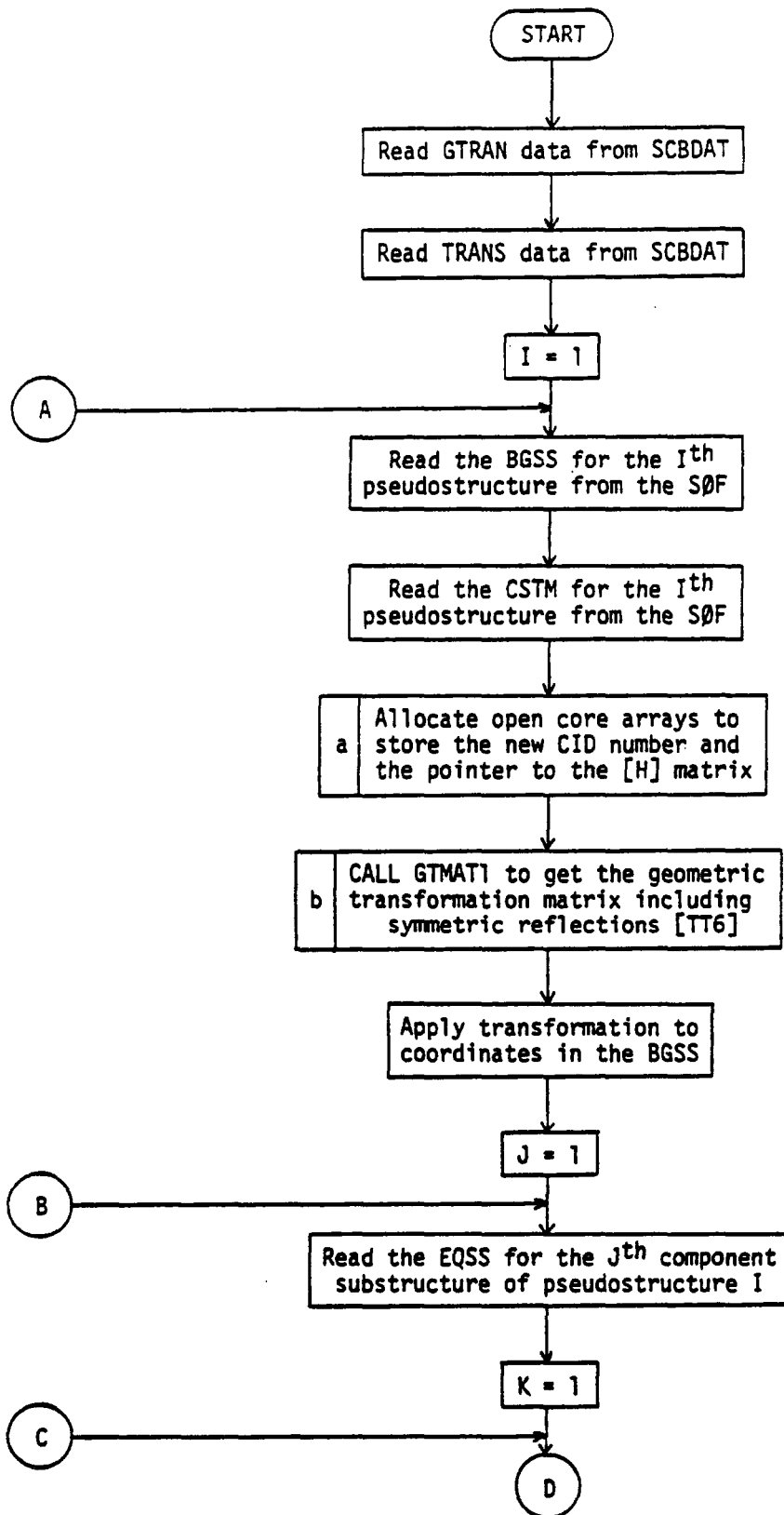


Figure 2a. Logical flow of Subroutine CMSFIL.

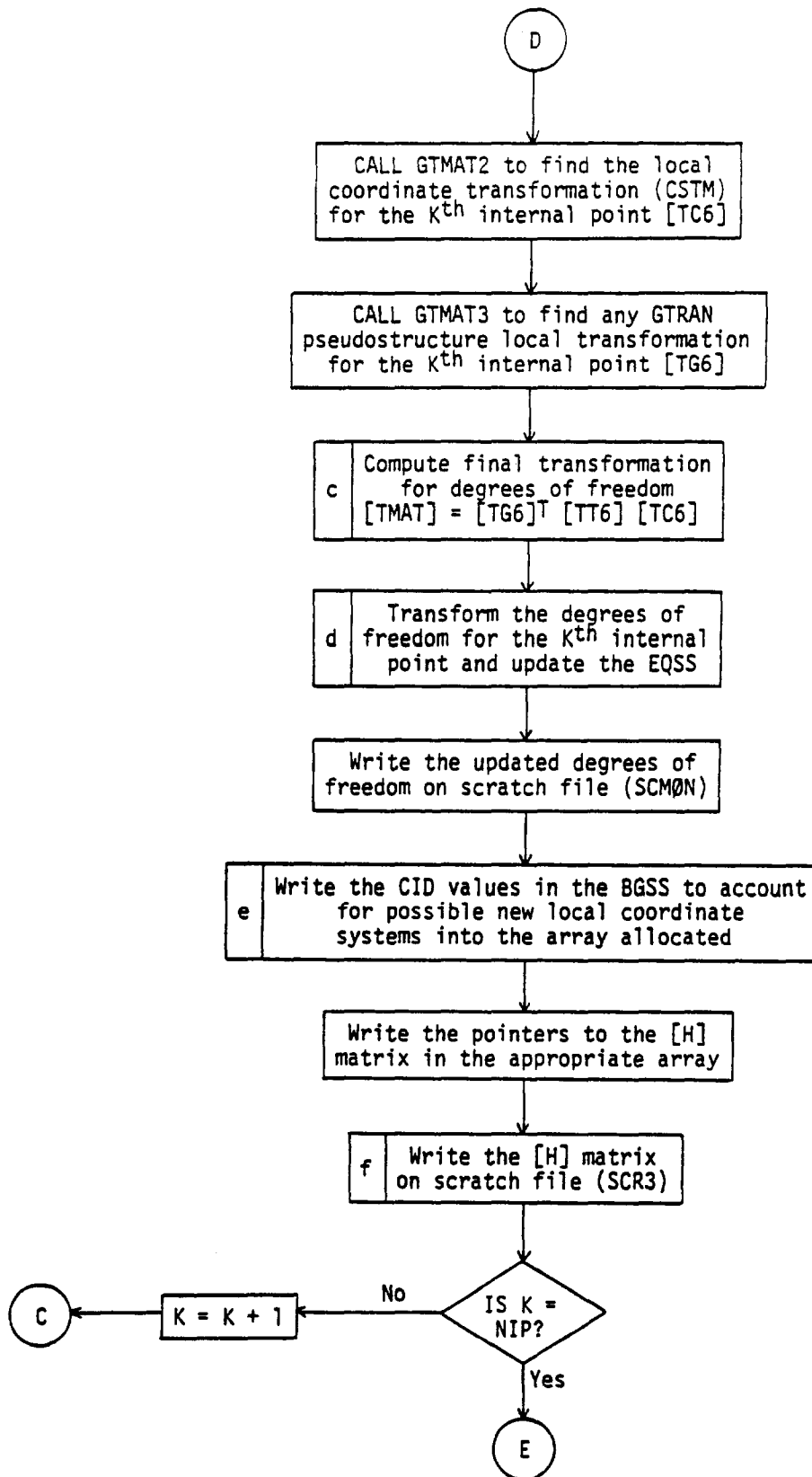
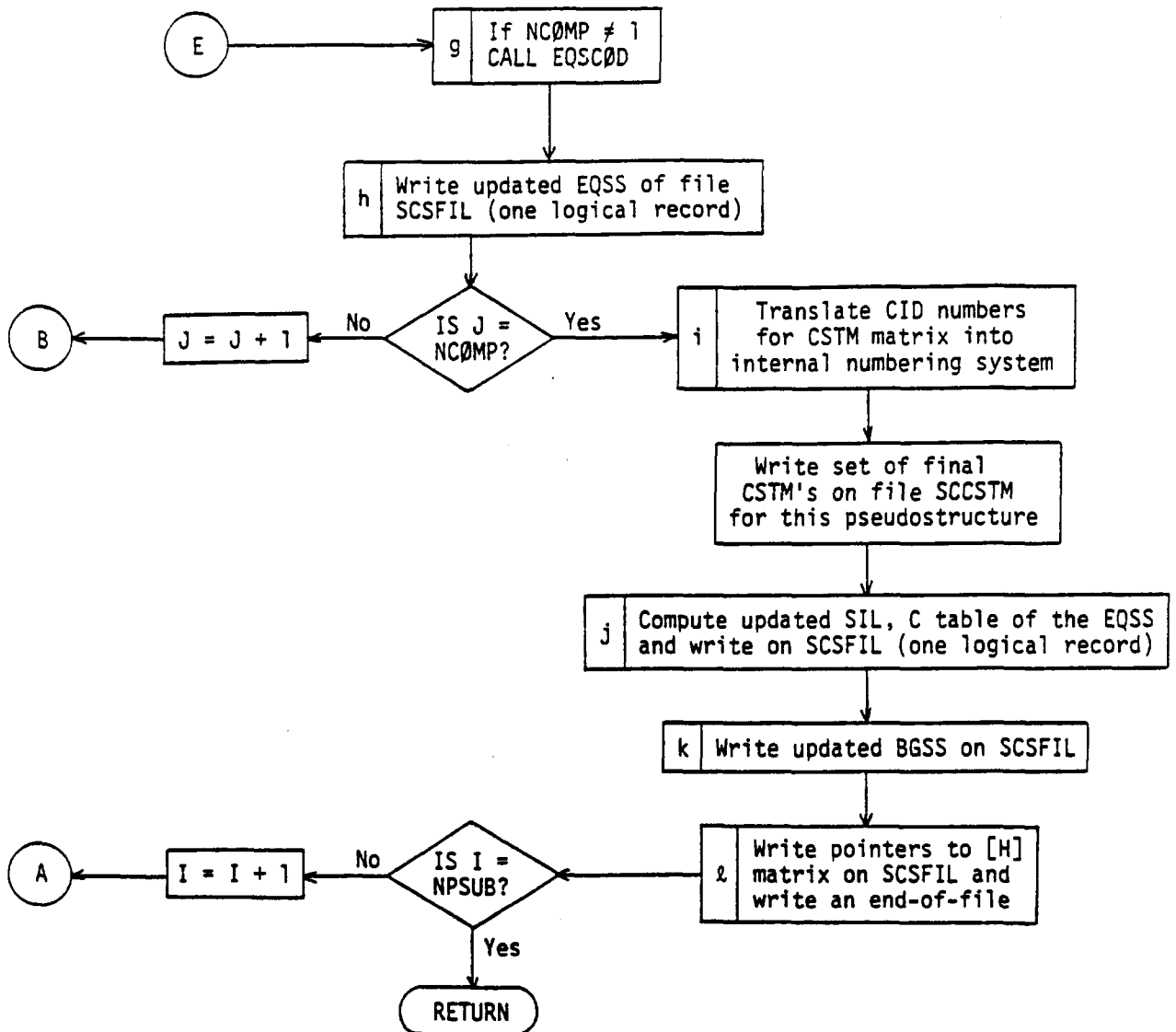


Figure 2b. Logical flow of Subroutine CSMFIL (Cont'd)

MODULE FUNCTIONAL DESCRIPTIONS



NOMENCLATURE:

- I - Counter for pseudostructure
- J - Counter for component substructures of a given pseudostructure
- K - Counter for internal point numbers (I_p)
- TRAN(I) - A TRANS set ID for I
- SYMT(I) - The symmetric reflection for I
- NPSUB - The number of pseudostructures being combined
- NCØMP - The number of component substructures in a given pseudostructure
- NIP - The number of internal points

Figure 2c. Logical flow of Subroutine CTSFIL (Concluded)

FUNCTIONAL MODULE COMB1 (SUBSTRUCTURE COMBINATION, STEP 1)

		No GTRAN	TRAN = 0	$T_G \neq TT$	$T_G = TT$
No Grid Point Local System CID = 0	[TMAT]	[TT]	[TT]	$[T_G]^T [TT]$	[I]
	$[T_C]_{NEW}$	None	None	$[T_G]$ Add new CID	$[T]$ Add new CID
Grid Point Local System CID \neq 0	[TMAT]	[I]	$[TT][T_C]$	$[T_G]^T [TT][T_C]$	$[T_C]$
	$[T_C]_{NEW}$	$[TT][T_C]_{OLD}$	SET CID = 0	$[T_G]$ Add new CID	$[T_G]$ Add new CID

Note that all transformations in the above table are of the form

$$[TMAT] = [T_G]^T [TT] [T_C]$$

This is the computation made with appropriate matrices some of which may be identity.

d. The old component degrees of freedom, $\{\delta_{OLD}\}$, are transformed by

$$\{\delta_{NEW}\} = [TMAT] \{\delta_{OLD}\}$$

Since $\{\delta\}$ are boolean vectors, after the product is computed, the $\{\delta_{NEW}\}$ are determined by

$$\begin{aligned} \delta_{NEW} &= 1 & \text{if} & & |\delta_{NEW}| > 10^{-4} \\ &= 0 & \text{if} & & |\delta_{NEW}| < 10^{-4} \end{aligned}$$

The new degrees of freedom and I_p number are then written on scratch file SCMCØN.

e. The set identifier for a local coordinate system is written into the array provided for in (a). If the number entered refers to a TRANS ID it is entered as a negative number for later processing.

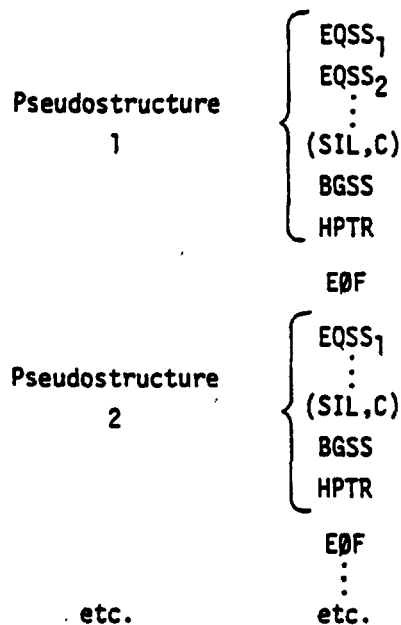
f. The final transformation computed corresponds to the initial [H] matrix for a given internal point. These matrices are written to scratch file SCR3 and a pointer to them stored in the array provided in (a).

g. If a pseudostructure contains more than one component substructure the possibility that a grid point is represented by several I_p numbers exists. Subroutine EQSCØD (See Sec. 4.128.8.20) is called, in such cases, to process such grid points.

MODULE FUNCTIONAL DESCRIPTIONS

- h. The EQSS for each component substructure is updated by writing the transformed degrees of freedom over the untransformed input values and is then written on SCSFIL in one logical record.
- i. After all component substructures of the I^{th} component substructure are processed, the CID values are renumbered internally to include all newly generated local systems and are written on file SCCSTM. Also, the actual values of CID in the BGSS are updated.
- j. The I_p and transformed degree of freedom codes written on scratch file SCMCØN in (d) above are sorted on I_p number and the new (SIL,C) group of the EQSS is computed and written on SCSFIL, in one logical record for each pseudostructure.
- k. The BGSS item which has been updated according to the geometric and symmetric transformations is written on SCSFIL, in one logical record, for each component substructure.
- l. The array containing pointers to the [H] matrix, HPTR, is written on SCSFIL, in one logical record, and an end-of-file mark is written.

The data structure for SCSFIL is then:



4.128.8.10 Subroutine Name: BDAT04

- 1. Entry Point: BDAT04
- 2. Purpose: To process the RELES bulk data input.
- 3. Calling Sequence: CALL BDAT04

FUNCTIONAL MODULE CØMB1 (SUBSTRUCTURE COMBINATION, STEP 1)

CØMMØN/CMBØØ1/

CØMMØN/CMBØØ2/ See Sec. 4.128.9 for the definition of labeled common blocks.

CØMMØN/CMBØØ3/

4. Method: For each group of RELES data that references the connection set identification number specified in case control, the following processing is done:
 - a. The component basic substructure names are processed as before (See Sec. 4.128.8.4).
 - b. The grid point ID number and degree of freedom code are read and processed as follows:
 - (1) The degree of freedom code is transformed to the bit pattern via subroutine ENCØDE (See Sec. 4.128.8.24).
 - (2) A call is made to FNDGRD (See Sec. 4.128.8.23) for the given grid point ID to determine the set of I_p numbers representing it.
 - c. The releases specified are applied to the degree of freedom codes returned from FNDGRD and all I_p , CRELES pairs are written in one logical record in the fourth file of bulk data file SCBDAT.

4.128.8.11 Subroutine Name: CMCØNT

1. Entry Point: CMCØNT
2. Purpose: To take the set of connection entries that were generated in subroutine BDATØ1 and BDATØ2 (See Secs. 4.128.8.4 and 5) in terms of grid point ID numbers and generate a corresponding set of connection entries in terms of I_p numbers.

3. Calling Sequence: CALL CMCØNT

CØMMØN/CMBØØ1/

CØMMØN/CMBØØ2/ See Sec. 4.128.9 for the definition of labeled common blocks.

CØMMØN/CMBØØ3/

4. Method: The logical flow of subroutine CMCØNT is presented in Figure 3. Lettered steps in the flowchart are explained in greater detail below:
 - a. The connection entries in terms of grid point ID numbers are found on scratch file SCR1. A second scratch file, SCR2, is used to store the updated connection entries. These two files will be 'ping-ponged,' first one is the input file and one the output file, then the roles of the files are reversed until all the data is process.

MODULE FUNCTIONAL DESCRIPTIONS

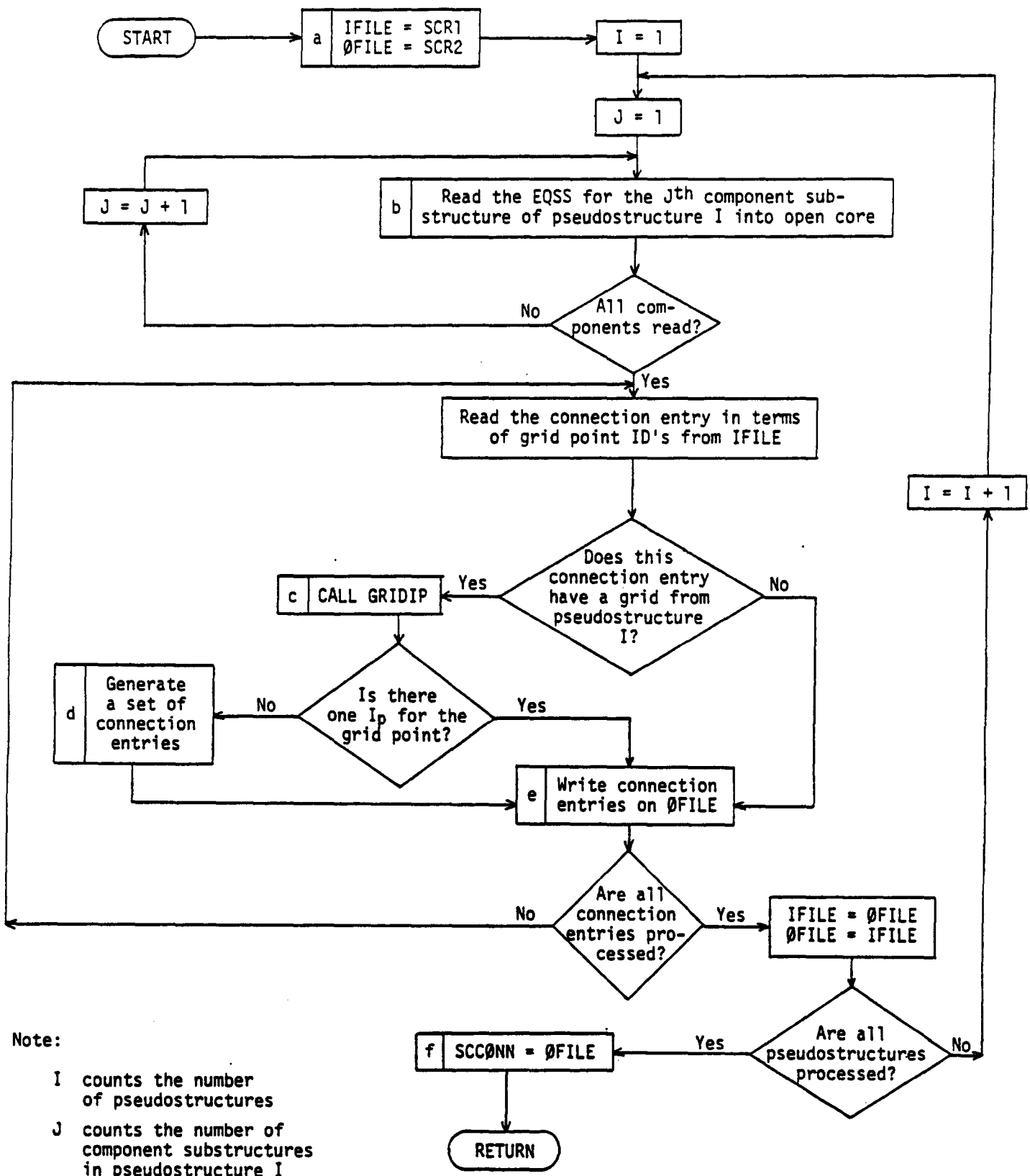


Figure 3. Logical flow of Subroutine CMC0NT.

FUNCTIONAL MODULE C0MB1 (SUBSTRUCTURE COMBINATION, STEP 1)

- b. The EQSS for each component substructure of a given pseudostructure is read into open core from SCSFIL and a list of pointers to the first address in each EQSS is kept.
- c. Subroutine GRIDIP (See Sec. 4.128.8.22) is called with each grid point identification number taken from the connection entry to determine whether the grid point is represented by several I_p numbers.
- d. If a grid point is represented by several I_p numbers, a set of connection entries is generated replacing the grid ID by each I_p and defining the component codes in terms of the I_p . As a clarifying example, consider the connection entry

111111/3/101/203

and suppose that grid point 101 of pseudostructure 1 is represented by the following I_p numbers

<u>I_p</u>	<u>C</u>
137	110000
152	001110
179	000001

Then, the first time through the loop with $I = 1$, the following new connection entries would be generated:

110000/3/137/203
001110/3/152/203
000001/3/179/203

- e. If the connection entry contains a grid point from the I^{th} pseudostructure, the grid ID is replaced by the I_p number and the connection entry written on ØFILE. In the case of multiple I_p numbers, the resulting set of connection entries is written. If the connection entry does not contain a grid from the I^{th} pseudostructure, the entry is simply written onto ØFILE as it is.
- f. When processing is complete, the last ØFILE is renamed SCCØNN and contains the complete collection of connection entries that have been manually specified.

4.128.8.12 Subroutine Name: CMCKCD

1. Entry Point: CMCKCD
2. Purpose: To check that the manually-specified connections are within the allowable geometric tolerance.

MODULE FUNCTIONAL DESCRIPTIONS

3. Calling Sequence: CALL CMCKCD

CØMMØN/CMBØØ1/

CØMMØN/CMBØØ2/ See Sec. 4.128.9 for the definition of labeled common blocks.

CØMMØN/CMBØØ3/

4. Method:

- a. The BGSS file of each pseudostructure is read into open core from file SCSFIL.
- b. The manually-specified connection entries from file SCCØNN are read and the coordinates for each I_p number contained in the entry are checked against the coordinates for all other I_p numbers in the entry. All pairwise connections must be within TØLER (user input) units of one another or an invalid connection has been made. Such connections are flagged as errors.
- c. If the complete set of connection entries is allowable, the analysis continues, otherwise,
- d. If errors have been found, the connection entries that are incorrect are listed for the user and the module execution is terminated after all checks have been made.

4.128.8.13 Subroutine Name: CMAUTØ

1. Entry Point: CMAUTØ

2. Purpose: To generate connection entries automatically between pseudostructures.

3. Calling Sequence: CALL CMAUTØ

CØMMØN/CMBØØ1/

CØMMØN/CMBØØ2/ See Sec. 4.128.9 for definition of labeled common blocks.

CØMMØN/CMBØØ3/

4. Method: The logical flow of subroutine CMAUTØ is shown in Figure 4. The algorithm used to find the connections is then shown in detail in Figure 5. Lettered steps in these flowcharts are described in detail below.

- a. CID is replaced by I_p in order to recover the I_p after the sort has taken place. (CID entries on SCSFIL are unaffected).
- b. The sort direction is chosen by the user in case control, the default option is the X direction.

FUNCTIONAL MODULE COMB1 (SUBSTRUCTURE COMBINATION, STEP 1)

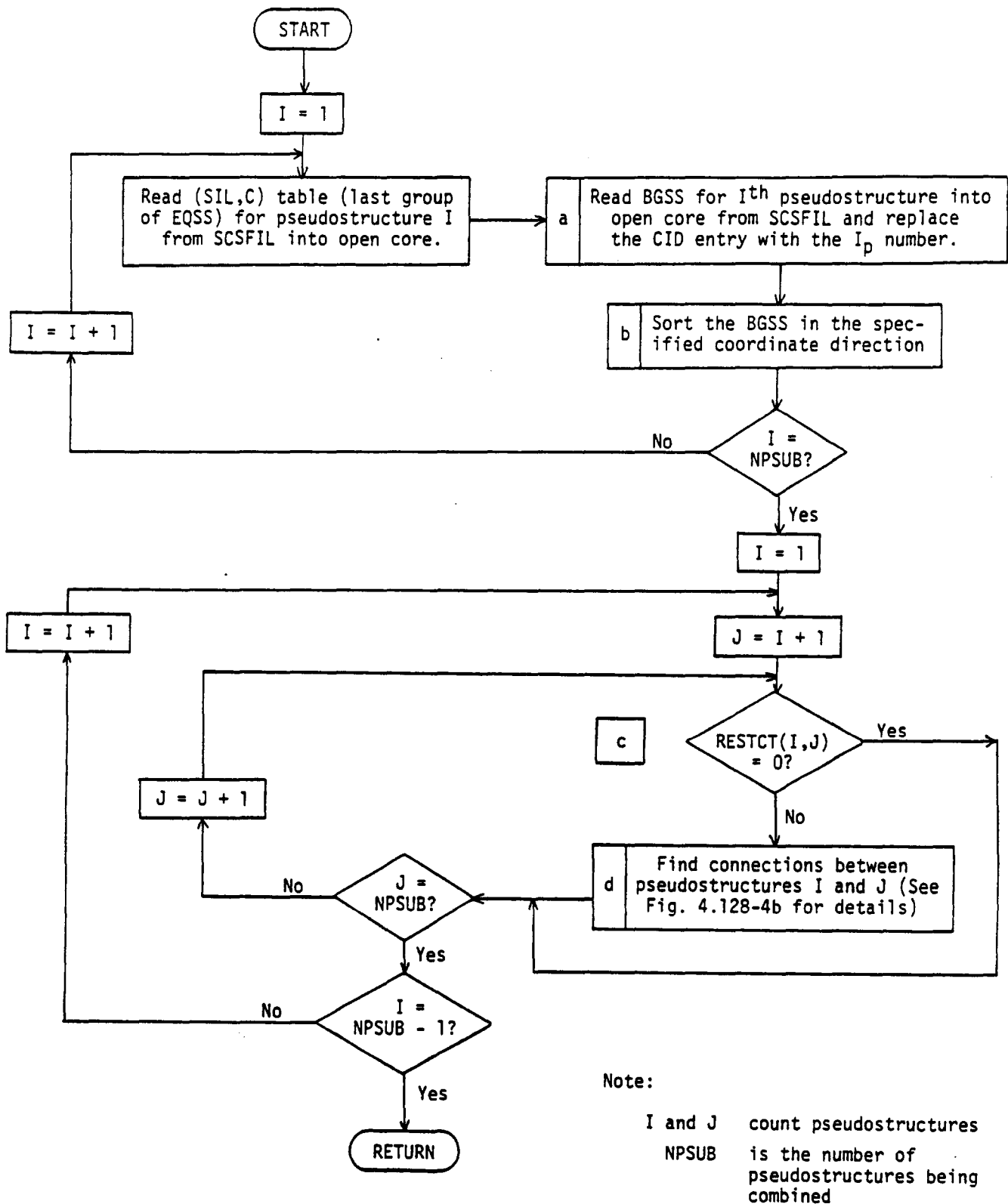


Figure 4. Logical flow of Subroutine CMAUT0.

MODULE FUNCTIONAL DESCRIPTIONS

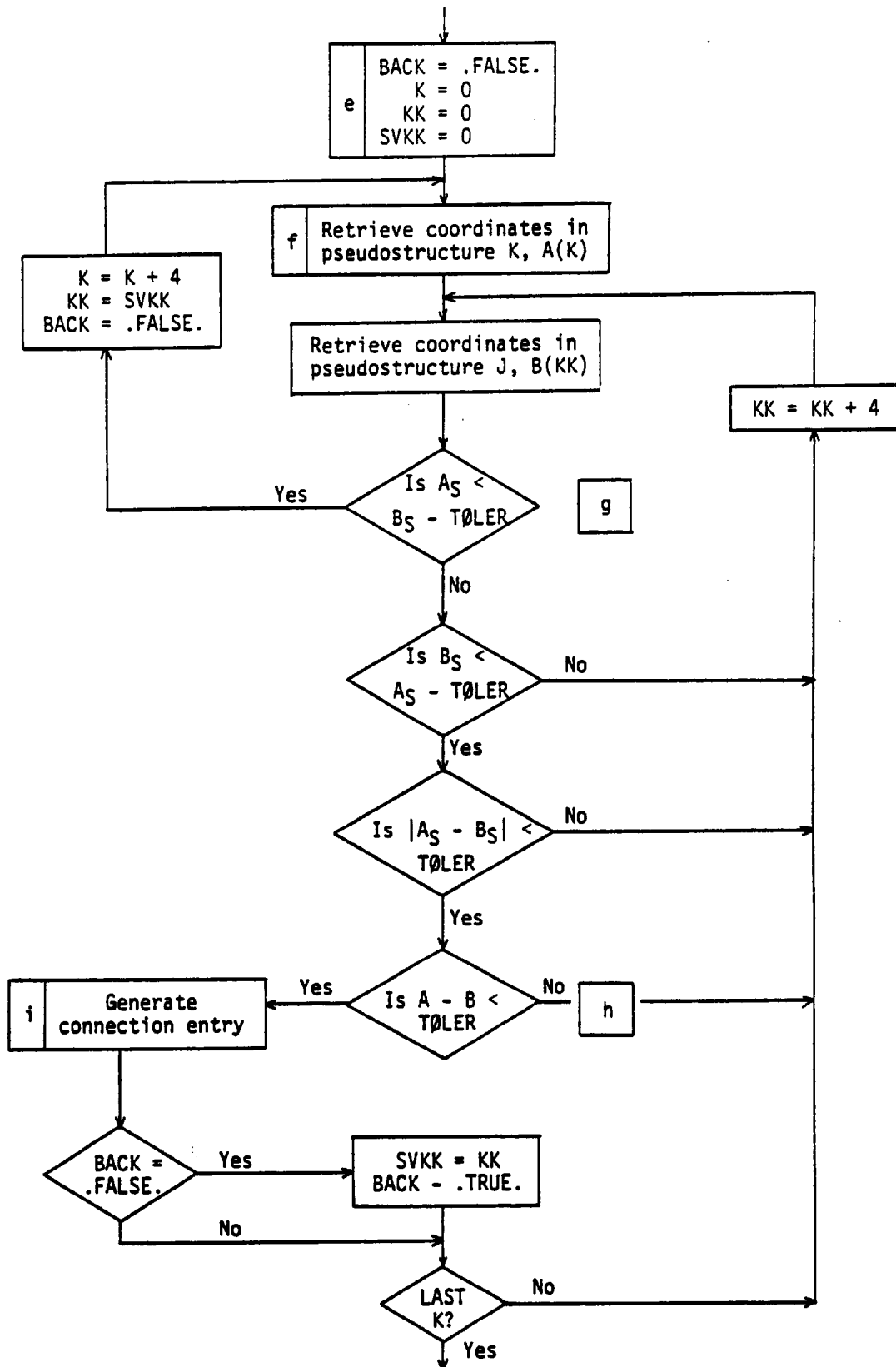


Figure 5. Location and Generation of Automatic Connection in Subroutine CMAUT0.

FUNCTIONAL MODULE COMB1 (SUBSTRUCTURE COMBINATION, STEP 1)

c. RESTCT(I,J) is a boolean array defining the combinations of substructures to be searched for automatic connections in order to increase the speed of this routine. The array is defined in subroutine CMCase (See Sec. 4.128.8.2) from SEARCH case control provided by the user.

d. To minimize the searching, the BGSS items for each pair of pseudostructures are searched in parallel, i.e., since each BGSS is sorted, the search is designed to be carried out only over the coordinates which are within close range of one another.

e. The definitions of these variables are:

K - Pointer into the current set of coordinates for pseudostructure I.

KK - Pointer into the current set of coordinates for pseudostructure J.

SVKK - A variable which saves the starting value of KK during a search for a match with coordinate K. A search never returns lower than SVKK for the next $K + 1$ sets of coordinates.

BACK - Is a logical flag used to define when to reset SVKK.

f. The notation A(K) and B(KK) refers to the set of coordinates found in the K^{th} and KK^{th} locations of the BGSS of the I^{th} and J^{th} pseudostructures, respectively.

g. The subscript S refers only to the preferred coordinate search direction. TOLER is the allowable dimensional discrepancy specified by the user.

h. All the remaining coordinate directions are tested.

i. The connection entries are generated such that:

(1) The component degrees of freedom for a set of matching points A and B are:

$$C = C_A \wedge C_B$$

implying connection will be made only at the intersection of degrees of freedom of the two points. (Note: The symbol \wedge represents the logical "and" function.)

(2) The connection code is

$$CC = 2^{I-1} + 2^{J-1}$$

(3) The two corresponding I_p numbers are taken directly from the core-held BGSS items.

MODULE FUNCTIONAL DESCRIPTIONS

(4) The connection entries are written on file SCCØNN following the manual connection data, if applicable, one logical record per entry.

4.128.8.14 Subroutine Name: CMRELS

1. Entry Point: CMRELS
2. Purpose: To apply user-specified RELES data to a given set of manually or automatically generated connection entries.
3. Calling Sequence: CALL CMRELS

CØMMØN/CMBØØ1/

CØMMØN/CMBØØ2/

See Sec. 4.128.9 for description of labeled common blocks.

CØMMØN/CMBØØ3/

CØMMØN/CMBØØ4/

4. Method:

- a. The RELES data that has been processed by routine BDATØ in terms of I_p numbers is read from SCBDAT into open core for each pseudostructure;

Pseudostructure 1	I_p	C
	⋮	⋮
	⋮	⋮
Pseudostructure 2	I_p	C
	⋮	⋮
	⋮	⋮
etc.	etc.	

A pointer to the first entry of the list for a given pseudostructure is kept along with the length of that list.

- b. The list for each pseudostructure is sorted on I_p number.
- c. The connection entries, in terms of I_p numbers from file SCCØNN, are read into open core following the data in step a.
- d. The connection code is decoded. If the entry was generated by CØNCT1 bulk data, no processing is done. Otherwise, the two connected I_p numbers are located in their corresponding (I_p , C) lists and the component degree of freedom code is updated by

$$C_{NEW} = C_{OLD} - (C_{OLD} \wedge C_{RELES})$$

FUNCTIONAL MODULE C0MB1 (SUBSTRUCTURE COMBINATION, STEP 1)

e. After processing has been completed for all connection entries, the updated set is written back on file SCC0NN.

4.128.8.15 Subroutine Name: CMMC0N

1. Entry Point: CMMC0N
2. Purpose: To determine whether multiply linked connections have been generated during an automatic combine operation or specified by the user in a manual operation.
3. Calling Sequence: CALL CMMC0N(NCE)

C0MM0N/CMB001/

C0MM0N/CMB002/ See Sec. 4.128.9 for description of labeled common blocks.

C0MM0N/CMB003/

NCE - The number of connection entries on file SCC0NN - integer - output.

4. Method: The complete set of connection entries is read into open core from file SCC0NN. The first word of each entry is overwritten with the connection entry ID number (CEID). This array is then sorted successively on the I_p number for each of the pseudostructures and a check is made of consecutive I_p numbers for that pseudostructure. A list is kept in open core of the values of I_p and the corresponding CEID each time a multiply listed I_p number occurs. If there are no multiple connections, the logical flag MC0N is set to .FALSE. and a return is made. If such connections do exist, they are sorted on CEID and the list of these connection entry ID numbers is written on file SCMC0N in one logical record.

4.128.8.16 Subroutine Name: CMC0MB

1. Entry Point: CMC0MB
2. Purpose: To merge all multiply linked connections into the smallest set of final connection entries.
3. Calling Sequence: CALL CMC0MB(NPS,NENT,ND0F,IC)

C0MM0N/CMB001/

See Sec. 4.128.9 for the definition of labeled common blocks.

C0MM0N/CMB002/

NPS - Number of columns allocated for array IC; corresponds to the number of pseudostructures being combined plus one - integer - input.

NENT - The number of connection entries to be processed - integer - input.

MODULE FUNCTIONAL DESCRIPTIONS

NDØF - The number of degrees of freedom allowed at each grid point - integer input.

IC - A dynamically dimensioned array, IC(NENT,NPS,NDØF), residing in open core, where the connection entries will be processed.

4. Method:

a. The connection entry ID numbers (CEID) defining the entries which represent multiple connections are read from file SCMCØN.

b. The CEID is a pointer into file SCCØNN to rapidly locate the connection entry. The appropriate entry is found and processed by the following steps:

(1) The array IC_{ij}^k is dynamically allocated with;

$i = 1, \dots, N_C$ N_C = Number of connection entries to be processed

$j = 1, \dots, N_S$ N_S = Number of pseudostructures being combined

$k = 1, \dots, 6$ Corresponds to the degree of freedom under consideration

(2) IC is then augmented with an additional column containing the connection code for the entry, i.e.,

$$[IC_{ij}^k \mid CC_i]$$

This will be used to expedite the recoupling of degrees of freedom when combining connection entries.

(3) The connection entry component degree of freedom code is translated into the bit pattern and appropriate values of the IC array are set as in the following example:

Given the connection entries

100111/3/108/217/0 Assume $N_S = 3$

110001/6/0/223/337

The IC array would become, for the first degree of freedom,

$$[IC_{ij}^1 \mid CC_i] = \begin{bmatrix} 108 & 217 & 0 & \mid & 3 \\ 0 & 223 & 337 & \mid & 6 \end{bmatrix}$$

and, for the second degree of freedom,

$$[IC_{ij}^2 \mid CC_i] = [108 \ 217 \ 0 \ \mid \ 3], \text{ etc.}$$

FUNCTIONAL MODULE COMB1 (SUBSTRUCTURE COMBINATION, STEP 1)

(4) When the leading I_p number of a given connection entry is already in the matrix, the rows will be merged forming a new multiple connection, e.g.,

Given $100111/7/110/227/333/0$ with $N_S = 4$
 $110001/14/0/227/333/418$

$$\left[IC_{ij}^1 \mid CC_i \right] = \left[\begin{array}{cccc|c} 110 & 227 & 333 & 0 & 7 \\ 0 & 227 & 333 & 418 & 14 \end{array} \right]$$

Since 227, leading I_p of the second connection entry appears already, the two rows are immediately merged yielding,

$$\left[IC_{ij}^1 \mid CC_i \right] = [110 \ 227 \ 333 \ 418 \mid 15]$$

The connection code for the merged row is defined as the logical bitwise "OR" of the connection codes of the rows being merged. This process is carried out for each IC_{ij}^k .

Note that if the rows being merged contain different I_p numbers from the same pseudostructure, then an error condition exists, i.e., a connection is being attempted within one pseudostructure.

(5) Having combined all appropriate entries for each individual degree of freedom in the above step, the connection entry arrays IC are written onto scratch file SCR2 for each degree of freedom k in the following order:

$$k/ IC_{i1}^k / \dots / IC_{iN_S}^k / CC_i$$

(6) The reprocessing of combined connection entries proceeds as follows:

(a) The list of entries are read from SCR2 file into open core and sorted on the connection code, CC_i .

(b) If two entries n and m have the same connection code,

$$CC_n \equiv CC_m$$

test for

$$IC_{nj}^{k_1} \equiv IC_{mj}^{k_2}, \quad j = 1, \dots, N_S$$

If this test is passed, then the entries are merged and the degrees of freedom

MODULE FUNCTIONAL DESCRIPTIONS

are combined replacing the identical sets with the one new set such that:

$$(k_1 \quad k_2) / IC_{n_1} / \dots / IC_{n_{N_S}}$$

(c) When all individual entries have been exhausted for the current connection code, the new combined entries are written onto the SCCØNN file. This process is then repeated for each new connection code until all multiple connectivity data has been processed. These entries are written onto the SCCØNN file following those entries which were found not to be multiply-defined connectivities.

4.128.8.17 Subroutine Name: CMDISC

1. Entry Point: CMDISC

2. Purpose: Given the set of connection entries defining the degrees of freedom by which the pseudostructures are connected, the routine CMDISC generates a description of all those degrees of freedom not connected and merges these two lists to define the I_p numbers of the resultant structure.

3. Calling Sequence: CALL CMDISC

CØMMØN/CMBØØ1/

CØMMØN/CMBØØ2/

See Sec. 4.128.9 for description of labeled common blocks.

CØMMØN/CMBØØ3/

CØMMØN/CMBØØ4/

4. Method:

a. For each component substructure identified in the CØMBINE instruction bring into core the last record from SCSFIL containing the (SIL,C) pairs ordered by I_p number for each component. The SIL values are zeroed and the locations will be used to accumulate the degrees of freedom which have been connected ($C_{CØNNECT}$). Open core contains:

	$C_{CØNNECT}$	$C_{ØLD}$
SUB A	$\begin{bmatrix} (I_{PA_1}) \\ \vdots \\ (I_{PA_N}) \end{bmatrix}$	$\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$
		$\begin{bmatrix} 101010 \\ \vdots \\ 111000 \end{bmatrix}$

FUNCTIONAL MODULE CØMB1 (SUBSTRUCTURE COMBINATION, STEP 1)

$$\begin{array}{ccc} \text{SUB B} & \left[\begin{array}{cc} 0 & \vdots \\ \vdots & \vdots \\ 0 & \vdots \end{array} \right. & \\ & \vdots & \end{array}$$

b. The connection entries are read from scratch file SCCØNN and for each I_p encountered its $C_{CØNNECT}$ is updated by:

$$C_{CØNNECT} = C_{CØNNECT} \vee C_{CØNNECTION \text{ ENTRY}}$$

When the list of connection entries has been exhausted in this manner all totally or partially disconnected points can be isolated. (A logical "or" is denoted by \vee .)

c. Complete the definition of connection properties for all degrees of freedom in the combined structure by isolating the unconnected degrees of freedom. The table generated in steps 1 and 2 is processed as follows:

If $C_{CØNNECT} \equiv 0$, then the given I_p and its degrees of freedom ($C_{ØLD}$) are totally disconnected and a "DISCØNNECTION" entry is generated:

$$C_{DIS}/CC'/I_p$$

where $C_{DIS} = C_{ØLD}$ and CC' , corresponding to the individual pseudostructure considered, is defined by:

$$CC' = CC_{SUBSTRUCTURE}^{**2}$$

If $C_{ØLD} \neq C_{CØNNECT}$ and $C_{CØNNECT} \neq 0$, then the point is partially disconnected and set as:

$$C_{DIS} = C_{ØLD} - C_{CØNNECT}$$

(Note: if $C_{DIS} \vee C_{CØNNECT} \neq C_{ØLD}$ then $C_{CØNNECT} \neq C_{ØLD}$ and an error has occurred.)

Then $C_{DIS}/CC'/I_p$ is generated.

If $C_{ØLD} \equiv C_{CØNNECT}$ then all degrees of freedom for the given point have been accounted for and no new connect/disconnect entries are defined.

The disconnection entries are written onto scratch file SCDISC as they are being generated.

MODULE FUNCTIONAL DESCRIPTIONS

d. All connection and disconnection entries are read into core from files SCCØNN and SCDISC and sorted on the connection code. This combined set of entries defines the status of all degrees of freedom in the combined structure. The entire set is then written on the connection scratch file SCCØNN.

Note: The order sequence defined by CC codes arranges the assembled structure degrees of freedom as follows:

<u>CC</u>	<u>STRUCTURAL CONNECTION</u>
2	AA
3	AB
4	BB
5	AC
6	BC
7	ABC
8	CC
9	AD
10	BD
11	ABD
12	CD
13	ACD
14	BCD
15	ABCD
16	DD
⋮	⋮
⋮	etc.
⋮	⋮

Note that this ordering approaches optimal banding of the resulting stiffness matrix.

4.128.8.18 Subroutine Name: CMSØFØ

1. Entry Point: CMSØFØ
2. Purpose: To generate the SØF items for the newly created combined pseudostructures.
3. Calling Sequence: CALL CMSØFØ

CØMMØN/CMBØØ1/

CØMMØN/CMBØØ2/

CØMMØN/CMBØØ3/

CØMMØN/CMBØØ4/

See Sec. 4.128.9 for description of labeled common blocks.

FUNCTIONAL MODULE CØMB1 (SUBSTRUCTURE COMBINATION, STEP 1)

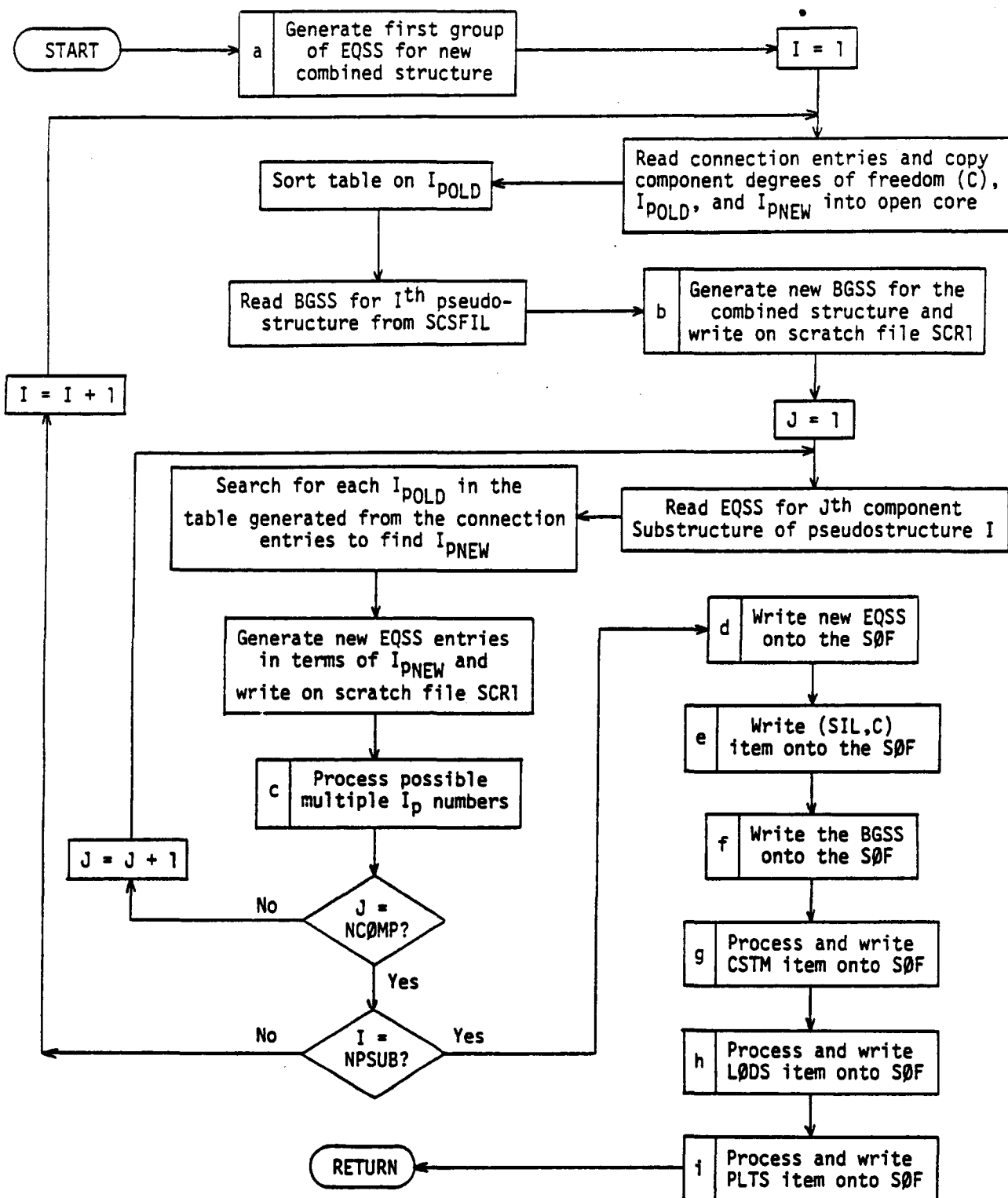
4. Method: The logical flow of subroutine CMSØFØ is given in Figure 6. Steps that are lettered in the flowchart are described below in detail.

- a. A call is made to subroutine SETLVL to enter the new pseudostructure into the SØF structure and the file SCTØC is brought in. With the information now available, the first group of the EQSS is written onto the SØF.
- b. A new entry of the BGSS, i.e., (CID,X,Y,Z), along with the new I_p number in the combined structure is written on scratch file SCR1 in one logical record.
- c. A search is made of the table generated from the connection entries to determine whether a given grid ID number is represented by several I_p numbers. If this is the case, additional entries to the EQSS are generated and written on SCR1. Each component substructure's EQSS is a single logical record.
- d. Each EQSS resident on file SCR1 is read back into open core and then written onto the SØF.
- e. The complete set of connection entries on file SCCØNN is read in and the (SIL,C) item is computed for the new pseudostructure and written onto the SØF.
- f. The modified BGSS for the new pseudostructure is read from scratch file SCR1. The entries are then sorted on I_{pNEW} and the BGSS is written onto the SØF in its proper form.
- g. The file SCCSTM generated in subroutine CMSFIL (Sec. 4.128.8.9) is read in and the set of local coordinate system transformations for the new pseudostructure is written onto the SØF.
- h. The LØDS items from each of the component substructures are merged and written onto the SØF.
- i. The geometric transformation matrices for each structure are merged and the PLTS item of the SØF is written for the new pseudostructure.

4.128.8.19 Subroutine Name: CMHGEN

1. Entry Point: CMHGEN
2. Purpose: To generate the [H] transformation matrix relating the degrees of freedom of each component pseudostructure to those of the final combined structure and to write these matrices on the SØF.

MODULE FUNCTIONAL DESCRIPTIONS



I - counts the pseudostructures (NPSUB)
 J - counts the component substructure of pseudostructure I (NCØMP)

Fig. 6. Logical flow of Subroutine CMSØFØ.

FUNCTIONAL MODULE CØMB1 (SUBSTRUCTURE COMBINATION, STEP 1)

3. Calling Sequence: CALL CMHGEN

CØMMØN/CMBØØ1/

CØMMØN/CMBØØ2/

CØMMØN/CMBØØ3/

CØMMØN/CMBØØ4/

See Sec. 4.128.9 for description of labeled common blocks.

4. Method:

- a. Read the master (SIL,C) group for the new combined structure into open core from the SØF (this was generated by routine CMDISC).

The following steps are repeated for each component substructure being combined.

- b. Read the (SIL,C) list from the substructure file SCSFIL into open core following the data from step 'a' above.
- c. The [H] matrix for the I_p under consideration is located on file SCCSTM using the pointers found on file SCSFIL.
- d. The connection entries on SCCØNN are then searched. For each I_p in the combined substructure, the I_{pOLD} from the component substructure and its component degree of freedom code are accessed. The [T'] matrix is processed as follows:

For each component j of C_{IPOLD} that equals zero, delete the jth row of [T'].

For each component k of $C_{CONNECTION ENTRY}$ that equals zero, delete the kth column of [T'].

This results in a reduced matrix with

N_A - The number of active DØF rows in A, and

N_C - The number of connected DØF columns on the connection entry, i.e.,

$$[T'']_{N_A \times N_C}$$

- e. The transformation matrix [H] is then built using the algorithm

$$H_{k,l} = T'_{i+l, j+l}$$

where

$$k = SIL_{IPOLD} + i, \quad l = SIL_{CONNECT} + j$$

and

$$i = 0, \dots, N_A - 1, \quad j = 0, \dots, N_C - 1$$

MODULE FUNCTIONAL DESCRIPTIONS

The matrix is written onto temporary scratch file SCR1 in packed format and, after the assembly is completed, onto the SØF via MTRXØ.

4.128.8.20 Subroutine Name: EQSCØD

1. Entry Point: EQSCØD

2. Purpose: During the combination of two or more substructures, of which at least one is a pseudostructure, the EQSS for the pseudostructure may contain a particular grid point identification number more than once. This routine adds a code, described below, to the component degree of freedom entry in the EQSS to facilitate the rapid location of all I_p numbers for a given grid ID.

3. Calling Sequence: CALL EQSCØD (LØC,N,Z)

LØC - The starting address in open core of the EQSS being searched - integer - input.

N - The number of words in the EQSS - integer - input.

Z - Open core array containing the EQSS group of the SØF

4. Method: With the EQSS residing in open core, the first grid ID is accessed. The remaining portion of the EQSS is checked sequentially for the appearance of the same grid ID. (Note: the EQSS is sorted on grid point ID.) If the next point is not the same, it becomes the object of a continued search. When a set of I_p numbers has been found for a given grid ID, a code is added to the component degree of freedom entry in the following manner:

Component Code ---

BITS 0 - 25	Degrees of freedom
BITS 26 - 28	Number of I_p 's representing given grid
BITS 29 - 31	The sequence number of the particular I_p in relation to the total number

Illustrative example: Suppose that grid ID 27 with component code 111111 is represented by four I_p numbers, 102,103,104;105, as follows:

Grid	I_p	Components
27	102	110000
27	103	000100
27	104	001001
27	105	000010

After processing by this routine the component codes become;

FUNCTIONAL MODULE CØMB1 (SUBSTRUCTURE COMBINATION, STEP 1)

<u>BIT POSITION</u>		
31 - 29	28 - 26	5 - 0
001	100	110000
010	100	000100
011	100	001001
100	100	000010

4.128.8.21 Subroutine Name: FINDER

1. Entry Point: FINDER
2. Purpose: To read the table of contents file for the combine operation (SCTØC), and for any basic substructure name returns the ID number of the pseudostructure containing it, and its position in the list of component substructures of the given pseudostructure.
3. Calling Sequence: CALL FINDER (NAM,SUBNØ,CØMNØ)

CØMMØN/CMBØØ1/

CØMMØN/CMBØØ2/ See Sec. 4.128.9 for description of labeled common blocks.

CØMMØN/CMBØØ3/

NAM - Two word name of basic substructure, BCD - input.

SUBNØ - The pseudostructure ID number, this corresponds to the order in which the pseudostructure names were entered in the substructure command CØMBINE - integer - output.

CØMNØ - The position of the given substructure name within the list of component substructures of the pseudostructure referred to by SUBNØ - integer - output.

4.128.8.22 Subroutine Name: GRIDIP

1. Entry Point: GRIDIP
2. Purpose: This subroutine finds a set of I_p numbers and their associated degree of freedom codes for any given grid point identification number contained in any of the pseudostructures being combined.
3. Calling Sequence: CALL GRIDIP (GRID,SEQSS,LEN,IPSET,CSET,NØ,Z,LLØC)

GRID - Grid point identification number relative to a basic substructure - integer - input.

SEQSS - The starting address of the EQSS in open core - integer - input.

MODULE FUNCTIONAL DESCRIPTIONS

- LEN - Number of words in the EQSS - integer - input.
- IPSET - The set of I_p numbers representing GRID - integer - output.
- CSET - Component degrees of freedom for the members of IPSET - integer - output.
- NØ - The number of I_p 's representing GRID - integer - output.
- Z - Open core array containing EQSS group of the SØF.
- LLØC - The address of the first appearance of GRID.

4. Method: A binary search of the EQSS is made to locate the grid point ID. The component degree of freedom code is checked for the additional code that may have been added to account for multiple I_p numbers (See Sec. 4.128.8.20). The complete set of I_p numbers and component degrees of freedom are collected and returned.

4.128.8.23 Subroutine Name: FNDGRD

1. Entry Point: FNDGRD
2. Purpose: Given a grid point identification number, the pseudostructure number, and the component substructure number, this routine locates the set of I_p numbers and their associated degree of freedom codes that represent the given grid point.
3. Calling Sequence: CALL FNDGRD (ISUB, ICØMP, IGRID, IP, C, N)

CØMMØN/CMBØØ1/

See Sec. 4.128.9 for the definition of labeled common blocks.

CØMMØN/CMBØØ2/

- ISUB - The pseudostructure number containing the grid point - integer - input.
- ICØMP - The component substructure number within ISUB - integer - input.
- IGRID - The grid point identification number - integer - input.
- IP - The set of I_p numbers that represent IGRID in the pseudostructure - integer - output.
- C - The set of component degrees of freedom at each I_p - integer - output.
- N - The number of I_p 's representing IGRID - integer - output.

4. Method: Using the values of ISUB and ICØMP as pointers into the file SCSFIL, the EQSS of the appropriate component substructure is accessed and subroutine GRIDIP (See Sec. 4.128.8.22) is called to find the I_p and C sets for IGRID.

4.128.8.24 Subroutine Name: ENCØDE

1. Entry Point: ENCØDE

FUNCTIONAL MODULE CØMB1 (SUBSTRUCTURE COMBINATON, STEP 1)

2. Purpose: To convert the degree of freedom codes as given in the usual bulk data form (i.e., a string of the integers 1 - 6 with, at most, one of each digit) to the bit pattern used in substructure analysis.

3. Calling Sequence: CALL ENCØDE (II)

II - The string to be encoded - integer - input; the encoded result - integer - output.

4. Method: The input string is divided by successive powers of ten to isolate the individual digits. The encoded result is then obtained by summing the appropriate powers of two that represent each degree of freedom present.

4.128.8.25 Subroutine Name: BITPAT

1. Entry Point: BITPAT

2. Purpose: To transform the degree of freedom code word into the binary bit pattern used by substructure analysis.

3. Calling Sequence: CALL BITPAT (ICØDE,IBITS)

ICØDE - The degree of freedom code - integer - input.

IBITS - An array of six words representing ICØDE as a binary string - integer - output.

4. Method: The value ICØDE is decoded and if the i^{th} bit ($i = 1, \dots, 6$) is on, IBITS _{i} is set to one.

4.128.8.26 Subroutine Name: EQSØUT

1. Entry Point: EQSØUT

2. Purpose: To generate and printout the connectivity map of a CØMBINE operation defining all structural connectivities that have been performed.

3. Calling Sequence: CALL EQSØUT

CØMMØN/CMBØØ3/ See Sec. 4.128.9 for description of labeled common blocks.

4. Method: The EQSS item of the SØF is accessed for the newly created pseudostructures. The names of the component substructures are copied into the first NWRD of open core and then the EQSS groups are copied. An additional word is added to the values contained in the EQSS. This packed word contains two codes: the number PSN of the contributing pseudostructure associated

MODULE FUNCTIONAL DESCRIPTIONS

with the particular grid ID and the component basic substructure number CN within the newly created pseudostructure. The table is stored in core in the configuration:

(PSN/CN)	G	I _p	C
⋮	⋮	⋮	⋮

This list is sorted on the internal point number (I_p). Next, the complete set of entries for any given I_p number (there may be many of the same I_p in a given pseudostructure) is isolated and passed to subroutine EQØUT1 (See Sec. 4.128.8.27) to perform the actual printing.

4.128.8.27 Subroutine Name: EQØUT1

1. Entry Point: EQØUT1

2. Purpose: This is a utility routine to do the actual reporting of the connectivity table with information passed by subroutine EQSØUT (See Sec. 4.128.8.26).

3. Calling Sequence: CALL EQØUT1 (IA,LEN1,NS,LEN2)

IA - Array containing the set of values passed by EQSØUT containing (PSN/CN G I_p C) for all points with identical I_p numbers - integer - input.

LEN1 - The length of IA - integer - input.

NS - Array containing the names of all component substructures of the given pseudostructure - integer - input.

LEN2 - The length of NS - integer - input.

4. Method: The portion of the EQSS passed to this routine is sorted on the first word; i.e., it is sorted on contributing pseudostructure number and component basic substructure number within the newly created pseudostructure. The connectivities are then extracted and printed out.

4.128.8.28 Subroutine Name: GTMAT1

1. Entry Points: GTMAT1

GTMAT2

GTMAT3

FUNCTIONAL MODULE C0MB1 (SUBSTRUCTURE COMBINATION, STEP 1)

2. Purpose: To find transformation matrices to be applied to grid points including TRANS geometric transformations, CSTM local coordinate system transformations, and GTRAN pseudo-structure local coordinate systems.

3. Calling Sequence:

CALL GTMAT1 (L0C1,LEN1,TRN,SYM,TT,T)

L0C1 - Address in open core where the TRANS transformation matrices reside, integer - input.

LEN1 - Length of TRANS data, integer - input.

TRN - TRANS set identification number, integer - input.

SYM - The code for symmetric reflections, integer - input.

TT - The transformation matrix, dimension 3 x 3, for TRN, real - output.

T - The transformation matrix, dimension 6 x 6, for TRN and SYM, real - output.

CALL GTMAT2 (L0C2,LEN2,ECPT,TC,TC6)

L0C2 - Address in open core where the CSTM coordinate system transformation matrices reside, integer - input.

LEN2 - Length of CSTM data, integer - input.

ECPT - An array, dimension 4, where,

ECPT(1) = CID (CSTM identification number)

ECPT(2) - ECPT(4) are the coordinates of the point being transformed
mixed - input.

TC - The transformation matrix, dimension 3 x 3, for CID, real - output.

TC6 - The transformation matrix, dimension 6 x 6, for CID, real - output.

CALL GTMAT3 (TRAN,TG,TG6,IKIND)

TRAN - The TRANS set ID pointed to by the GTRAN data, integer - input.

TG - The transformation matrix, dimension 3 x 3, for GTRAN, real - output.

TG6 - The transformation matrix, dimension 6 x 6, for GTRAN, real - output.

IKIND - Code word describing the matrices that have been located for a given point,
integer - output.

MODULE FUNCTIONAL DESCRIPTIONS

COMMON/CMBZZZ/Z(1) - Open core common block containing transformation matrices.

4. Method: GTMAT1 searches through the TRANS coordinate transformation matrices, [TT], locating the appropriate ID with subroutine PRETRS. The matrix for symmetric reflection, [SYM], is then constructed from the input SYM code and the final transformation is computed by:

$$[T]_{6 \times 6} = \begin{bmatrix} [TT] & \\ & [TT] \end{bmatrix} [SYM]_{6 \times 6}$$

IKIND is initialized to zero.

GTMAT2 searches for any CSTM for a given point by using PRETRS routines. If a CSTM does exist, IKIND is updated by:

$$IKIND = IKIND \vee 1$$

where "V" indicates the union operator.

GTMAT3 processes GTRAN transformations in the following manner:

If $TRAN \neq 0$, a call is made to PRETRS to locate the appropriate TRANS transformation matrix, and

if $TRAN = TRN$, then $IKIND = IKIND \vee 10$
 if $TRAN \neq TRN$, then $IKIND = IKIND \vee 2$

If $TRAN = 0$, the transformation is the identity, and

$$IKIND = IKIND \vee 6$$

the code IKIND is then a five-bit code having the following interpretation:

Bit 0	0 - No TRANS or SYMT
	1 - TRANS or SYMT
Bit 1	0 - No CSTM
	1 - CSTM
Bit 2	0 - No GTRAN
	1 - GTRAN
Bit 3	0 - $TRAN = 0$
	1 - $TRAN \neq 0$
Bit 4	0 - $TRAN = TRN$
	1 - $TRAN \neq TRN$

4.128.8.29 Subroutine Name: CMCKDF

1. Entry Point: CMCKDF
2. Purpose: To ensure that the degrees of freedom at several grid points being connected are compatible.
3. Calling Sequence: CALL CMCKDF

CØMMØN/CMBØØ1/

CØMMØN/CMBØØ2/ See Sec. 4.128.9 for description of labeled common blocks.

CØMMØN/CMBØØ3/

4. Method: The complete set of coordinate system transformation matrices (CSTM) are read into open core. These are followed by the BGSS items for each of the component substructures read from file SCSFIL. The set of connection entries residing on SCCØNN are read, one at a time, and for each Ip number in the entry, the ID number of the CSTM for that point is found. A comparison is then made of all CSTM's relating to the points in the connection entry. If any of these transformations differs from the others an error has been made and the degrees of freedom at the connection are incompatible. User Fatal Message 6528 is then issued.

4.128.8.30 Subroutine Name: CMTRCE

1. Entry Point: CMTRCE
2. Purpose: To convert a set of internal point numbers to grid point ID's and component substructures for diagnostic purposes.
3. Calling Sequence: CALL CMTRCE (IERTAB,IWDS,ITØMNY)

IERTAB - Array containing IP numbers - input - integer.

IWDS - Length of IERTAB array - input - integer.

ITØMNY - Flag set to 1 if IERTAB does not contain all IP's found - input - integer.

4. Method: This routine reads each EQSS into open core and searches for the IP contained in IERTAB. It then translates the IP to the Grid Point ID in the component substructure and issues a user fatal message in tabular form.

MODULE FUNCTIONAL DESCRIPTIONS

4.128.9 Labeled Common Blocks

4.128.9.1 Common Block CMB001

CØMMØN/CMB001/SCR1,SCR2,SCR3,SCBDAT,SCSFIL,SCCØNN,SCMCØN,SCTØC,SCCSTM,GEØM4,CASECC

<u>Name</u>	<u>GINØ File No.</u>	<u>Type</u>	<u>Contents</u>
SCR1	301	I	Temporary scratch
SCR2	302	I	Temporary scratch
SCR3	303	I	Temporary scratch
SCBDAT	304	I	Contains processed input data
SCSFIL	305	I	Processed SØF for substructure being combined
SCCØNN	306	I	File containing connection entries
SCMCØN	307	I	Contains list of multiple connected points
SCTØC	308	I	Holds the substructure table of contents for CØMBINE operation
SCCSTM	309	I	Contains local coordinate system transformation matrices and [H] matrices
GEØM4	102	I	Bulk data NASTRAN input data block
CASECC	101	I	NASTRAN case control data block

4.128.9.2 Common Block CMB002

CØMMØN/CMB002/BUF1,BUF2,BUF3,BUF4,BUF5,SCØRE,LCØRE,INPUT,ØUTT

<u>Name</u>	<u>Type</u>	<u>Contents</u>
BUF1	I	Address of GINØ IØ buffers
BUF2	I	Address of GINØ IØ buffers
BUF3	I	Address of GINØ IØ buffers
BUF4	I	Address of GINØ IØ buffers
BUF5	I	Address of GINØ IØ buffers
SCØRE	I	Starting address of open core
LCØRE	I	Length of open core

FUNCTIONAL MODULE CØMB1 (SUBSTRUCTURE COMBINATION, STEP 1)

INTP I Card input Fortran unit number
 ØUTT I Printer output Fortran unit number

4.128.9.3 Common Block CMB003

CØMMØN/CMB003/CØMBØ(7,5),CØNSET,IAUTØ,TØLER,NPSUB,CØNECT,TRAN,MCØN,RESTCT(7,7)

<u>Name</u>	<u>Type</u>	<u>Contents</u>
CØMBØ	MIXED	Case control and names of component substructures
CØNSET	I	Manual connection set ID number
IAUTØ	L	.TRUE. if automatic connection option is specified
TØLER	R	Allowable geometric tolerance for connections
NPSUB	I	Number of substructures being combined
CØNECT	L	.TRUE. if a connection set has been specified
TRAN	L	.TRUE. if any TRANS coordinate transformations have been specified
MCØN	L	.TRUE. if multiple connections have been specified
RESTCT	I	Array containing selective search controls for automatic connect option

4.128.9.4 Common Block CMB004

CØMMØN/CMB004/TDAT(6),NIPNEW,CNAM(2)

<u>Name</u>	<u>Type</u>	<u>Contents</u>
TDAT(1)	L	.TRUE. if CØNCT1 bulk data has been input
TDAT(2)	L	.TRUE. if CØNCT bulk data has been input
TDAT(3)	L	.TRUE. if TRANS bulk data has been input
TDAT(4)	L	.TRUE. if RELES bulk data has been input
TDAT(5)	L	.TRUE. if GNEW bulk data has been input
TDAT(6)	L	.TRUE. if GTRAN bulk data has been input
NIPNEW	I	Number of I_p in new combined structure
CNAM(2)	BCD	Name to be given to newly created structure

MODULE FUNCTIONAL DESCRIPTIONS

4.128.10 Design Requirements

Open core resides in common block CMBZZZ.

4.128.11 Diagnostics

Diagnostic messages 6501-6523, 6528, and 6530-6532 are issued by CØMB1 and its subroutines.

FUNCTIONAL MODULE C0MB2 (SUBSTRUCTURE COMBINATION, STEP 2)

4.129 FUNCTIONAL MODULE C0MB2 (SUBSTRUCTURE COMBINATION, STEP 2)

4.129.1 Entry Point: C0MB2

4.129.2 Purpose

This module performs the transformation and addition of the matrices for the substructure combine operation. Either stiffness, mass, or load matrices may be operated on.

4.129.3 DMAP Calling Sequence

C0MB2 MAT1,MAT2,MAT3,MAT4,MAT5,MAT6,MAT7/MATC/V,N,DRY/C,N,TYPE/C,N,NA1/C,N,NA2/C,N,NA4/
C,N,NA5/C,N,NA6/C,N,NA7 \$

4.129.4 Input Data

4.129.4.1 GIN0 Data Blocks

MAT1 - MAT7 Substructure matrices for stiffness, mass, or loads.

Note: Any input matrix may be purged. See note for parameters NA1 - NA7.

4.129.4.2 S0F Items

H0RG - Transformation matrices

KMTX - Stiffness matrices (g-set)

MMTX - Mass matrices (g-set)

PVEC - Load matrices (g-set)

Note: H0RG must exist for each substructure being combined.

4.129.5 Output Data

4.129.5.1 GIN0 Data Blocks

MATC Matrix for combined substructure

Note: MATC may not be purged.

4.129.5.2 S0F Items

None

4.129.6 Parameters

DRY - Input and output, integer. On input, if DRY < 0, no operations are performed. On output, if a serious error occurred, DRY = -2.

MODULE FUNCTIONAL DESCRIPTIONS

TYPE - Input, BCD. TYPE = K; process stiffness matrices; M, mass; P, loads.

NA1 - NA7 - Input, BCD. Names of substructures whose matrices are on MAT1 - MAT7. Must be blanks for matrices which are not being combined. If NAI is non-blank and MATi is purged, MATi must be on the SØF.

4.129.7 Method

CØMB2 begins by fetching the matrix control blocks (or trailers) for each substructure indicated by a non-blank NAI parameter. If the GINØ file corresponding to one of these parameters is purged, CØMB2 fetches the MCB from the SØF. Next, the MCB's of the HØRG transformation matrices are read from the SØF. Fatal errors may occur if: (1) an indicated matrix cannot be found on its GINØ input data block or on the SØF, (2) an HØRG item cannot be found, (3) matrix dimensions are not compatible. After all MCB's have been checked, CØMB2 branches on matrix type.

For mass and stiffness matrices, the following method of combination is used:

$$[K_C] = \sum [H_i]^T [K_i] [H_i]$$

or $[M_C] = \sum [H_i]^T [M_i] [H_i]$

where K_i and M_i are the structural matrices for each of the substructures and $[K_C]$ and $[M_C]$ are the matrices of the combination (MATC).

For each matrix contributing to the final combination matrix, the following sequence of operations is performed:

1. Move the HØRG item from the SØF to SCR2.
2. If the input matrix is on the SØF, move it to SCR1.
3. Post-multiply the matrix by HØRG. Store the result on SCR3.
4. Pre-multiply the result of step 3 by the transpose of HØRG, and add to the result of the previous step 4. Store the result on either MATC or SCR4. MATC and SCR4 are used alternatively so that after the last multiply and add, the result is on MATC.

For load matrices, the following method of combination is used:

$$[P_C] = \left[\begin{array}{c|c|c|c} H_1^T P_1 & H_2^T P_2 & H_3^T P_3 & \text{-----} \end{array} \right]$$

where P_1, P_2 , etc., are the load vector matrices for the contributing substructures. The load vectors are not combined but are created as separate columns in the matrix preparatory for the combination at solution time.

FUNCTIONAL MODULE CØMB2 (SUBSTRUCTURE COMBINATION, STEP 2)

For each matrix contributing to the final combination load matrix the following steps are performed:

1. Move the HØRG item from the SØF to SCR2.
2. If the input matrix is on the SØF, move it to SCR1.
3. Pre-multiply the matrix by the transpose of HØRG and store the result on MATC, SCR3, or SCR4.
4. Construct a partitioning vector to merge the result of step 3 with the result of the previous step 5.
5. Merge result of step 3 with the result of the previous step 5 and store on MATC, SCR3, or SCR4.

The files MATC, SCR3, and SCR4 used in steps 3 and 5 are alternated so that the final result will be on MATC when all matrices have been merged together.

4.129.8 Subroutines

CØMB2 uses matrix subroutines MPYAD and MERGE.

4.129.9 Diagnostic Messages

CØMB2 may issue "fatal" messages 6301, 6302, 6303, 6304, and 3008. However, it will not terminate execution. Instead, it will set the DRY parameter equal to -2 and return.

FUNCTIONAL MODULE EXIØ (EXTERNAL INPUT/OUTPUT FOR THE SØF)

4.130 FUNCTIONAL MODULE EXIØ (EXTERNAL INPUT/ØUTPUT FOR THE SØF)

4.130.1 Entry Point: EXIØ

4.130.2 Purpose

EXIØ copies selected items between the SØF and an external user file.

4.130.3 DMAP Calling Sequence

EXIØ //V,N,DRY/C,N,MACH/C,N,DEVICE/C,N,UNIT/C,N,FØRM/C,N,MØDE/C,N,PØSN/C,N,TYPE/C,N,NA1/
C,N,NA2/C,N,NA3/C,N,NA4/C,N,NA5 \$

4.130.4 Input Data

4.130.4.1 GINØ Data Blocks

None

4.130.4.2 SØF Items

Any SØF item may be indicated by the DMAP parameters.

4.130.5 Output Data

4.130.5.1 GINØ Data Blocks

None

4.130.5.2 SØF Items

Any SØF item may be indicated by the DMAP parameters.

4.130.6 Parameters

DRY - Input and output, integer, no default. If a serious error occurs, DRY is assigned the value of -2. Otherwise, it is not changed.

MACH - Input, integer, no default. For external coded tapes (parameter FØRM = 'EXTERNAL'), MACH specifies the computer (360, 1108, or 6600) where the tape was created or where the tape is to be read.

DEVICE - Input, BCD, no default. The device type on which the external file is located. Allowable values are 'TAPE' or 'DISK'.

UNIT - Input, BCD, no default. Name of the unit where the external file is located. For internal uncoded files (parameter FØRM = 'INTERNAL'), allowable values are 'INPT', 'INP1', 'INP2', ..., 'INP9'. For external coded tapes (FØRM = 'EXTERNAL'), allowable

MODULE FUNCTIONAL DESCRIPTIONS

values are 'FØRT1', 'FØRT2', ..., 'FØRT32', which refer to FØRTRAN unit numbers. Any FØRTRAN unit which does not conflict with the NASTRAN units may be used.

FØRM - Input, BCD, no default. 'INTERNAL' for an internal file which will be read and written with GINØ. 'EXTERNAL' for an external tape which will be read or written with FØRTRAN formatted I/Ø.

MØDE - Input, BCD, no default. MØDE specifies the operation to be performed by the EXIØ module:

'SØFIN' - Copies items from the external file to the SØF.

'SØFØUT' - Copies items from the SØF to the external file.

PØSN - Input, BCD, no default. File positioning parameter for the external file. Allowable values are 'REWIND', 'NOREWIND', and 'EØF'. 'EØF' positions the file to the point immediately preceding the first end-of-file indicator on the external file.

TYPE - Input, BCD, default = 'ALL'. TYPE specifies which items of the selected substructures are to be copied. The allowable options are:

1. 'ALL' - All items
2. 'TABLES' - EQSS, BGSS, CSTM, LØDS, PLTS, SØLN
3. 'MATRICES' - KMTX, MMTX, PVEC, PØVE, UPRT, HØRG, UVEC, QVEC
4. 'PHASE3' - SØLN, UVEC, QVEC
5. Actual item name

NA1 - Input, BCD, default = 'WHØLESØF'. The name of a substructure whose items are to be copied. If NA1 = 'WHØLESØF', all substructures on the SØF are copied out. Or, all substructures from the current position of the external file (specified by PØSN) to the end-of-file are copied in.

NA2 }
NA3 } Input, BCD, default = 'XXXXXXXX'. Names of substructures whose items are to be copied.
NA4 }
NA5 }

4.130.7 Method

The EXIØ module consists of two major sections. Subroutine EXIØ1 performs all operations for external files which are read and written with GINØ. Subroutine EXIØ2 is the driver subroutine for all operations with external tapes which are read and written with FØRTRAN.

The methods of EXIØ1 and EXIØ2 are described in the next section.

FUNCTIONAL MODULE EXIØ (EXTERNAL INPUT/OUTPUT FOR THE SØF)

4.130.8 Subroutines

4.130.8.1 Subroutine Name: EXIØ

1. Entry Point: EXIØ
2. Purpose: EXIØ module driver
3. Calling Sequence: CALL EXIØ
4. Method: EXIØ tests the FØRM parameter and calls either EXIØ1 or EXIØ2.

4.130.8.2 Subroutine Name: EXIØ1

1. Entry Point: EXIØ1
2. Purpose: To copy substructure items between the SØF and an external file using the GINØ subroutines.
3. Calling Sequence: CALL EXIØ1
4. Method: EXIØ1 tests the MØDE parameter and branches to the sections of code used to process the requested operation.

SØFØUT. The external file is opened and a nine-word ID record is written. This record contains an ID flag, the SØF password, the current date, and the time of day.

The TYPE parameter is used to construct an array of names of items which are to be copied out for each substructure.

For each substructure and each item which is copied out, a 10-word header record is written. This record contains a header flag, the substructure name, the item name, the date, and the time of day. Each group of the item is written as one GINØ logical record. When the entire item has been copied, a one-word end-of-item record is written.

SØFIN. The TYPE parameter is used to construct an array of names of items which are to be copied in for each substructure. The total number of items to be copied is computed and stored in variable NCØPY. The external file is opened at the requested position and the first record is read. If the first record is not an ID or header record as described for SØFØUT, error number 6343 occurs. The first record is stored in array HDREC. All ID or header records read subsequently are compared to HDREC to insure that the external file is scanned only once. The external file is scanned from the current position until all requested items have been copied to the SØF or until the entire file has been scanned. Note that if an

MODULE FUNCTIONAL DESCRIPTIONS

end-of-file is encountered before all copy requests have been satisfied, the external file is rewound.

For each header record found, the following processing takes place:

- a. The header record is compared to the first record read. If identical, a list of requested items which were not found is printed and EXIØ1 returns.
- b. The substructure and item names found in the header record are compared to those which are to be copied. If this item is not desired, the next header record is found and read and control passes to step 'a'.
- c. The header record is compared to the header records of all items which have been copied so far. (Header records are stored in open core.) If a duplicate is found, the most recent version is used. Otherwise, this header record is added to those saved in open core.
- d. The data is copied from the external file to the SØF. Each logical record is written as one group until the end-of-item record is encountered.
- e. The number of items copied is incremented and compared to NCØPY. If equal, EXIØ1 returns. Otherwise the next header record is read and control passes to step 'a'.

4.130.8.3 Subroutine Name: EXIØ2

1. Entry Point: EXIØ2
2. Purpose: To copy substructure items between the SØF and an external file using FØRTRAN formatted IØ. The external file may be read or may have been written on a different type of computer.
3. Calling Sequence: CALL EXIØ2

COMMON/EXIØ2F/FMT(1)

COMMON/EXIØ2P/NF,FP(5,1)

FMT - Array of variable formats initialized in block data subroutine EXIØBD.

NF - Number of formats in FMT.

FP - Array of pointers to the variable formats in FMT, 5 words per format:

FUNCTIONAL MODULE EXIØ (EXTERNAL INPUT/OUTPUT FOR SØF)

FP(1,i) = pointer to first word of ith format

FP(2,9) = length of ith format in words

FP(3,i) = position to blocking factor

FP(4,i) = 1108 blocking factor

FP(5,i) = 360/6600 blocking factor - The blocking factor adjusts the length of each format in FMT.

4. Method: EXIØ2 tests the MACH parameter and the machine type word in /SYSTEM/ and initializes the blocking factors in FMT. If MACH equals 1108 or the 22nd word of /SYSTEM/ equals 3, the blocking factor is chosen so that each FØRTRAN record is 132 characters long. Otherwise, the blocking factor adjusts the length of each FØRTRAN record to 1280 characters.

For SØFØUT operations, subroutine EXØ2 is called to generate the external file. If MACH equals 1108 and the computer on which NASTRAN is running is not an 1108, subroutine EXØ2EB is called subsequently to convert the external file into a form which can be read on the 1108.

For SØFIN operations, subroutine EXI2 is then called to copy the data to the SØF.

4.130.8.4 Subroutine Name: EXØ2

1. Entry Point: EXØ2
2. Purpose: To copy items from the SØF to an external file with FØRTRAN formatted IØ.
3. Calling Sequence: CALL EXØ2
4. Method: EXØ2 constructs the array of names of items which are to be copied. For each item, EXØ2 branches to a section of code where the item is read from the SØF and the proper format specifications are chosen. Subroutine EXFØRT is called to perform the output.

The external file is organized as blocks of FØRTRAN logical records each written with the same format specifications. Each block is preceded by a header record which contains the following information:

- a. Substructure name
- b. Item name
- c. Format number

MODULE FUNCTIONAL DESCRIPTIONS

- d. Number of words of data in the following block
- e. Double precision matrix flag
 - 1 - single precision
 - 2 - double precision
- f. End-of-group flag
 - 0 - more data for this group follows
 - 1 - this block terminates group
 - 2 - this block terminates item

The first two blocks written contain the DIT and the first two words of the MDI for all substructures on the SØF.

4.130.8.5 Subroutine Name: EXI2

- 1. Entry Point: EXI2
- 2. Purpose: To copy items from the external file to the SØF using FØRTRAN formatted IØ.
- 3. Calling Sequence: CALL EXI2
- 4. Method: The DIT and MDI blocks are copied into open core. All items on the external file are copied to the SØF. Any user requests for particular substructures or items are ignored. The header record of each block is used to interpret the data which is contained in the block. The description of subroutine EXØ2 describes the header record and the file organization.

FUNCTIONAL MODULE EXIØ (EXTERNAL INPUT/OUTPUT FOR SØF)

4.130.8.6 Subroutine Name: EXFØRT

1. Entry Point: EXFØRT

2. Purpose: To perform FØRTRAN formatted IØ for module EXIØ.

3. Calling Sequence: CALL EXFØRT (IRW,UNIT,FMT,BUF,NWDS,DPMAT,DBUF)

IRW - Read/write flag. Integer - input.

1 - read

2 - write

UNIT - FØRTRAN unit number. Integer - input.

FMT - Array containing the format specification. BCD - input.

BUF - Array where data read will be stored or from which data will be written. Input and output.

NWDS - Number of words to be transmitted.

DPMAT - Double precision matrix flag. Integer - input.

1 - Data is not a double precision matrix

2 - Data is a double precision matrix

DBUF - The same array as BUF. Double precision, input and output.

4. Method: If a physical end-of-file is found in the read mode, BUF(3) is set to -1.

4.130.8.7 Subroutine Name: EXLVL

1. Entry Point: EXLVL

2. Purpose: To add a substructure name found on the external file to the SØF and set the MDI pointers for it.

3. Calling Sequence: CALL EXLVL (NØS,MD,NAME)

MODULE FUNCTIONAL DESCRIPTIONS

NØS - Number of substructures in array MD. Integer - input.

MD - Array of dimension MD(4,NØS) containing the substructure name and the first two words of the MDI from the originating SØF.

NAME - Name of substructure to be added to the resident SØF. Two word BCD array.

4. Method: EXLVL looks for a match between NAME and the first two words of each entry of the MD array. If no match is found, the substructure is added to the SØF as a basic substructure and EXLVL returns.

If a match is found, the substructure is added to the SØF and an attempt is made to find its old higher level substructure on the SØF. If present, the MDI pointer is set to indicate it. EXLVL also attempts to find and set pointers for all combined substructures and lower level substructures. Pointers for primary and secondary substructures are ignored.

4.130.9 Design Requirements

On CDC computers, integer function FØRFIL is used to obtain the FØRTRAN unit number of GINØ file 301 which is used for this purpose.

4.130.10 Diagnostic Messages

EXIØ may issue the following messages:

3001,3002,3003,3008

6101,6102,6103

6333 through 6359

FUNCTIONAL MODULE RCØVR (RECOVER PHASE 2 SUBSTRUCTURE RESULTS)

4.131 FUNCTIONAL MODULE RCØVR (RECOVER PHASE 2 SUBSTRUCTURE RESULTS)

4.131.1 Entry Point: RCØVR

4.131.2 Purpose

RCØVR recovers Phase 2 solution data for lower level and basic substructures. This module computes the SØLN item on the SØF in Phase 2 for use in Phase 3 processing. It calculates the displacement vectors for any of the substructures comprising the final solution structure (FSS). If an SPCFØRCE output request exists, it also calculates the reaction forces using those displacement vectors. All recovered data is saved on the SØF. In addition, if output requests are present, ØFP print data blocks are created.

4.131.3 DMAP Calling Sequence

RIGID FORMATS 1 AND 2 (STATIC ANALYSIS)

RCØVR CASESS,GEØM4,KGG,MGG,PG,UGV,,,,/ØUGV1,ØPG1,ØQG1,U1,U2,U3,U4,U5/DRY/ILØØP/STEP/FSS/
RFNØ/O/LUI/U1NM/U2NM/U3NM/U4NM/U5NM/S,N,NØSØRT2/V,Y,UTHRESH/V,Y,PTHRESH/V,Y,
QTHRESH \$

RIGID FORMAT 3 (MODAL ANALYSIS)

RCØVR CASESS,LAMA,KGG,MGG,,PHIG,,,,/ØPHIG,,ØQG1,U1,U2,U3,U4,U5/DRY/ILØØP/STEP/FSS/
RFNØ/NEIGV/LUI/U1NM/U2NM/U3NM/U4NM/U5NM/S,N,NØSØRT2/V,Y,UTHRESH/V,Y,PTHRESH/V,Y,
QTHRESH \$

RIGID FORMAT 8 (FREQUENCY ANALYSIS)

RCØVR CASESS,GEØM4,KGG,MGG,PPF,UPVC,DIT,DLT,BGG,K4GG,PPF/ØUGV1,ØPG1,ØQG1,U1,U2,U3,
U4,U5/DRY/ILØØP/STEP/FSS/RFNØ/O/LUI/U1NM/U2NM/U3NM/U4UN/U5NM/S,N,NØSØRT2/V,Y,
UTHRESH/V,Y,PTHRESH/V,Y,QTHRESH \$

RIGID FORMAT 9 (TRANSIENT ANALYSIS)

RCØVR CASESS,GEØM4,KGG,MGG,PPT,UPV,DIT,DLT,BGG,K4GG,TØL/ØUGV1,ØPG1,ØQG1,U1,U2,U3,U4,U5/
DRY/ILØØP/STEP/FSS/RFNØ/O/LUI/U1NM/U2NM/U3NM/U4UN/U5NM/S,N,NØSØRT2/V,Y,UTHRESH/
V,Y,PTHRESH/V,Y,QTHRESH \$

MRECOVER (ANY RIGID FORMAT)

RCØVR ,,,,,,,/ØPHIG,,ØQG1,U1,U2,U3,U4,U5/DRY/ILØØP/STEP/FSS/3/NEIGV/LUI/U1NM/U2NM/
U3NM/U4NM/U5NM/S,N,NØSØRT2/V,Y,UTHRESH/V,Y,PTHRESH/V,Y,QTHRESH \$

MODULE FUNCTIONAL DESCRIPTIONS

4.131.4 Input Data

4.131.4.1 GINØ Data Blocks

CASESS - Substructuring Case Control
GEØM4 - Substructuring load combination data
LAMA - Real eigenvalue table
KGG - Stiffness matrix (g-set) for FSS
MGG - Mass matrix (g-set) for FSS
PG - Static load matrix (g-set) for FSS
PPF - Frequency Response Dynamic Loads (g-set) for FSS
PPT - Transient Dynamic Loads (g-set) for FSS
UGV - Static displacement matrix (g-set) for FSS
PHIG - Eigenvector matrix (g-set) for FSS
UPVC - Frequency Response Solution Vector (g-set) for FSS
UPV - Transient Solution vector (g-set) for FSS
DIT - Direct input tables
DLT - Dynamic loads table
BGG - Viscous damping matrix (g-set) for FSS
K4GG - Structure damping matrix (g-set) for FSS
TØL - Transient output list

Notes:

1. CASESS may not be purged.
2. All remaining data blocks may be purged if the request PRINT or SAVE operation is not for FSS.
3. GEØM4 may be purged if there are no external static loads requested or a SØLN item already exists for FSS.
4. KGG, MGG, BGG and K4GG may be purged if they are not required to compute reaction forces and displacement vectors or if they have already been saved on the SØF.

FUNCTIONAL MODULE RCØVR (RECOVER PHASE 2 SUBSTRUCTURE RESULTS)

5. PG or PPT may be purged.
6. UGV, PHIG, UPVC or UPV must be present unless they already have been saved on the SØF.
7. PPF or TØL must be present for a transient or frequency response problem unless a SØLN item already exists for FSS.

4.131.4.2 SØF Items

EQSS - Substructure equivalence table
LØDS - Load set identification list
HØRG - Transformation matrices
KMTX - Stiffness matrix (g-set)
MMTX - Mass matrix (g-set)
PVEC - Load matrix (g-set)
PØVE - Load matrix for points omitted in reductions
UPRT - Partitioning vectors from reductions
UVEC - Displacement or eigenvector matrices (g-set)
QVEC - Reaction force matrix (g-set)
SØLN - Solution description data
LMTX - Decomposition product from reductions
PHIS - Modal reduction eigenvectors (g-set)
LAMS - Modal reduction eigenvalues
BMTX - Viscous damping matrix (g-set)
K4MX - Structural damping matrix (g-set)

4.131.5 Output Data

4.131.5.1 GINØ Data Blocks

ØUGV1 - Output displacement requests
ØPHIG - Output eigenvector requests
ØPG1 - Output load vector requests
ØQG1 - Output reaction force requests

MODULE FUNCTIONAL DESCRIPTIONS

U1 - U5 Substructure displacement matrices. Used in subsequent execution of this module for both input and output.

Notes:

1. If an output data block is purged, any output requests for it will be ignored.
2. Any or all U1 - U5 data blocks may be purged.

4.131.5.2 SØF Items

SØLN - Solution description data

UVEC - Displacement or eigenvector data (g-set)

QVEC - Reaction force matrix (g-set)

4.131.6 Parameters

- DRY - Integer - input. Dry run parameter.
 DRY < 0 Input checked and SØLN generated
 DRY = 0 Complete module execution except SØLN will not be generated
 DRY > 0 Complete module execution
- ILØØP - Integer - input and output. Position of the substructure name to be processed for the current RECOVER command, ILØØP = -1 when all substructures have been processed.
- STEP - Integer - input. CASESS record number of the current RECOVER command.
- FSS - BCD - input. Name of the final solution structure currently being processed.
- RFNØ - Integer - input. Rigid Format number. (1,2,3,8 or 9)
- NEIGV - Integer - input. Number of eigenvalues. (Default = -1)
- LUI - Integer - input and output. Sequence number (1 - 5) of the last U1 - U5 data block which has been written by a previous call to this module.
(Default = 0)
- U1NM - U5NM - BCD - input and output. BCD name of the substructure whose displacement or eigenvector matrix resides on data blocks U1 - U5. Should be blanks for any data block which is purged. (Default = blank)

FUNCTIONAL MODULE RCØVR (RECOVER PHASE 2 SUBSTRUCTURE RESULTS)

NØSORT - Integer - output. True if SØRT1 output is desired or false if SØRT2 output is desired.

UTHRESH - Real - input. Displacement printout threshold value. Any result whose absolute value is less than UTHRESH will be printed as a zero. (Default = 0.0)

PTHRESH - Real - input. Load vector printout threshold value. (Default = 0.0)

QTHRESH - Real - input. Reaction force printout threshold value. (Default = 0.0)

4.131.7 Method

The RCØVR module is itself modular in design. It consists of four major subroutines which are called in turn by a small module driver subroutine.

RCØVØ reads the CASESS record for the current RECOVER command and processes any user output requests.

RCØVA computes the SØLN item for the final solution structure (FSS) when the parameter DRY \neq 0. It also copies KGG, MGG, BGG, K4GG and UGV or PHIG or UPVC or UPV to the SØF if this has not already been done.

RCØVB performs the back-substitution to recover the displacements of lower level and basic substructures from those of the final solution structure. If the displacements of an intermediate level substructure have already been recovered, it may not be necessary to back-track all the way to the FSS in order to recover the requested displacements. Note that the displacements of only one substructure (indicated by the values of the ILØØP and STEP parameters) can be recovered in any one call to the RCØVR module.

RCØVC computes matrices of reaction forces and writes output data blocks for use by the ØFP. RCØVC is executed only if the RECOVER option is not 'SAVE'. User output requests are made in the Case Control Deck or the Substructure Control Deck. For displacement output, ØUGV1 or ØPHIG is generated directly from the displacement matrix. Loads output is generated directly only if the recovery is for the FSS and PG is not purged. Otherwise, it must be computed using the SØLN item of the FSS and the PVEC item of the requested substructure. Reaction force matrices (the QVEC item) are computed only if there is an SPCFØRCE output request. Note that ØQGI contains reactions at all g-set points, not just the points which are single point

MODULE FUNCTIONAL DESCRIPTIONS

constraints. Output set specifications in Case Control or Substructure Control are honored by RC0VC. However, if output is being generated for a non-basic substructure, only those external grid-point ID's which remain in a substructure after reduction and combination are printed. The printout set list is applied to basic substructures according to the BASIC command. Elements of a set which are not found on a basic substructure are ignored.

RC0VE computes the energies on the modal coordinates of a modal reduced substructure. It will also compute the energies on the modes excluded from the modal reduce. RC0VE is executed only if an energy request was made for a non-statics solution.

The detailed methods of these subroutines are described in the next section.

4.131.8 Subroutines

For purposes of communications between subroutines of the RC0VR module, the following common blocks are significant.

<u>Block</u>	<u>Variables in Order</u>
/RC0VER/	IC0RE - start of available open core LC0RE - length of available open core BUF1 - GIN0 buffer BUF2 - GIN0 buffer BUF3 - GIN0 buffer BUF4 - GIN0 buffer S0F1 - S0F buffer S0F2 - S0F buffer S0F3 - S0F buffer
/RC0VCM/	MRECVR - True if a MRECOVER or false if a RECOVER command UA - GIN0 file for displacements PA - GIN0 file for loads QA - GIN0 file for reactions IOPT - -1 if error occurred

FUNCTIONAL MODULE RCØVR (RECOVER PHASE 2 SUBSTRUCTURE RESULTS)

0 if SAVE command
1 if output sort is subcase modes, time or frequency steps
2 if output sort is by basic substructure
RSS(2) - Requested substructure on PRINT or SAVE subcommand
ENERGY - energy output request
UIMPRØ - nonzero if improved displacements requested
RANGE(2) - energy range requests
IREQ - start of output request block in open core relative to blank
common
LREQ - length of output request block
LBASIC - number of words for each basic in output request block.

4.131.8.1 Subroutine Name: RCØVR

1. Entry Point: RCØVR
2. Purpose: RCØVR module driver
3. Calling Sequence: CALL RCØVR

4.131.8.2 Subroutine Name: RCØVØ

1. Entry Point: RCØVØ
2. Purpose: To process user output requests from the Substructure Control deck.
3. Calling Sequence: CALL RCØVØ
4. Method:

The CASESS data block is opened and the record for the current RECOVER command is located. If no PRINT or SAVE command is present, an automatic SAVE command is generated so the solution data is not lost. A list of all basic substructures composing the requested substructure is obtained from the EQSS item. If any output requests exist on CASESS, they are read and checked for correctness. This data is then stored on a table at the bottom of open core for later use by subroutine RCØVC. The format of this table is as follows

MODULE FUNCTIONAL DESCRIPTIONS

BUF(IREQ)	UFLAG	} nonzero if any requests present
	PFLAG	
	QFLAG	
	# of points	
	# of basics	
	BASIC NAME (2)	} repeated for each basic substructure
	DISP set	
	ØLOAD set	
	SPCF set	
	SUBCASES set	
	MØDES set	
	RANGE (2)	
	VELØ set	
	ACCE set	
	STEPS set	
	BASIC or MØDAL subs	

4.131.8.3 Subroutine Name: RCØVA

1. Entry Point: RCØVA

2. Purpose:

To compute the SØLN item for the final solution substructure.

3. Calling Sequence: CALL RCØVA

4. Method:

RCØVA first copies KGG, MGG, BGG, K4GG and UGV or PHIG or UPVC or UPV to the SØF if they don't already exist. For a MRECOVER execution the PHIS item on the SØF is copied directly to UVEC. If a SØLN item for substructure FSS exists, RCØVA returns. Otherwise one of the following subroutines is used to generate the SØLN item

FUNCTIONAL MODULE RCØVR (RECOVER PHASE 2 SUBSTRUCTURE RESULTS)

RCØVSS - static solution (RFNØ 1 or 2)

RCØVMS - modal solution (RFNO 3)

RCØVDS - dynamic solution (RFNO 8 or 9)

4.131.8.4 Subroutine Name RCØVSS

1. Entry Point: RCØVSS

2. Purpose:

To compute the SØLN item for a statics solution (rigid format 1 or 2).

3. Calling Sequence: CALL RCØVSS

4. Method:

The SØLN item is initially created on SCR1, writing one GINØ logical record for each SØF group. This facilitates the generation of SØLN groups for SYMCØM and SUBCØM subcases.

Group 0 is created by reading the number of basic substructures and their names from EQSS, the number of load vectors for each substructure from LØDS, and by counting the number of Case Control subcases on CASESS. It is written as record 0 of SCR1.

GEØM4 is opened and positioned to the LØADC bulk data cards using PRELØC and LØCATE. For each regular subcase, one group of SØLN is created as follows:

- a. The LØDS item is read into open core if it is not already there.
- b. The LØADC cards are searched for the load set ID of this subcase.
- c. For each basic substructure load set referenced by the LØADC card, the load vector number is found from the EQSS and LØDS data. This number and its corresponding scale factor from the LØADC card are stored in open core.
- d. When the entire LØADC card has been processed, one group of the SØLN is complete. It is sorted on the load vector numbers and written on SCR1.

For each SYMCØM or SUBCØM subcase, one group of the SØLN is created as follows:

- a. The SYMSEQ or SUBSEQ sequence is read into open core.

MODULE FUNCTIONAL DESCRIPTIONS

b. Previous SØLN groups which are not SUBCØM or SYMCØM subcases are read into open core, one for each member of the sequence. If there is insufficient core, the LØDS data is overwritten.

c. The scale factors of these groups are multiplied by the SYMSEQ or SUBSEQ factors.

d. The new SØLN group is then sorted by load vector number and written on SCR1 following the last group written.

After SØLN is complete on SCR1, it is copied to the SØF and RCØVSS returns.

4.131.8.5 Subroutine Name RCØVMS

1. Entry Point: RCØVMS

2. Purpose:

To compute the SØLN item for a modal solution. (rigid format 3)

3. Calling Sequence: CALL RCØVMS

4. Method:

For a RECØVER command, group 0 is written directly to the SØF. If NEIGV is not positive, RCØVMS then returns. Otherwise, record 2 of LAMA is copied to the SØLN item and RCØVMS returns.

For a MRECØVER command, the contents of the LAMS item are copied to the SØLN item and RCØVMS returns.

4.131.8.6 Subroutine Name RCØVDS

1. Entry Point: RCØVDS

2. Purpose:

To compute the SØLN item for a dynamics solution (rigid formats 8 or 9).

3. Calling Sequence: CALL RCØVDS

FUNCTIONAL MODULE RCQVR (RECOVER PHASE 2 SUBSTRUCTURE RESULTS)

4. Method:

Group 0 is partially created by reading the basic substructures and their names from the EQSS, the number of load vectors for each substructure from the LØDS item, and the number of solution steps is computed from the UPVC or UPV trailer. It is then stored in the top of open core.

The DLØAD, TLØADi and RLØADi bulk data is then processed to assemble a list of possible static loads.

- a. The selected DLØAD set ID is obtained from the CASESS data block.
- b. The DLT data block is opened and the DLØAD, RLØADi and TLØADi load ID's are saved in core. The DLØAD card ID's are then checked against the case control DLØAD request. If no match is found, it is assumed the DLØAD ID points directly to RLØADi or TLØADi data. If a DLØAD card is located, the requested combination data is read from the DLT data block and stored in core.
- c. The RLØADi (Rigid Format 8) or TLØADi (Rigid Format 9) data is then processed. Each card that has been requested by the DLØAD set or the DLØAD combination data is read from the DLT and saved in core.

The LØADC data is processed to identify any static loads that actually exist. GEOM4 is opened and the LØADC bulk data cards are located. The set number for each card is read and compared against the requested load requests on each RLØADi or TLØADi card. If a match is found, the LØADC combination data is stored in core.

The substructure, load set ID pair for each LØADC entry saved is then converted to an internal load number. This is done with the data in the LØDS item. A list of these internal load ID's is then created and sorted and any duplicates are removed.

Group 0 of the SØLN item is then written using the Group 0 data obtained at first and the set of internal scalar load ID's just created.

Group 1 of the SØLN item is written by copying the appropriate number of step values from PPF (frequencies for rigid format 8) or TØL (times for rigid format 9).

MODULE FUNCTIONAL DESCRIPTIONS

Each time or frequency step is now processed with the following logic. Note that if no solution loads were requested the processing is not required. Initially, subroutine PRETAB is called to read the dynamics table data into core. The required table ID's are obtained from the RLØADi and TLØADi data. For each step:

- a. A column vector (number of scalar loads long) is zeroed in core. This will hold the load factors for each load ID for this step.
- b. Each TLØADi or RLØADi card which points to a static load is processed. It's scale factor for the current step is then calculated using the appropriate method for each card type. This factor is then multiplied by the LOADC scale factor and the results are added into the proper load slot in the in core load factor vector.
- c. The in core load factor vector is then written as the next group on the SØLN item.

An end-of-item is then written on the SØLN item and RCØVDS returns.

4.131.8.7 Subroutine Name: RCØVB

1. Entry Point: RCØVB

2. Purpose:

Back-substitution to recover displacements.

3. Calling Sequence: CALL RCØVB

4. Method:

The method used by RCOVB is clearly illustrated by a flowchart in Figure 1. From this it can be seen that if the requested substructure is the final solution structure (FSS), no recovery is necessary.

If a lower level substructure is to be processed, each subsequent higher level is checked for the existence of solution displacement vectors on the SØF. Either a previously calculated intermediate level or the actual FSS displacements will be used to recover the displacements of the requested substructure.

FUNCTIONAL MODULE RCØVR (RECOVER PHASE 2 SUBSTRUCTURE RESULTS)

The solution vectors are transformed backward through each stage of the substructure process until the requested output substructure displacements are obtained. At each stage, the solutions are copied to the SØF file and saved as a NASTRAN file U1, U2, or U3, etc., for processing of subsequent requests to follow. The SØLN item for the recovered substructure is also created at each stage by editing the SØLN item of the FSS.

At each stage of substructure recovery processing a set of displacement vectors, U_g , are known. The displacements of the next lower level substructure of substructures, U_A , are obtained by a simple transformation. The actual equation depends on the type of substructure, the current rigid format and user requests as shown below.

Static analysis:

combined substructure -

$$\{U_A\} = [H_{AB}] \{U_B\} \quad (1)$$

reduced substructure -

$$\{U_A\} = [G_{AB}] \{U_B\} + \{U_A^0\} \quad (2)$$

where

$$\{U_A^0\} = [K_{00}]^{-1} \{p_A^0\} \quad (3)$$

MODULE FUNCTIONAL DESCRIPTIONS

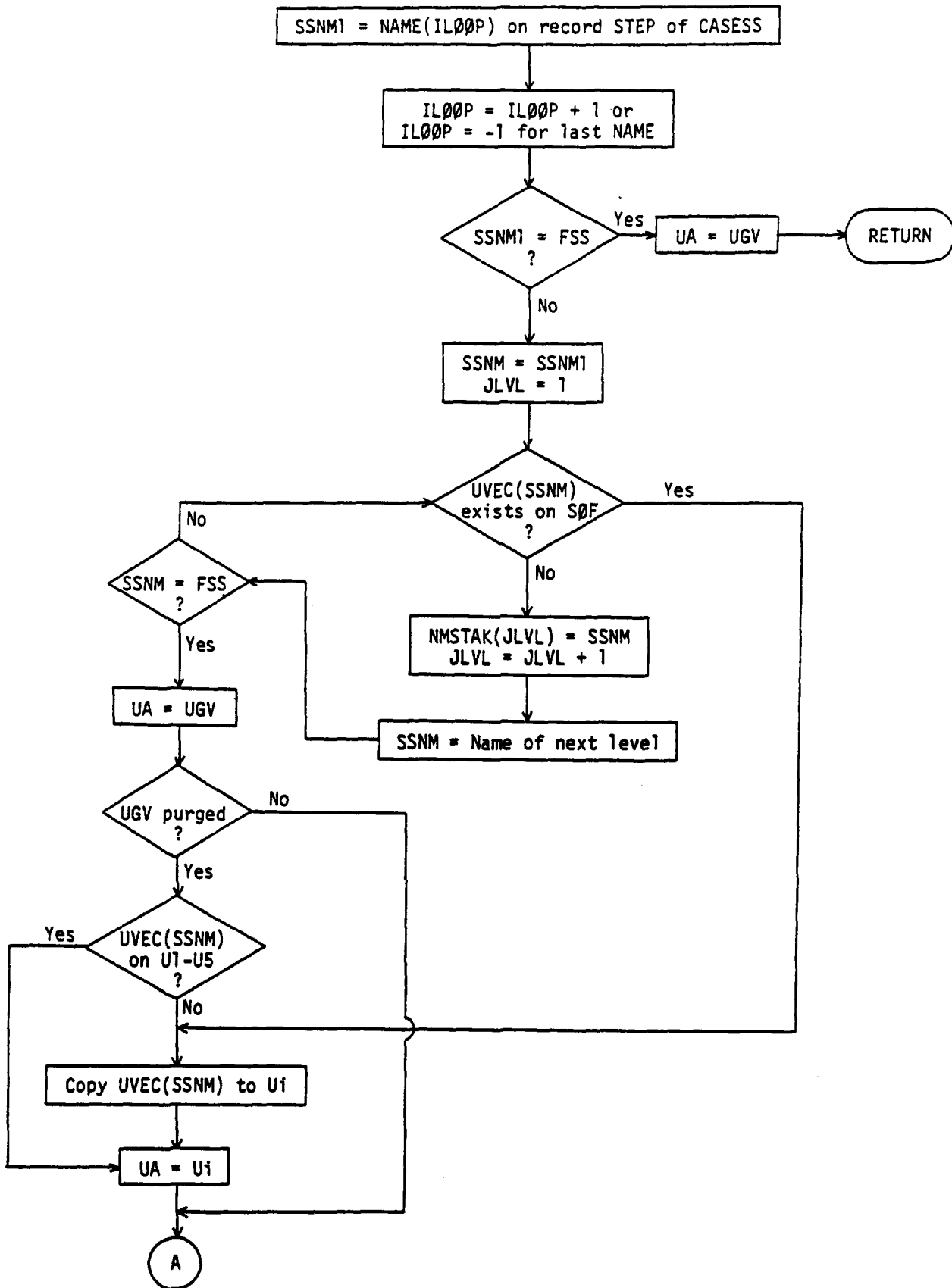
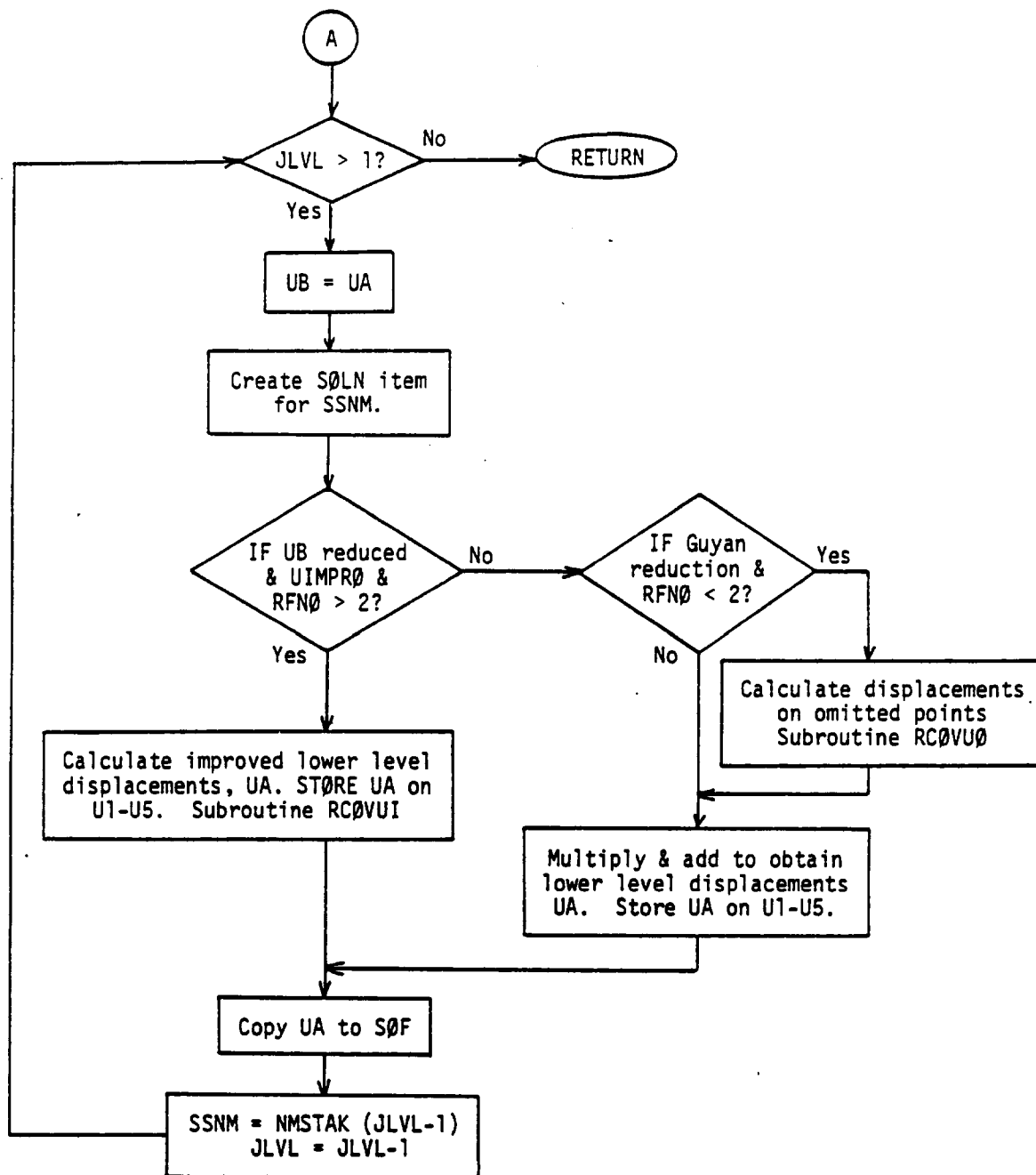


Figure 1a. Logical flow of Subroutine RC0VB.

FUNCTIONAL MODULE RCØVR (RECOVER PHASE 2 SUBSTRUCTURE RESULTS)



Definitions:

- SSNM1, RSS - Name of substructure which user specified to recover
- FSS - Final solution substructure
- UGV, Ui - Module input and output data blocks
- NMSTAK - Array of names of substructures which must be successively recovered to recover SSNM1
- UIMPRØ - User-requested improved displacement flag
- SSNM - Name of substructure currently being recovered

Figure 1b. Logical flow of Subroutine RCØVB (cont'd).

MODULE FUNCTIONAL DESCRIPTIONS

Modal analysis or dynamic analysis.

combined substructure -

$$\{U_A\} = [H_{AB}] \{U_B\} \quad (4)$$

reduced substructure - (no improved displacements requested)

$$\{U_A\} = [G_{AB}] \{U_B\} \quad (5)$$

reduced substructure - (improved displacements requested)

$$\{\tilde{U}_A\} = [G_{AB}] \{U_B\} \quad (6)$$

$$\{P_A^V\} = -[M] \{A_A\} - [B] \{V_A\} \quad (7)$$

where $\{A_A\}$ and $\{V_A\}$ are either computed from $\{\tilde{U}_A\}$ or calculated with Equation (6).

$$\{U_A^0\} = [K_{00}]^{-1} \{P_A^0 + P_A^V\} \quad (8)$$

$$\{U_A\} = \{U_A^1\} + \{U_A^0\} \quad (9)$$

where:

$$\{U_A^1\} = \{\tilde{U}_A\} \text{ for REDUCE} \quad (10)$$

$$\{U_A^1\} = \left[\begin{array}{c} -I \\ -G \end{array} \right] \{U_B\} \text{ for MREDUCE} \quad (11)$$

The vectors P_A^0 are obtained by combining the vectors on the PØVE item for substructure A on the SØF. The PØVE vectors are multiplied by the solution load factors and added to produce the solution loads.

Another way of explaining the task of substructure data recovery is to visualize the organization of the structure as an analogy to a management organization chart. An example is shown in Figure 2.

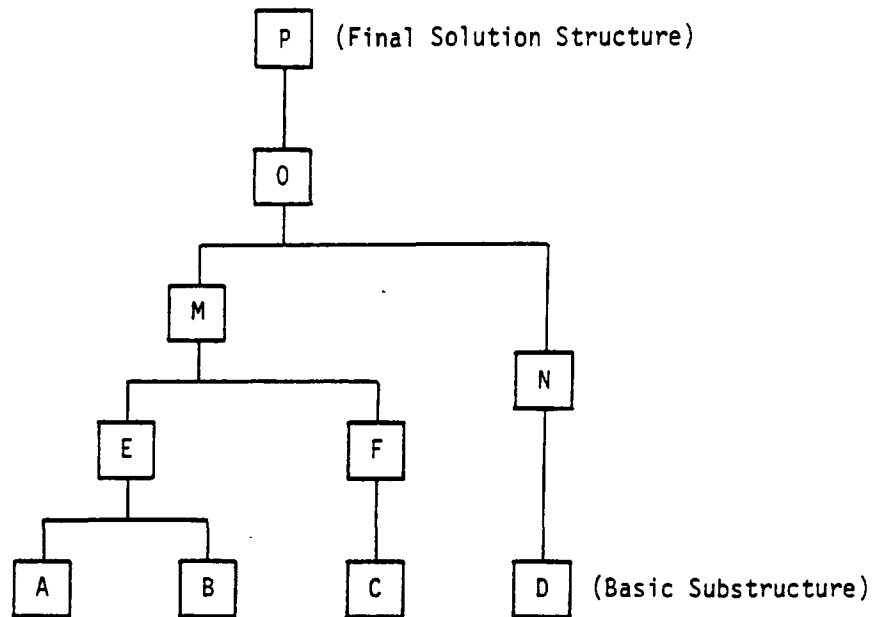


Figure 2. Example of substructure organization.

Substructures A, B, C, and D are basic Phase 1 substructures. They are combined and reduced repeatedly to produce the solution structure, P, which has a solution U_p .

The task of data recovery is similar to a downward flow of information to selected units. If U_A is requested, calculate:

$$U_p \rightarrow U_O \rightarrow U_M \rightarrow U_E \rightarrow U_A$$

U_B may then be obtained from U_E .

If U_C is requested and U_M is known, calculate:

$$U_M \rightarrow U_F \rightarrow U_C$$

If U_D is requested and U_O is known, calculate:

$$U_O \rightarrow U_N \rightarrow U_D$$

This method is the most economical when the number of columns in the solution matrix, U_p , is less than the size of the K_p matrices. A future enhancement would be to implement an alternate method of recovery to improve the efficiency of dynamic matrix recovery where a large number of columns are in U_p . This method would recover the displacements of a lower level substructure directly by first forming the proper transformation matrix consisting of the products of any intermediate transformation matrices.

MODULE FUNCTIONAL DESCRIPTIONS

4.131.8.8 Subroutine RCØVLS

1. Entry Point: RCØVLS

2. Purpose:

To create the SØLN item of a lower level substructure by editing the SØLN item of the solution substructure FSS.

3. Calling Sequence: CALL RCØVLS (LASTSS)

LASTSS - Substructure currently being recovered for which the SØLN item is to be calculated - BCD - input.

4. Method:

The method used to edit the SØLN item of the solution substructure depends on the current rigid format.

Static analysis -

- a. A list of the component substructures of LASTSS is obtained from it's EQSS item.
- b. Group 0 of the new SØLN item is obtained by editing out the unneeded component names from group 0 of the SØLN item for FSS.
- c. Each additional group is created by editing out the load vectors which do not belong to the components of LASTSS.

Modal analysis -

- a. The new SØLN item is a direct copy of the SØLN item for the solution substructure FSS.

Dynamic analysis -

- a. A list of the component substructures of LASTSS is obtained from it's EQSS item.
- b. Group 0 of the new SØLN item is obtained by editing out the unneeded components and the load vector ID's which no longer apply from group 0 of the SØLN item for FSS.

FUNCTIONAL MODULE RCØVR (RECOVER PHASE 2 SUBSTRUCTURE RESULTS)

- c. Group 1 is copied from the SØLN item of FSS to LASTSS.
- d. The remaining groups are created by editing out the scale factors for the loads which no longer apply.

4.131.8.9 Subroutine RCØVUØ

1. Entry Point: RCØVUØ

2. Purpose:

To calculate the displacements on any omitted points when recovering a reduced substructure.

3. Calling Sequence: CALL RCØVUØ (PID,UAØ,LASTSS)

PID - Optional inertia and damping load matrix, 0 set, to be added to the static omitted point loads - integer - input, output.

UAØ - GINØ file to contain the resultant displacements. UAØ will be set to zero if no displacements can be generated - integer - input.

LASTSS - Name of the substructure to generate the displacements for - bcd - input.

4. Method:

The following setups are used to generate the displacements on the omitted points.

a. Any static loads on the omitted points, P_0 , are generated from the SØLN and PØVE items by subroutine RCØVSL.

b. If the optional inertial and damping force load, P_0^V , are provided on file PID, they are added to any static loads on the omitted points.

$$\{P_0^T\} = \{P_0\} + \{P_0^V\}$$

otherwise:

$$\{P_0^T\} = \{P_0\}$$

c. If no static loads exist and PID is not provided, UAØ is set to zero and RCØVUØ returns.

MODULE FUNCTIONAL DESCRIPTIONS

- d. The existence of the LMTX item for substructure LASTSS is checked. If the item exists, it is copied from the SØF and processing skips to step g.

$$[L_{00}] = \text{LMTX item}$$

- e. The KMTX item for subroutine LASTSS is copied from the SØF and partitioned to the 0 set:

$$[KMTX] \rightarrow \left[\begin{array}{c|c} K_{00} & 0 \\ \hline 0 & 0 \end{array} \right]$$

The partitioning vector is obtained from item UPRT on the SØF.

- f. The stiffness matrix for the omitted points is decomposed.

$$[K_{00}] \rightarrow [L_{00}] [L_{00}]^T$$

- g. The 0 size displacements are obtained through a forward backward substitution.

$$\{U_0\} = [L_{00}] [L_{00}]^T \{P_0\}$$

- h. The 0 size displacements are expanded to A size

$$\{U_a\} \leftarrow \left\{ \begin{array}{c} U_0 \\ 0 \end{array} \right\}$$

and returned on file UAØ. For transient analysis a null velocity and acceleration vector is merged in with each displacement vector so the matrix is compatible with the recovered displacements.

4.131.8.10 Subroutine RCØVUI

1. Entry Point: RCØVUI

2. Purpose:

To compute the improved lower level displacements on a reduced substructure.

3. Calling Sequence: CALL RCØVUI (UB, LASTSS)

UB - GINØ file where displacements of the higher level substructure are stored - integer - input.

FUNCTIONAL MODULE RCØVR (RECOVER PHASE 2 SUBSTRUCTURE RESULTS)

LASTSS - Substructure for which improved displacements are to be recovered - bcd - input.

4. Method:

The following steps are used to generate the improved displacements for substructure LASTSS.

- a. The partial displacements are generated.

$$\{\tilde{U}\} = [G_{AB}] \{U_B\}$$

- b. The velocities and acceleration vectors, $\{\tilde{V}\}$ and $\{\tilde{A}\}$, are generated using these partial displacements by subroutine RCØVVA.

- c. The inertial and damping loads are then calculated

$$\{P^V\} = -[M] \{\tilde{A}\} - [B] \{\tilde{V}\}$$

- d. The inertial and damping loads are partitioned to Ø set.

$$\{P^V\} \rightarrow \begin{Bmatrix} P^V \\ -\frac{0}{0} \\ 0 \end{Bmatrix}$$

- e. The displacements on the omitted points, U_0 , are computed from the static, inertial and damping loads by subroutine RCØVUØ.

- f. If modal reduction was used, the item GIMS = G is used in the following process

Partition:

$$\{\tilde{U}_A\} \rightarrow \begin{Bmatrix} \tilde{U}_b \\ \tilde{U}_0 \end{Bmatrix}$$

Multiply and Add:

$$\{\delta U_0\} = [G] \{\tilde{U}_b\} - \{\tilde{U}_0\}$$

Note $\{\delta U_0\} = 0$ for the REDUCE case.

- g. The final displacements are then calculated

$$\{U\} = \{\tilde{U}\} + \{U_0\} + \{\delta U_0\}$$

MODULE FUNCTIONAL DESCRIPTIONS

4.131.8.11 Subroutine RCØVC

1. Entry Point: RCØVC

2. Purpose:

Compute reaction forces and generate output data blocks for later processing by ØFP.

3. Calling Sequence: CALL RCØVC

FUNCTIONAL MODULE RCØVR (RECOVER PHASE 2 SUBSTRUCTURE RESULTS)

4. Method:

The Case Control subcases on CASESS are scanned and flags are set for displacements, velocities, accelerations or eigenvectors, applied loads, and reaction force output requests although these requests will not override those given in the substructure control deck. If the loads or reactions have been requested and the current run is a statics analysis, the applied loads are generated as follows. If the requested substructure is the final solution structure (FSS), the loads may be found on PG, if it is not purged. Otherwise, they are computed from the SØLN and PVEC item using subroutine RCØVSL.

If the current run is a frequency response run, the velocity and acceleration vectors are computed by Subroutine RCØVVA for possible output.

Next, the flag for computing and printing reactions is checked. Note that reactions may be requested by the user using the SPCFØRCE Case Control card. However, reactions are printed at all requested points regardless of whether they are single point constraints. If requested, the reactions are computed by subroutine RCØVQV.

Output Processing: A prototype of the ØFP ID record is created. Note that the rightmost words of the subtitle are replaced with 'SUBSTRUCTURE namexxxx'. The list of the SIL numbers is read into open core from EQSS. (An open core diagram may be found on the comment cards of this subroutine.) The header records are written on the output data blocks.

All of the following may be executed repeatedly, once for each basic substructure, if the value of the variable IØPT, set by the SØRT command, is equal to 2. This causes all output for each basic substructure to be grouped together. Otherwise, the following is executed once only for all basic substructures. This causes all output for each subcase to be grouped together.

The EQSS data for the basic substructure(s) currently being processed is read into open core if it will fit. If it does not fit, spill logic will be invoked. For each subcase, mode or step the following steps are performed:

MODULE FUNCTIONAL DESCRIPTIONS

- a. The output requests, titles, and set information is read from CASESS.
- b. Each type of output vector is processed turn, in the following steps.
- c. The vector to be printed is unpacked into open core.
- d. Each basic substructure to be printed for this subcase is processed in turn in the following steps.
- e. The \emptyset FP ID record is completed and written on the output data block.
- f. Each external grid-point ID found in the EQSS data for the current basic substructure which is a member of the output set is used to generate one line of output. Each line of output corresponds to six degrees of freedom. Nonexistent degrees of freedom will have zero values in the printout.

When all basic substructures, subcases, and output types have been processed, RC \emptyset VC returns.

4.131.8.12 Subroutine RC \emptyset VSL

1. Entry Point: RC \emptyset VSL
2. Purpose:

To build a static load matrix for use in the Phase II and Phase III recovery processes.

3. Calling Sequence: CALL RC \emptyset VSL

(NAME,ITEM,IVEC,SCR1,SCR2,SCR3, \emptyset VEC,Z,IZ,LC \emptyset RE,FIRST)

NAME - Name of substructure being recovered - BCD array - input
ITEM - S \emptyset F item containing input load vectors - BCD - input
IVEC - GIN \emptyset file containing input load vectors - BCD - input
SCR1 - Scratch file - integer - input
SCR2 - Scratch file - integer - input
SCR3 - Scratch file - integer - input
 \emptyset VEC - GIN \emptyset file containing resultant load vectors - integer - input
Z,IZ - Open core array - real, integer - input
LC \emptyset RE - Length of open core array - integer - input

FUNCTIONAL MODULE RCØVR (RECOVER PHASE 2 SUBSTRUCTURE RESULTS)

FIRST - Logical - input

.FALSE. - Build factor matrix from SØLN data.

.TRUE. - Bypass building of factor matrix and use matrix on SCR1 instead.

4. Method:

If necessary, the load matrix is copied from the specified SØF item onto scratch file SCR2 using subroutine MTRXI. If FIRST = .FALSE., the load scale factor data from the SØLN item is assembled into the load factor matrix and packed onto SCR1. If FIRST = .TRUE., the matrix of scale factors is assumed to already exist on SCR1. The load combination matrix is then computed via MPYAD.

$$\{P\} = [FACT] \{PVEC \text{ or } PØVE\}$$

4.131.8.13 Subroutine RCØVVA

1. Entry Point: RCØVVA

2. Purpose:

To compute the velocities and acceleration vectors for a given set of displacement vectors.

3. Calling Sequence: CALL RCØVVA

(IN,INTYP,ØUTT,ØUTU,ØUTV,ØUTA,SSNM,RZ,DZ,CZ)

IN - file containing the input displacement vectors - integer - input.

INTYP - describes operation to be performed - integer - input.

0 - IN contains U only and V and A are to be calculated

+1 - IN contains U, V and A which are to split apart

-1 - ØUTU, ØUTV and ØUTA are to be merged onto ØUTT.

ØUTT - output file for U, V and A - integer - input.

ØUTU - output file for U only - integer - input.

ØUTV - output file for V only - integer - input.

ØUTA - output file for A only - integer - input.

SSNM - substructure for which velocities and accelerations are to be computed - bcd - input.

RZ,DZ,CZ - start of open core array - real, double and complex - output.

MODULE FUNCTIONAL DESCRIPTIONS

4. Method:

The method used to generate the velocity and accelerations depends on the value of INTYP and the current rigid format.

INTYP = 0

The equations used to calculate the velocities and accelerations are shown below. The results are written to either ØUTT or ØUTU, ØUTV, and ØUTA as requested.

Real Normal Modes:

$$\begin{aligned}\{V\} &= 0 \\ \{A\} &= -[2\pi f]^2 \{U\}\end{aligned}$$

where f are the frequencies.

Complex Normal Modes:

$$\begin{aligned}\{V\} &= [P] \{U\} \\ \{A\} &= [P] \{V\}\end{aligned}$$

where $[P]$ is the diagonal matrix of complex poles or eigenvalues.

Frequency Response:

$$\begin{aligned}\{V\} &= [2\pi if] \{U\} \\ \{A\} &= [2\pi if] \{V\}\end{aligned}$$

where f are the frequencies.

Transient response:

Since V and A are already calculated in the solution, INTYP = 1 is assumed.

INTYP = 1

The IN vectors are unpacked and split to the three output files ØUTU, ØUTV, and ØUTA. Any of the three outputs may be purged.

INTYP = -1

The vectors on files ØUTU, ØUTV, and ØUTA are unpacked and merged to file ØUTT. All three inputs must exist.

4.131.8.14 Subroutine RCØVQV

1. Entry Point: RCØVQV

2. Purpose:

To compute the reaction forces for the requested substructure.

3. Calling Sequence: CALL RCØVQV

4. Method:

The equations used to calculate the reaction forces are shown below.

Static analysis:

$$\{Q\} = [K]\{U\} - \{P\}$$

Real normal modes:

$$\{Q\} = [K]\{U\} + [M]\{A\}$$

Complex normal modes:

$$\{Q\} = [K]\{U\} + [B]\{V\} + [M]\{A\}$$

Dynamic analysis:

$$\{Q\} = [K]\{U\} + [B]\{V\} + [M]\{A\} - \{P\}$$

If either the mass or damping matrices do not exist, that term of the equation is ignored. The velocity and acceleration vectors are calculated using subroutine RCØVVA.

4.131.8.15 Subroutine RCØVE

1. Entry Point: RCØVE

2. Purpose:

To print the energies on the included and excluded modes of a modal reduced substructure.

3. Calling Sequence: CALL RCØVE

4. Method:

Energy Calculations

The following steps are followed to calculate the modal energies on the requested substructure.

MODULE FUNCTIONAL DESCRIPTIONS

- a. If the current solution is a static analysis, RCØVE returns because no modal energy calculations can be made.
- b. The name of the higher level substructure, HIGHER, to the requested substructure, RSS, is obtained.
- c. If HIGHER was not the results of a MREDUCE or a CREDUCE, RCØVE returns.
- d. The EQSS of HIGHER is read to locate the SIL number of the first modal dof. Also the number of any inertial dof are counted.
- e. If HIGHER was the result of a MREDUCE and the solution rigid format was 3 or 8, the energies on the excluded modes of RSS are calculated with subroutine RCØVEM.
- f. The modal energies on the modal coordinates of HIGHER are calculated with subroutine RCØVIM.

Output Processing

The following steps are performed to output the modal energies.

- a. The LAMS item for substructure RSS is read and the frequency, mode number and mode type, included or excluded, is kept for each mode.
- b. The CASESS data block is read to locate the requested ENERGY set information.
- c. For each column of the solution matrix the following operations are performed to generate the energy output.
 - i. The frequency or time step value for the current column is obtained from the SØLN item for substructure RSS. If output for this step is requested via the ENERGY and RANGE commands, processing continues. Otherwise we skip to the next step.
 - ii. The computed kinetic and potential energies for both the included and excluded modes for this step are unpacked.

FUNCTIONAL MODULE RCØVR (RECOVERY PHASE 2 SUBSTRUCTURE RESULTS)

iii. For each modal coordinate a line of output is generated.

iv. A final line of output is generated which gives the total energies for this step.

4.131.8.16 Subroutine RCØVEM

1. Entry Point: RCØVEM

2. Purpose:

To compute the energies on the modes of a modal reduced substructure which were excluded from the reduce.

3. Calling Sequence: CALL RCØVEM (NØEXCL,NRØWE)

NØEXCL - Set to true if no excluded mode energies can be calculated - logical - output.

NRØWE - The number of excluded modes - integer - output.

4. Method:

From the last group on LAMS, a partitioning vector is generated to separate the included and excluded modes. The PHIS matrix for the requested substructure is then copied from the SØF and partitioned.

$$[\phi] \rightarrow [\phi_n \mid \phi_k]$$

The modal load matrix is then calculated from the excluded modes, static loads and reaction forces.

$$[P_K] = [\phi_K]^T ([P] + [Q])$$

The modal mass and frequency for each excluded mode are then read from LAMS and stored in core.

The excluded mode energies are calculated by looping through the modal load matrix and the SØLN item. The modal energies for each solution step and each excluded mode i , are calculated as follows

MODULE FUNCTIONAL DESCRIPTIONS

$$\delta_k^d(i) = \frac{-S^2 P_k(i)}{M_k W_k^4 \left(1 + \frac{S^2}{W_k^2}\right)}$$

$$\delta_k^s(i) = \frac{P_k(i)}{M_k W_k^2}$$

and

$$\text{Potential} = \frac{1}{2} M_k W_k^2 \left| (2 \delta_k^s + \delta_k^d) \delta_k^d \right|$$

$$\text{Kinetic} = \left| \frac{S^2}{W_k^2} \right| \times \text{Potential}$$

where the following data describes each excluded mode

M_k = Modal mass

W_k = 2π * Modal frequency

and the following data describes the solution steps

Real eigenvalues and frequency response:

$$S^2 = -(2\pi f)^2$$

Complex eigenvalues:

S = pole

The modal energies are only calculated for those modes excluded because of the RANGE or NMAX parameters. Modal energies for modes excluded because of nonpartition are set to zero.

4.131.8.17 Subroutine RCØVIM

1. Entry Point: RCØVIM

2. Purpose:

To compute the energies on the modal coordinates of a modal reduced substructure.

FUNCTIONAL MODULE RCØVR (RECOVERY PHASE 2 SUBSTRUCTURE RESULTS)

3. Calling Sequence: CALL RCØVIM (HIGHER)

4. Method:

The displacement matrix for substructure HIGHER is copied from the SØF and the accelerations are computed by subroutine RCØVVA.

The kinetic energies are computed by first computing the inertia forces

$$\{F_I\} = [M] \{V\}$$

The individual kinetic energy terms are computed by a scalar multiplication

$$\{T\} = \{F_I\} \{a\}$$

and the total kinetic energies are the sum of each column.

The potential energies are computed by first computing the static forces

$$\{F_S\} = [K] \cdot \{u\}$$

The individual energy terms are computed by a scalar multiplication

$$\{V\} = \{F_S\} \cdot \{a\}$$

and the total potential energies are the sum of each column.

The total energy terms are appended to the end of each column in the output matrices.

4.131.9 Design Requirements

RCØVR uses the following matrix utility routines:

MPYAD

SDCØMP

FBS

SADD

MERGE

PARTN

FUNCTIONAL MODULE RCØVR3 (RECOVER SUBSTRUCTURE RESULTS FOR PHASE 3)

4.132 FUNCTIONAL MODULE RCØVR3 (RECOVER SUBSTRUCTURE RESULTS FOR PHASE 3)

4.132.1 Entry Point: RCØVR3

4.132.2 Purpose

To retrieve data for a basic substructure which has been stored on the SØF during Phase 2 and write it on standard NASTRAN data blocks for further processing. RCØVR3 also combines loads and enforced displacements which were specified in Phase 1 to form the vectors corresponding to the actual Phase 2 solution.

4.132.3 DMAP Calling Sequence

RCØVR3 PG,PS,PØ,YS/UAS,QAS,PGS,PSS,PØS,YSS, $\left\{ \begin{array}{l} \text{PPF} \\ \text{LAMA} \\ \text{TØL} \end{array} \right\} / \text{C,N,RFNØ/C,N,NAME \$}$

4.132.4 Input Data

4.132.4.1 GINØ Data Blocks

- PG - Static load matrix, g-set
- PS - Static load matrix, s-set
- PØ - Static load matrix, o-set
- YS - Enforced displacement matrix, s-set

Note: Any data block may be purged.

4.132.4.2 SØF Items

- UVEC - Displacement matrix, g-set of Phase 2 (equivalent to a set of Phase 3)
- QVEC - Reaction force matrix, g-set of Phase 2 (equivalent to a set of Phase 3)
- SØLN - Solution data

Note: Any item may not exist.

4.132.5 Output Data

4.132.5.1 GINØ Data Blocks

- UAS - Displacement matrix for substructure solution - a-set
- QAS - Reaction force matrix for substructure solution - a-set
- PGS - Static load matrix for substructure solution - g-set
- PSS - Static load matrix for substructure solution - s-set
- PØS - Static load matrix for substructure solution - o-set

MODULE FUNCTIONAL DESCRIPTIONS

YSS - Enforced displacement matrix for substructure solution - s-set
LAMA - Real eigenvalue table for substructure solution
PPF - Frequency list for frequency response analysis
TØL - Time step list for transient response analysis

Notes:

1. UAS may be purged only if UVEC does not exist.
2. QAS may be purged only if QVEC does not exist.
3. PGS, PSS, PØS, or YSS may be purged only if PG, PS, PØ, or YS, respectively, is purged.
4. LAMA may be purged if SØLN does not exist, or if RFNØ ≠ 3.
PPF may be purged if SØLN does not exist, or if RFNØ ≠ 8.
TØL may be purged if SØLN does not exist, or if RFNØ ≠ 9.

4.132.5.2 SØF Items

None

4.132.6 Parameters

RFNØ - Integer - input. Rigid Format number.
NAME - BCD - input. Name of the basic substructure to be processed.

4.132.7 Method

RCØVR3 begins by checking the correspondence of input data blocks and items to the output data blocks. If any input data block or item exists and its corresponding output data block does not, a fatal error occurs.

The displacements are copied from the UVEC item to UAS and the reaction forces are copied from QVEC to QAS. Both copy operations use SØF subroutine MTRXI. When they are complete, RCØVR3 branches on solution type.

Rigid Format 1, 2 - Statics, Inertial Relief

A compatibility check is made on the group 0 data of the SØLN item. If necessary, a null reaction force matrix is written to prevent a fatal error in the UMERGE module. For each input

FUNCTIONAL MODULE RCØVR3 (RECOVER SUBSTRUCTURE RESULTS FOR PHASE 3)

load matrix, RCØVSL is called to combine the load vectors together based on the scale factors found in the SØLN item.

Rigid Format 3 - Normal Modes

Record 1 of LAMA (the ØFP ID record) is generated in open core from data in SØLN group 0 and /ØOUTPUT/. After it has been written out, SØLN group 1 is copied to LAMA record 2.

Rigid Format 8 - Direct Frequency Response

The two-word file header and the frequencies from SØLN group 1 are written to PPF record 0.

Rigid Format 9 - Direct Transient Response

The time steps from SØLN group 1 are written to TØL record 1.

4.132.9 Design Requirements

Open core is in common /RCØV3X/.

4.132.10 Diagnostic Messages

RCØVR3 may issue the following messages:

3002,3008,3037

6321,6322,6323,6324

FUNCTIONAL MODULE REDUCE (REDUCTION OF SUBSTRUCTURE DEGREES OF FREEDOM)

4.133 FUNCTIONAL MODULE REDUCE (REDUCTION OF SUBSTRUCTURE DEGREES OF FREEDOM)

4.133.1 Entry Point: REDUCE

4.133.2 Purpose

Given a substructure, A, and a specified boundary set, N_B , defined in terms of component substructures of A, this module performs two basic functions to generate the new reduced structure, B:

1. Computes PVX - a Boolean vector whose elements are 1 if the degree of freedom is retained from A to B; and 0 if the degree of freedom is to be reduced out. PVX then becomes the reduction partitioning vector used in a series of DMAP alters to define structure B. Also, the USX (a USET vector) is generated for use in matrix operation modules. And,
2. Generates table data for the new structure, B, to be written on the updated Substructure Operating File (SOF).

4.133.3 DMAP Calling Sequence

REDUCE CASECC,GEOM4/PVX,USX,INX/C,N,STEP/V,N,DRY \$

4.133.4 Input Data

4.133.4.1 GINØ Data Blocks

CASECC - The Case Control data block from which the reduce instructions are obtained. These instructions include:

REDUCE A
BOUNDARY = BSET
NAME = B
OUTPUT = n

GEOM4 - NASTRAN data block of displacement set definitions and substructuring bulk data.

Extracted from this block is the BDYC data corresponding to the boundary set given in CASECC and the BDYS and BDYS1 data specified on BDYC.

4.133.4.2 SOF Items

EQSS	- Substructure equivalence tables	}	For original substructure
BGSS	- Basic gridpoint definition table		
ESTM	- Coordinate system transformation matrices		
LØDS	- Load set data		
PLTS	- Plot set data		

MODULE FUNCTIONAL DESCRIPTIONS

4.133.5 Output Data

4.133.5.1 GINØ Data Blocks

- PVX - The partitioning vector for substructure reduction
- USX - USET equivalent vector (U_a and U_o only) corresponding to PVX
- INX - Identity matrix to be merged with the reduction transformation to define

$$[G_{AB}] = \begin{bmatrix} I \\ -G_o \end{bmatrix}$$

4.133.5.2 SØF Items

- | | | |
|--|---|--------------------------|
| EQSS - Substructure equivalence tables | } | For reduced substructure |
| BQSS - Basic grid point definition table | | |
| CSTM - Coordinate system transformation matrices | | |
| LØDS - Load set data | | |
| PLTS - Plot set data | | |

4.133.6 Parameters

- STEP - Identifies the Case Control step from which the REDUCE module is executed. Used to pick up the required data from CASECC.
- DRY - A flag which is set during module execution if error conditions prevent a requested GØ operation.

4.133.7 Method: The following logical sequence is performed by the REDUCE MODULE:

- a. Initialize GINØ and SØF files and IØ buffers.
- b. Process the Case Control data block.

The allowable case control mnemonics for the REDUCE operation are NAMA, the name of the pseudostructure being reduced; NAMB, the name given to the resultant pseudostructure; BØUN, the boundary set identification number; and ØUTP, the selective output control flag. These values are extracted from the CASECC input data block.

- c. The names of all the component substructures contained in the pseudostructure being reduced are read from the SØF and placed in the first NWDS of open core.

FUNCTIONAL MODULE REDUCE (REDUCTION OF SUBSTRUCTURE DEGREES OF FREEDOM)

- d. Read the boundary set data into open core.

The BDYC bulk data is accessed from file GEØM4 and the set ID corresponding to BØUN is located and read into open core. Note that each name for which a boundary set is specified must be a Phase I basic substructure. The table residing in open core is

NAME ₁	SID ₁
NAME ₂	SID ₂
⋮	⋮

This list is then sorted on SID (set ID of boundary set for NAME).

- e. Process the BDYS and BDYS1 bulk data.

For each SID specified in step 'd', the appropriate input in terms of either BDYS or BDYS1 bulk data is located on file GEØM4. Any BDYS1 data is reprocessed to the same form as BDYS data and written into open core as:

BDYS	[(SID)	G _b	C _b
			G _b	C _b
			⋮	⋮
		(SID)	G _b	C _b
			G _b	C _b
			⋮	⋮
		⋮	⋮	⋮

BDYS1	[(SID)	G _b	C _b
			G _b	C _b
		⋮	⋮	⋮
		⋮	⋮	⋮

In the above table the component code as specified on the bulk data card has been translated to the bit pattern used in substructure analysis. The SID numbers are not written in the table but a set of pointers into this table is kept.

- f. Process the EQSS for each component substructure.

The EQSS is read into core for the given substructure, i.e.,

MODULE FUNCTIONAL DESCRIPTIONS

SUB _i	GRID	I _p	C	C _B
	⋮	⋮	⋮	⋮

and a boundary set is found for the substructure, if no such set exists, the entire substructure will be reduced out.

Subroutine EQSCØD (See Sec. 4.133.8.1) is called to account for the possibility that grid points may be represented by several I_p numbers. The boundary set specified is located on file SCR1 and read two words at a time. The grid ID is located in the EQSS by calling subroutine GRIDIP (See Sec. 4.133.8.2). For each grid point in the boundary set the retained component degrees of freedom are checked against the original degrees of freedom to insure they are an appropriate subset. The values I_p and C_B are written on scratch file SCR2. C_B represents the retained degrees of freedom and is defined by the logical "AND" function:

$$C_B = C_B \wedge C_b$$

g. Process master SIL list.

Read the master SIL list (last record of EQSS) into core, augment with the array C_{NEW} which will define degrees of freedom in the reduced structure and set up table:

<u>SIL_{OLD}</u>	<u>C_{OLD}</u>	<u>C_{NEW}</u>
⋮	⋮	0
⋮	⋮	0
⋮	⋮	0
⋮	⋮	⋮

For each component substructure on SCR2, read I_p and C_B and compute C_{NEW} by a logical "OR"

$$C_{NEW I_p} = C_{NEW I_p} \vee C_B$$

When this has been accomplished the table in core will be, for example,

<u>SIL_{OLD}</u>	<u>C_{OLD}</u>	<u>C_{NEW}</u>
1	110101	000101
4	111111	111111
10	000111	0
13	000111	0
17	111111	101101
⋮	⋮	⋮

Those DØF in C_{OLD} but not in C_{NEW} will be eliminated during the reduction process

FUNCTIONAL MODULE REDUCE (REDUCTION OF SUBSTRUCTURE DEGREES OF FREEDOM)

The partitioning vector, PVX, is then defined. For this step it is necessary to cast the degree of freedom codes, C, in a different form called the "DØF numbers." This simply defines the bits in the DØF code that are "ØN" starting with the least significant bit. For example,

If DØF code = 101101
 then DØF numbers = 1,3,4,6
 If DØF code = 100011
 then DØF numbers = 1,2,6 etc.

Now we build PVX_i using the table above.

Starting with the first SIL_{OLD} in the table, allocate $SIL_{OLD_{i+1}} - SIL_{OLD_i}$ rows to PVX.

Find the "DØF numbers" for C_{OLD} and C_{NEW} . Order these for C_{OLD} , e.g.,

C_{OLD} = 101101
 DØF Numbers = 1,3,4,6
 C_{NEW} = 000101
 DØF Numbers = 1,3

Find the DØF numbers in C_{OLD} that are retained in C_{NEW} and their order, i.e., for the example above, C_{NEW} DØF numbers = 1 and 3, correspond to the first two components of C_{OLD} . Therefore, rows of PVX numbered $SIL_{OLD} + 1 - 1$ and $SIL_{OLD} + 2 - 1$ are set to 1 and the remaining two rows, $SIL_{OLD} + 3 - 1$ and $SIL_{OLD} + 4 - 1$, are set to 0. This is repeated for each SIL_{OLD} , and the PVX data block is written in packed format.

Now the USX vector is built. Using the PVX vector, compute USX (the USET equivalent vector) setting only the flags for U_a and U_o , and write the USX data block.

h. Build new EQSS file.

Replace SIL in current list with I_{pNEW} (new internal point numbers of retained degree of freedom subsets). If $C_{NEW} = 0$ do not define an I_p for that point, e.g.,

MODULE FUNCTIONAL DESCRIPTIONS

<u>I_{PNEW}</u>	<u>C_{NEW}</u>
1	000101
2	111111
0	0
0	0
3	101101
⋮	⋮

Then, read the old EQSS file, find I_{PNEW} in position I_{POLD} and replace the old I_p and C if $C_{NEW} \neq 0$. Write the new EQSS file for the reduced pseudostructure.

Compute the new master SIL list:

<u>SIL</u>	<u>C_{NEW}</u>
1	000101
3	111111
9	101101
⋮	⋮

and write it on the SØF only if $C_{NEW} \neq 0$.

- i. Generate new BGSS, CSTM and LØDS items for the SØF.
- j. Generate INX.

INX is an identity matrix equal in size to the number of retained degrees of freedom after reduction. Write the INX data block in packed format.

4.133.8 Design Requirements

Open core resides in common block /REDZZZ/.

4.133.9 Diagnostic Messages

Diagnostic messages 6536-6538 and 6601-6618 are issued from module REDUCE. Errors in input are treated as fatal when no further processing is possible. Some errors which may be due to ambiguities in data are diagnosed, but processing continues.

FUNCTIONAL MODULE SGEN (SUBSTRUCTURE TABLE GENERATOR)

4.134 FUNCTIONAL MODULE SGEN (SUBSTRUCTURE TABLE GENERATOR)

4.134.1 Entry Point: SGEN

4.134.2 Purpose

This module will produce data blocks defining a pseudostructure as required for the SOLVE operation. Bulk data cards with substructure name identification are converted to their non-substructure equivalents. Tables defining the pseudostructure as a set of N scalar points, where N is the number of degrees of freedom in the g set, are generated. This allows the execution of general purpose functional modules to obtain the solution.

4.134.3 DMAP Calling Sequence

```
SGEN    CASECC,GEØM3,GEØM4,DYNAMICS/CASESS,CASEI,GPL,EQEXIN,GPDT,BGPDT,SIL,GE3S,GE4S,  
        DYNS/S,N,DRY/*NAMESØLS*/S,N,LUSET/S,N,NØGPDT $
```

4.134.4 Input Data

4.134.4.1 GINØ Data Blocks

```
CASECC  - Substructure case control table  
GEØM3   - Load data  
GEØM4   - Constraint data and substructure load combination data  
DYNAMICS - Dynamics data
```

Notes:

1. CASECC may not be purged.
2. GEØM3 may be purged.
3. GEØM4 may be purged only if no MPC, SPC or LØAD set selections have been made in the Case Control Deck.
4. DYNAMICS may be purged only if no dynamics-related set selections have been made in the Case Control Deck.

4.134.4.2 SØF Items

```
EQSS    - Substructure grid point equivalence table  
LØDS    - Load sets identification table  
PVEC    - Load vectors, g set
```

MODULE FUNCTIONAL DESCRIPTIONS

Notes:

1. EQSS must exist.
2. LØDS and PVEC must exist if any LØADC cards are referenced in Case Control.

4.134.5 Output Data

4.134.5.1 GINØ Data Blocks

CASESS - Substructure Case Control Table
CASEI - Case Control Data Table
GPL - Grid Point List for SØLVE operation
EQEXIN - Grid point equivalence table for SØLVE operation
GPDT - Grid Point Definition Table for SØLVE operation
BGPDT - Basic Grid Point Definition Table for SØLVE operation
SIL - Scalar Index List for SØLVE operation
GE3S - GEØM3 data block for SØLVE operation
GE4S - GEØM4 data block for SØLVE operation
DYNs - DYNAMICS data block for SØLVE operation

Notes:

1. CASESS, CASEI, GPL, EQEXIN, GPDT, BGPDT, and SIL may not be purged.
2. GE3S may be purged if no loads are defined.
3. GE4S may be purged only if GEØM4 is also purged.
4. DYNs may be purged only if DYNAMICS is also purged.

4.134.5.2 SØF Items

None

4.134.6 Parameters

DRY - Input, integer. Dry run parameter. If DRY < 0, the PVEC item is allowed to pseudo-exist only.

NAMESØLS - Input, BCD. Name of the substructure to be solved.

LUSET - Output, integer. Number of degrees of freedom, g set.

NØGPDT - Output, integer. Number of degrees of freedom, g set.

FUNCTIONAL MODULE SGEN (SUBSTRUCTURE TABLE GENERATOR)

4.134.7 Method

This module will process constraints, loads and dynamics data specified by substructure name in Phase 2, build pseudostructure geometry tables, and separate the substructure command data and the usual case control data.

An equivalence table between substructure grid points and internal scalar indices is built in open core as follows:

- a. Read the entire EQSS item for the substructure from the SØF into core. A table is built containing basic substructure names and pointers to the data in core.
- b. The I_p value for each entry in the EQSS is replaced by its SIL value. (The position of the SIL value in the SIL group equals the I_p value.) The table now resides in core for the following processes.

1. GEØM4 Processing: MPCs, SPCS, SPCS1, and SPCSD bulk data cards are converted to reflect the internal scalar points and the conversions are stored on a scratch file. The specifications for these conversions are shown below.

MPCS - MPC Conversion

In: $SID, SS_1, G_1, C_1, F_1, SS_2, G_2, C_2, F_2, \dots$

Out: $SID, SIL_1, 0, F_1, SIL_2, 0, F_2, \dots$

where SS_i is a basic substructure name

G_i is an external grid ID

C_i is a degree of freedom component

F_i is a factor

SIL_i is a scalar index

SPCS - SPC Conversion (done by subroutine SGENA)

In: $SID, SS, G_1, C_1, G_2, C_2, \dots$

Out: $SID, SIL_1, 0, 0, SID, SIL_2, 0, 0, \dots$

where C_i is a degree of freedom component code (e.g., 126)

MODULE FUNCTIONAL DESCRIPTIONS

SPCS1 - SPC1 Conversion (done by subroutine SGENB)

In: SID,SS,C,G₁,G₂,G₃,...

Out: SID,0,SIL₁,-1,SID,0,SIL₂,-1,...

SPCSD - SPCD Conversion (done by subroutine SGENA)

In: SID,SS,G₁,C₁,Y₁,G₂,C₂,Y₂

Out: SID,SIL₁,0,Y₁,SID,SIL₂,0,Y₂

where Y_i is the enforced displacement

2. DYNAMICS Processing: DAREAS, DELAYS, DPHASES, and TICS bulk data cards are converted to reflect the internal scalar points and the conversions are stored on a second scratch file. The specifications for these conversions are shown below.

DAREAS - DAREA Conversion (done by subroutine SGENA)

In: SID,SS,G₁,C₁,A₁,G₂,C₂,A₂,...

Out: SID,SIL₁,0,A₁,SID,SIL₂,0,A₂,...

where A_i is an area

DELAYS - DELAY Conversion (done by subroutine SGENA)

In: SID,SS,G₁,C₁,T₁,G₂,C₂,T₂,...

Out: SID,SIL₁,0,T₁,SID,SIL₂,0,T₂,...

where T_i is a time delay value

DPHASES - DPHASE Conversion (done by subroutine SGENA)

In: SID,SS,G₁,C₁,θ₁,G₂,C₂,θ₂,...

Out: SID,SIL₁,0,θ₁,SID,SIL₂,0,θ₂,...

where θ_i is a phase angle

TICS - TIC Conversion (done by subroutine SGENA)

In: SID,SS,G₁,C₁,U₁,V₁,G₂,C₂,U₂,V₂,...

Out: SID,SIL₁,0,U₁,V₁,SID,SIL₂,0,U₂,V₂,...

where U_i is an initial displacement value

V_i is an initial velocity value

FUNCTIONAL MODULE SGEM (SUBSTRUCTURE TABLE GENERATOR)

3. Merging Process: Subroutine SGEM is called to merge the converted GEOM4 data with the already existing GEOM4 data. A fatal error occurs if a data type exists in both its substructure and regular NASTRAN format (e.g. MPCs and MPC). The same process is then repeated for the DYNAMICS data.
4. GEOM3 Processing: LOAD and SLOAD cards on GEOM3 are the only card types on this data block which are used. Non-zero terms of load vectors selected by the user are extracted from the PVEC item, converted to SLOAD card data, merged with SLOAD cards from GEOM3, and written on GE3S. The sequence of operations is as follows:
 - a. The LOADS item is read into open core.
 - b. LOADC cards are read from GEOM4, converted, and written on a scratch file as shown below:
$$\begin{aligned} \text{In: } & \text{SID}, A_0, SS_1, ID_1, A_1, SS_2, ID_2, A_2, \dots \\ \text{Out: } & \text{SID}, A_0, J_1, A_1, J_2, A_2, \dots \end{aligned}$$

where SID is the set ID of the LOADC card which may be referenced on a LOAD case control card; A_i are scale factors; SS_i are basic substructure names; ID_i are load set ID's corresponding to those in the LOADS item; and J_i are the column numbers of the indicated load vectors in the PVEC item.
 - c. LOAD cards are copied from GEOM3 to GE3S, GEOM3 is then positioned to the SLOAD record, if any SLOAD cards exist.
 - d. The PVEC item is copied to a scratch file where it can be read with UNPACK.
 - e. Each logical LOADC card is read from the scratch file and a linear combination of vectors from the PVEC item is computed in open core.
 - f. Each non-zero term of the combined vector is written as an SLOAD card on GE3S.
 - g. SLOAD cards from GEOM3 are merged with those created in steps 'e' and 'f' and written on GE3S such that all set ID's are in sort.

MODULE FUNCTIONAL DESCRIPTIONS

3. Case Control Processing: The CASECC data block is copied to CASESS. The standard case control records (subcase definitions) only are copied to CASEI.

4. Geometry Table Processing: Data blocks GPL, EQEXIN, GPDT, BGPDT, and SIL are written to define a pseudostructure consisting of N scalar points where

$$\text{External ID} = \text{Internal ID} = \text{SIL} = I, \quad I = 1, 2, 3, \dots, N.$$

The coordinate system ID's are -1 and the X, Y, Z values are zero in the GPDT and BGPDT data blocks.

4.134.8 Subroutines

4.134.8.1 Subroutine Name: SPLT10

1. Entry Point: SPLT10

2. Purpose: To decode degree of freedom codes stored as digits.

3. Calling Sequence: CALL SPLT10 (CØDE, CØMPS, NCØMP)

CØDE - Component code stored as digits (e.g., 345). Integer, input.

CØMPS - Array of length 9 where the value of each digit is stored. Integer, output.

NCØMP - The number of values stored in CØMPS.

4. Method: Modulo arithmetic is used to successively strip the low-order digit from CØDE and store it in CØMPS. Zero digits are not stored. When all digits have been processed, CØMPS is sorted and duplicates are removed.

4.134.8.2 Subroutine Name: SGENA

1. Entry Point: SGENA

2. Purpose: To convert bulk data cards with substructure name identification to scalar non-substructure equivalents.

3. Calling Sequence: CALL SGENA (TYPE, BUF, MCB, IFILE, ICØDE, IEXTRA, ØFILE, ØCØDE, ØEXTRA)

TYPE - BCD card name.

BUF - GINØ buffer for input file.

MCB - Matrix control block for input file.

FUNCTIONAL MODULE SGEN (SUBSTRUCTURE TABLE GENERATOR)

IFILE - Input file name.
ICODE - Locate code for input card type.
IEXTRA - Number of extra words (after grid) to be read.
OFILE - Output file name.
OCODE - Locate code for output card type.
OEXTRA - Number of extra words (after grid) to be written.

4.134.8.3 Subroutine Name: SGENB

1. Entry Point: SGENB
2. Purpose: To convert bulk data cards with substructure name identification to scalar non-substructure equivalents. Differs from SGENA in how it formats output cards (SPC1 type format).
3. Calling Sequence: CALL SGENB (TYPE,BUF,MCB,IFILE,ICODE,IEXTRA,OFILE,OCODE,OEXTRA)
Parameters as defined in Section 4.134.8.2

4.134.8.4 Subroutine Name: SGENM

1. Entry Point: SGENM
2. Purpose: To merge converted substructuring data with existing NASTRAN data.
3. Calling Sequence: CALL SGENM (NTYPE,IFILE,SFILE,OFILE,ICODE,OCODE,CTYPES,CTYPEO)
NTYPE - Number of different substructuring cards.
IFILE - Input file name.
SFILE - Scratch file name.
OFILE - Output file name.
ICODE - Locate codes for input card types.
OCODE - Locate codes for output card types.
CTYPES - BCD names of substructuring cards.
CTYPEO - BCD names of corresponding NASTRAN cards.

MODULE FUNCTIONAL DESCRIPTIONS

4.134.9 Design Requirements

SGEN creates data blocks which are representative of a math model consisting wholly of scalar points. If any bulk data cards such as ASET, MPC, OMIT, SPC, or SUPORT are used, they must reference the SIL values (or degree of freedom numbers) output by SGEN. If these values are not known, the user may use the SØFPRINT command to print the EQSS item of the substructure for which a solution is desired. This will print the equivalence table between external grid point ID's, internal point numbers, and SIL values.

4.134.10 Diagnostic Messages

SGEN may issue the following messages:

3001,3002,3003,3008,3037

6022.

6101,6102,6103,6106,6107,

6328,6329,6330,6331,6332

FUNCTIONAL MODULE SØFI (SØF INTO GINØ MATRIX COPIER)

4.135 FUNCTIONAL MODULE SØFI (SØF INTO GINØ MATRIX COPIER)

4.135.1 Entry Point: SØFI

4.135.2 Purpose

This module copies selected matrix items from the SØF into NASTRAN matrix data blocks.

4.135.3 DMAP Calling Sequence

SØFI /A,B,C,D,E/V,N,DRY/C,N,NAME/C,N,IA/C,N,IB/C,N,IC/C,N,ID/C,N,IE \$

4.135.4 Output Data Blocks

A,B,C,D,E - NASTRAN data block corresponding to parameters IA, IB, etc.

4.135.5 Parameters

DRY - Input and output - integer (Default = -1). If DRY = -1 on input, only the existence of the SØF file is checked. If DRY > -1 on input, the copying of data takes place. If a serious error occurs, the parameter is set to -2.

NAME - Input - BCD (No default). Name of substructure.

IA,IB, etc. - Input - BCD (Default = blank). Item names of data blocks A, B, etc. Allowable values are: KMTX, MMTX, PVEC, PØVE, UPRT, HØRG, UVEC, QVEC.

4.135.6 Method

Subroutine SØFI is the main control for the module. The existence of each SØF item is first checked. If a serious error occurs, DRY is set to -2, a warning message describing the error is issued, and all further operations continue if possible. If DRY > -1, the SØF data is copied into NASTRAN data blocks by a call to the SØF utility subroutine MTRXI.

FUNCTIONAL MODULE SØFØ (SØF OUT FROM GINØ MATRIX COPIER)

4.136 FUNCTIONAL MODULE SØFØ (SØF OUT FROM GINØ MATRIX COPIER)

4.136.1 Entry Point: SØFØ

4.136.2 Purpose

This module is used to transfer NASTRAN data blocks to the SØF file for purposes of saving the data for subsequent runs or subsequent execution steps.

4.136.3 DMAP Calling Sequence

SØFØ A,B,C,D,E//V,N,DRY/C,N,NAME/C,N,IA/C,N,IB/C,N,IC/C,N,ID/C,N,IE \$

4.136.4 Input Data Blocks

A,B,C,D,E - NASTRAN data blocks to be written on the SØF. Note: Any or all data blocks may be purged.

4.136.5 Parameters

DRY - Input and output - integer (Default = -1). If DRY = -1 on input, no matrices are copied to the SØF. If DRY > -1 on input, the matrices are copied to the SØF. If a serious error occurs, the parameter is set to -2.

NAME - Input - BCD (No default). Name of substructure.

IA,IB, etc. - Input - BCD. Names of the SØF items to be generated corresponding to data blocks A, B, etc. Allowable item names are: KMTX, MMTX, PVEC, PØVE, UPRT, HØRG, UVEC, QVEC.

4.136.6 Method

Subroutine SØFØ is the main control for the module. The existence of each item's data on the SØF is checked. An item that already exists on the SØF will not be written over. In order to rewrite it, the old item should be deleted before the new one is written. The NASTRAN data blocks are transferred to the SØF by a call to the SØF utility subroutine MTRXØ. If a serious error occurs, DRY is set to -2, a warning message describing the error is issued, and all further operations continue if possible.

FUNCTIONAL MODULE SØFUT (SØF UTILITY MODULE)

4.137 FUNCTIONAL MODULE SØFUT (SØF UTILITY MODULE)

4.137.1 Entry Point: SØFUT

4.137.2 Purpose

This module performs the tasks of altering the SØF file for the purpose of editing, purging and equivalencing the data items of selected substructures. It also prints item and the SØF table of contents.

4.137.3 DMAP Calling Sequence

SØFUT //V,N,DRY/C,N,NAME1/C,N,ØPER/C,N,ØPT/C,N,NAME2/C,N,PREFX/C,N,IA/C,N,IB/C,N,IC/
 C,N,ID/C,N,IE \$

4.137.4 Parameters

DRY - Output - integer. If a serious error occurs, the parameter is set to -2.

NAME - Input - BCD (No default). Name of substructure to be operated on or equivalenced.

ØPER - Input - BCD (No default). Operation flag, values allowed are: EDIT, DESTRØY, EQUIV, SØFPRINT, DELETE, or RENAME.

ØPT - Input - integer (Default = 0). Option code.

EDIT Value is the sum of the following integers reflecting the type of data to be removed.

- 1 = Stiffness Matrices (item KMTX)
- 2 = Mass Matrices (item MMTX)
- 4 = Load Data (items LØDS, PVEC)
- 8 = Solution Data (items UVEC, QVEC, SØLN)
- 16 = Transformation Data (items HØRG, UPRT, PØVE)
- 32 = All data for structure (all items of structure)

SØFPRINT Indicates print option for SØF table of contents, SØF data items, or both.

- > 0 Print data items only
- = 0 Print table of contents only
- < 0 Print both

MODULE FUNCTIONAL DESCRIPTIONS

NAME2 - Input - BCD (Default = blank). Name of equivalent structure (new structure) in EQUIV operation. New name of substructure NAME in RENAME operation.

PREFX - Input - single BCD character (Default = blank). The names of all substructures contributing to substructure NAME1 are equivalenced to a new set having the PREFX value appended to the names.

IA,IB,IC, - Input - BCD (Default = blank). Names of the items to be printed for the ID,IE SØFPRINT operation, or deleted in the DELETE operation.

4.137.5 Method

Subroutine SØFUT is the main control for the module. It initially calls to the SØF utility subroutines which perform the requested operations.

EDIT Operation: In the EDIT operation, only selected items of the selected substructure are removed from the SØF file. The ØPT parameter determines the SØF items to be removed for substructure NAME.

DESTRØY Operation: The DESTRØY operation is similar to the EDIT operation except that all items are removed from the SØF for the structure and all subsequent level structures. The "next level" substructure is removed and its "next level" is also removed, etc. See documentation for subroutine DSTRØY, Section 3.6.8.

EQUIV Operation: The EQUIV operation causes a new set of substructures from an existing substructure updating the MDI and copying the EQSS, LØDS, and PLTS items with modifications. The operations are:

1. A list of substructure names contributing to substructure NAME1 is built. This is done by finding all substructures whose next level is NAME1. The process repeats for these substructures, finding substructures which have their next level in the list of names.
2. A parallel list of "new" names is built, having new names defined by adding a prefix to the old names. If a new name already exists in the MDI list and DRY ≠ 0 an error exists. If DRY = 0 the new names must exist.
3. A new set of MDI pointer tables is built for the "new" contributing substructures. The pointer data is exactly the same as for their equivalent substructure for the following items: EQSS, BGSS, CSTM, LØDS, PLTS, UPRT, HØRG, KMTX, MMTX, PVEC, PØVE (The UVEC and QVEC and SØLN data is flagged as undefined).

FUNCTIONAL MODULE SØFUT (SØF UTILITY MODULE)

The items for substructure NAME2, on the other hand, are all flagged as undefined except the following items: EQSS, BGSS, CSTM, LØDS, PLTS, KMTX, MMTX, PVEC. The omitted item flags will be entered into the MDI tables as the "new" NAME2 substructure is used in subsequent combinations or reductions.

SØFPRINT Operation: To print the SØF table of contents, each block of the DIT and MDI are fetched and their contents are formatted and printed. The amount of unused space on the SØF is also printed. To print a data item, each word of the data is examined to determine its type (integer, real, or BCD) and an approximate variable format is created. The data is printed 10 words per line.

DELETE Operation: Subroutine DELETE is used to remove the selected items.

RENAME Operation: The block of the DIT containing substructure NAME is fetched and NAME is replaced by NAME2. The DIT is then marked as updated.

FUNCTIONAL MODULE SUBPH1 (SUBSTRUCTURE PHASE 1)

4.138 FUNCTIONAL MODULE SUBPH1 (SUBSTRUCTURE, PHASE 1)

4.138.1 Entry Point: SUBPH1

4.138.2 Purpose

To convert data blocks to substructure items.

4.138.3 DMAP Calling Sequence

SUBPH1 CASECC,EQEXIN,USET,BGPDT,CSTM,GPSETS,ELSETS//V,N,DRY/C,N,NAME/C,N,PSET \$

4.138.4 Input Data

4.138.4.1 GINØ Data Blocks

CASECC - Case Control Data Block

EQEXIN - Equivalence tables between external grid numbers and internal point numbers

USET - Displacement set definition table

BGPDT - Basic grid point definition table

CSTM - Coordinate system transformation matrices

GPSETS - Grid point sets related to the element plot sets

ELSETS - Element plot set connection tables

Notes:

1. CASECC, EQEXIN, USET, and BGPDT may not be purged.
2. CSTM may be purged if no coordinate systems have been defined.
3. GPSETS and ELSETS may be purged if no plots are desired in Phase 2.

4.138.4.2 SØF Items

None

4.138.5 Output Data

4.138.5.1 GINØ Data Blocks

None -

4.138.5.2 SØF Items

EQSS - Substructure equivalence table

BGSS - Substructure basic grid point table

CSTM - Coordinate system transformation matrices

MODULE FUNCTIONAL DESCRIPTIONS

PLTS - Plot data for substructures

LØDS - Load set data

4.138.6 Parameters

DRY - Output, integer. If a serious error occurs, DRY = 2.

NAME - Input, BCD. Name of Phase 1 substructure.

PSET - Input, integer. Plot set ID for Phase 2 plots.

4.138.7 Method

Five basic items on the substructure SØF file are created by the module. The five tasks are described separately below:

1. EQSS Generation: The EQSS data item will contain three groups. The first group is the header, containing the substructure name given by the parameter. The second group contains a table of one entry per grid point retained in the structure, sequenced by grid point identification numbers. Each entry contains: (1) the external grid point number G_i , (2) the internal point number, I_p , (i.e., the index to the sequenced array of grid points; if no resequencing has occurred, the values will be 1, 2, 3, ..., N), and (3) the component code word C with each component degree of freedom corresponding to a bit position; if the component is retained, the bit is "on".

The calculation steps are:

- a. The USET data block is read into core. Each word corresponds to a degree of freedom in the U_g vector. The words are coded to indicate the displacement sets to which the degree of freedom belongs.
- b. Record 2 of the EQEXIN data block is read and decoded into two words (G_i and SIL_i) at a time. SIL_i is the pointer to USET entry corresponding to the first degree of freedom of the point. The USET entries corresponding to the grid point are tested. Each degree of freedom belonging to the U_a set turns a bit "on" in the C_i word. G_i and C_i are written on the scratch file if $C_i \neq 0$.
- c. An area of core is set to zero and the scratch file is opened to read the data generated above. Two words, G_i and C_i , are read. Record 1 of the EQEXIN is read two words at a time (G_j and I_g) until a match is obtained on the G values. (I_g corresponds to the original grid point index.) The C_i value is stored in core in

FUNCTIONAL MODULE SUBPH1 (SUBSTRUCTURE PHASE 1)

position $2 * I_g$. The process continues until all points on the scratch file have been read.

d. The running count of the nonzero C_i terms, I_p , is stored in the odd position.

An example of the data stored in core is:

<u>I_g (Implied)</u>	<u>I_p</u>	<u>C</u>
1	1	0111
2	0	0
3	0	0
4	2	11011
5	3	0111
6	0	0
7	4	0001

etc.

The locations correspond to the original I_g numbers; the values I_p and C are the desired output quantities except that the desired sort sequence is by grid numbers.

e. After the header group of the EQSS item is written, the EQEXIN (Record 1) is read again, two words (G_i and I_g) at a time. If $I_p(I_g) \neq 0$, G_i , I_p , and C are written on the EQSS item.

f. The SIL group of the EQSS item is created by counting the number of bits turned on in the previous C values, i.e.,

$$SIL_{i+1} = SIL_i + N_{C_i}$$

where N_{C_i} is the number of bits turned on in word C_i . Only points with $C \neq 0$ are output. In the example above, the SIL table will contain the values 1, 4, 8, 11, 12, etc.

2. **BGSS Generation:** The BGSS data item is created by copying the entries in the BGPDT data block corresponding to the new points. Each entry is read and the corresponding position in core is checked to determine if a nonzero C value exists. If not, the entry is skipped. If $C \neq 0$, the four words CID, X, Y, and Z are copied to the BGSS, and the process continues to the next grid point.

MODULE FUNCTIONAL DESCRIPTIONS

3. CSTM Generation: The CSTM data on the SØF file is identical to the format in the NASTRAN data block.

4. LØDS Generation: The LØDS data item contains the load set identifiers for static loads. The CASECC data block contains these set ID values. Each subcase in the CASECC data block corresponds to one load vector. The substructure load set ID is defined as follows:

- a. Word 4 (Static Load Set) is selected first.
- b. If Word 4 is zero, Word 8 (Thermal Load Set) is used.
- c. If both Word 4 and Word 8 are zero, Word 6 (Element Deformations) is used.
- d. If Words 4, 6, and 8 are all zero, zero is stored in LØDS for that subcase.
- e. A user message is issued describing the substructure load set ID for each subcase with loads.
- f. If no loads have been defined for any subcase, the LØDS item is not generated.

5. PLTS Generation: The PLTS item is a copy of all or part of data blocks BGPDT, EQEXIN, GPSET, and ELSETS. The BGPDT data block is copied in its entirety to the PLTS item. Record 1 of EQEXIN is then copied. The data for the plot set indicated by the PSET parameter is copied to the PLTS item from GPSETS and ELSETS.

4.138.8 Subroutines

None

4.138.9 Diagnostic Messages

SUBPH1 may issue the following messages:

6012-6014, 6050, 6325-6327, and 6361

FUNCTIONAL MODULE PLTMRG (SUBSTRUCTURE PLOT SET DATA MERGE)

4.139 FUNCTIONAL MODULE PLTMRG (SUBSTRUCTURE PLOT SET DATA MERGE)

4.139.1 Entry Point: PLTMRG

4.139.2 Purpose

To generate data blocks necessary to produce undeformed substructure plots.

4.139.3 DMAP Calling Sequence

PLTMRG CASESS,PCDB/PLTP,GPS,ELS,BGP,CASEP,EQEX/C,N,SSNM/V,N,NGP/V,N,LSIL/V,N,NPSET \$

4.139.4 Input Data

4.139.4.1 GINØ Data Blocks

CASESS - Case Control for substructures

PCDB - Plot Control Data Block

Note: No input data block may be purged.

4.139.4.2 SØF Item

EQSS - Substructure equivalence tables

PLTS - Plot data

Notes:

1. The EQSS and PLTS items of the substructure to be plotted must exist.
2. The PLTS items of the basic substructures which comprise the substructure to be plotted should all exist. Otherwise, portions of the substructure may not be plotted.

4.139.5 Output Data

4.139.5.1 GINØ Data Blocks

PLTP - Plot parameters and plot control table

GPS - Grid point sets related to the element plot sets

ELS - Element plot set connection tables

BGP - Basic grid point coordinates

CASEP - Case Control data for plots

EQEX - Equivalence between external grid or scalar numbers and internal numbers for plots.

Note: No output data block may be purged.

MODULE FUNCTIONAL DESCRIPTIONS

4.139.5.2 SØF Items

None

4.139.6 Parameters

SSNM - BCD, input, no default. Name of the substructure to be plotted.

NGP - Integer, output, no default. Total number of structural grid points.

LSIL - Integer, output, no default. Last scalar index value.

NPSET - Integer, output, no default. Number of plot sets. Set to -1 if an error occurs.

4.139.7 Method

PLTMRG consists of a single subroutine which computes each output data block in turn.

CASEP. CASESS is examined record by record until the CASECC header record is found. This record and all subsequent records are copied to CASEP.

BGP. The basic substructure transformation data is read from the PLTS item of the substructure to be plotted (SSNM). For each component basic substructure (CBSS), the basic grid point data is read from its PLTS item, transformed to the basic coordinate system of SSNM and written on BGP. During this processing, the parameter NGP is computed.

EQEX. For each CBSS, the external/internal equivalence data is read into open core from its PLTS item. The internal ID numbers are incremented by the number of grid points on the CBSS's read previously (during this step). Each point is flagged with its substructure number. The combined equivalence data for CBSS's is then sorted on external ID's and written on SCR1. The external and internal ID's are written as the first logical record of EQEX.

To compute record 2 of EQEX, the EQSS item of SSNM is first read into open core. The data on SCR1 is then processed one point at a time. The external ID is used to locate the SIL number in the EQSS data. If it cannot be found, the SIL number is set to -1. The external ID and SIL number are then written on EQEX.

LSIL is set to the highest SIL number in EQSS.

PLTP. Only one plot set is allowed and it must have been defined in Phase 1. Therefore, PCDB is simply copied to PLTP. NPSET is set equal to one.

GPS. Record 1 of GPS is written. It consists of the single plot set ID of 1. The number of grid points in the element sets is read from the GPSETS data of the PLTS items of each CBSS, summed, and written as the first word of the second record of GPS.

FUNCTIONAL MODULE PLTMRG (SUBSTRUCTURE PLOT SET DATA MERGE)

For each CBSS, the GPSETS data is read from its PLTS item into open core. The pointers in this data are incremented by the total number of grid points in the element sets of the CBSS's read previously (during this step). The result is written on GPS.

ELS. For each CBSS, the ELSETS data is read from its PLTS item, one element at a time. Each grid point connection index is incremented by the number of structural grid points on the CBSS's read previously (during this step). The result is written on ELS.

4.139.8 Diagnostic Messages

PLTMRG does not generate any fatal messages. If an error which would prevent the generation of plot occurs, NPSET is set to -1.

The following warning messages may be issued by PLTMRG:

3001,3002,3003,3008

6101,6102,6103,6106,6107

UTILITY MODULE TIMETEST

4.140 UTILITY MODULE TIMETEST

4.140.1 Entry Point

TIMTST

4.140.2 Purpose

TIMTST provides timing data for various NASTRAN unit operations which may be used to make comparisons between computers or to evaluate compilers.

4.140.3 DMAP Calling Sequence

TIMETEST / , / C,Y,N=n / C,Y,M=m / C,N,t / C,Y,Ø1 / C,Y,Ø2 \$

4.140.4 Input Data Blocks

None.

4.140.5 Output Data Blocks

Not presently used.

4.140.6 Parameters

n* - External Loop Index

m* - Internal Loop Index

t - Data Item Type (1=RSP, 2=RDP, 3=CSP, 4=CDP)

Ø1 - Type of timing data required

Ø2 - Code indicating which I/Ø unit operations are to be tested.
(Ø1 = 1 only)

*Total unit operations = n*m

4.140.7 Method

TIMTST examines the value of Ø1 of the DMAP statement and calls the corresponding TIMTSx routine to process the user request for timing data, where x is the value of Ø1 (values 1 thru 8 are allowed).

MODULE FUNCTIONAL DESCRIPTIONS

4.140.8 Subroutines

4.140.8.1 Subroutine Name: TIMTS1

1. Entry Point: TIMTS1
2. Purpose: To provide timing data for GINØ and the PACK routines.
3. Calling Sequence:

CALL TIMTS1

COMMON / / N, M, TYPE, ØPT1, ØPT2

Communication area for TIMTST

COMMON / TIM1XX / A(1)

Open core area

4. Method: TIMTS1 tests ØPT2 and performs I/O tasks according to the following table:

ØPT2 =	1 - Perform standard GINØ WRITE Operations
	2 - Perform standard GINØ READ Operations
	4 - Perform standard GINØ READ Backwards Operations
	8 - Perform standard BLDPK Operations
	16 - Perform standard INTPK Operations
	32 - Perform standard PACK Operations
	64 - Perform standard UNPACK Operations
	128 - Perform standard PUTSTR Operations
	256 - Perform standard GETSTR Operations

The operations performed by this routine are performed n*m times in the type as defined by the TYPE parameter. Task timing is obtained through the utility subroutine SECØND, and the results are output on the System Output File.

Notes:

- a. Combinations of the above operations may be requested by adding the values together.
- b. Certain operations cannot be requested without requesting a previous operation, i.e., GINØ READ operations (ØPT2=2) cannot be performed without the corresponding GINØ operations (ØPT2=1).

4.140.8.2 Subroutine Name: TIMTS2

1. Entry Point: TIMTS2
2. Purpose: To provide timing data for typical loops within the NASTRAN program logic.

3. Calling Sequence:

CALL TIMTS2

COMMON / / N, M, TYPE, OPT1, OPT2

Communication area for TIMTST

COMMON / TIM2XX / A(1)

Open Core area.

4. Method: TIMTS2 performs typical looping operations n^m times in the type defined by the TYPE parameter. Standard loops are coded in FORTRAN and in a manner to simulate tight, medium, and loosely defined loops. Task timing is obtained through the utility subroutine SECOND, and the results are output on the System Output File.

4.140.8.3 Subroutine Name: TIMTS3

1. Entry Point: TIMTS3
2. Purpose: TIMTS3 is identical in function to TIMTS2 with the exception that all arithmetic operations are performed in the Assembler Language of the operation computer.

4.140.8.4 Subroutine Names: TIMTS4 and TIMTS5

1. Entry Point: TIMTS4 - TIMTS5
2. Purpose: Both TIMTS4 and TIMTS5 are dummy decks complete with open core and are supplied to permit user definition of standard operations for which timing would be required.

4.140.8.5 Subroutine Name: TIMTS6

1. Entry Point: TIMTS6
2. Purpose: Identical to TIMTS2 except compiled with the FORTRAN G level compiler on the IBM 360.

MODULE FUNCTIONAL DESCRIPTIONS

4.140.8.6 Subroutine Name: TIMTS7

1. Entry Point: TIMTS7
2. Purpose: Identical to TIMTS2 except compiled with the FØRTRAN H, ØPT=0 compiler on the IBM 360.

4.140.8.7 Subroutine Name: TIMTS8

1. Entry Point: TIMTS8
2. Purpose: Identical to TIMTS2 except compiled with the FØRTRAN H, ØPT=1 compiler on the IBM 360.

4.140.9 Design Requirements

Each functional subroutine has defined within it an open core area for it to use during its function. The only communication between routines is accomplished through blank common.

4.140.10 Diagnostic Messages

Apart from the desired timing information the following error messages may be generated: 2195, 2196, and 2197.

4.141 FUNCTIONAL MODULE DDRMM (DYNAMIC DATA RECOVERY - MATRIX METHOD)

4.141.1 Entry Point: DDRMM

4.141.2 Purpose

1. To arrive at the user output requests for transient or frequency point displacements, point velocities, point accelerations, forces of single-point constraint, element stresses, and element forces.
2. To format the above output requests into data blocks for direct output by the Output File Processor (ØFP) module.

4.141.3 DMAP Calling Sequences

Transient Response (SØRT2 inputs and outputs)

DDRMM,CASEXX,UHVT,TOL,IPHIP2,IQP2,IES2,IEF2,EST,MPT,DIT/ZUPV2,ZQP2,ZES2,ZEF2,\$

Frequency Response (SØRT1 inputs and outputs)

DDRMM,CASEXX,UHVF,PPF,IPHIP1,IQP1,IES1,IEF1,/ZUPVC1,ZQPC1,ZESC1,ZEFC1,\$

Frequency Response (SØRT2 inputs and outputs)

DDRMM,CASEXX,UHVF,PPF,IPHIP2,IQP2,IES2,IEF2,/ZUPVC2,ZQPC2,ZESC2,ZEFC2,\$

4.141.4 Input Data Blocks

CASEXX - Case Control data table for dynamics problems.

UHVF - Modal frequency response solution vectors - h set.

UHVT - Modal transient response solution vectors - h set.

PPF - Header record supplies frequency list.

TØL - Table of transient time step list.

IPHIP1 - Modal displacements (p set, SØRT1).

IPHIP2 - Modal displacements (p set, SØRT2).

IQP1 - Modal forces of single-point constraint (p set, SØRT1).

IQP2 - Modal forces of single-point constraint (p set, SØRT2).

IES1 - Modal element stresses (SØRT1).

MODULE FUNCTIONAL DESCRIPTIONS

- IES2 - Modal element stresses (SØRT2).
- IEF1 - Modal element forces (SØRT1).
- IEF2 - Modal element forces (SØRT2).
- EST - Element summary table.
- MPT - Material properties table
- DIT - Direct input tables.

4.141.5 Output Data Blocks

- ZUPV2 - Output transient response displacement vector requests (p set, SØRT2).
- ZUPVC1 - Output frequency response displacement vector requests (p set, SØRT1, complex).
- ZUPVC2 - Output frequency response displacement vector requests (p set, SØRT2, complex).
- ZQP2 - Output transient response forces of single-point constraint requests (p set, SØRT2).
- ZQPC1 - Output frequency response forces of single-point constraint requests (p set, SØRT1, complex).
- ZQPC2 - Output frequency response forces of single-point constraint requests (p set, SØRT2, complex).
- ZES2 - Output transient response element stress requests (SØRT2).
- ZESC1 - Output frequency response element stress requests (SØRT1, complex).
- ZESC2 - Output frequency response element stress requests (SØRT2, complex).
- ZEf2 - Output transient response element force requests (SØRT2).
- ZEFC1 - Output frequency response element force requests (SØRT1, complex).
- ZEFC2 - Output frequency response element force requests (SØRT2, complex).

4.141.6 Parameters

None.

4.141.7 Method

A matrix of modal solution data is formed from each of the modal SDR solution data blocks input to the module. These data matrices and the matrix of solution vectors are multiplied via the MPYAD subroutine to form an output matrix.

If the modal solutions input to the module are SØRT1, the equation is:

$$[\text{Output Matrix}]^1 = [\text{Data Matrix}]^1 \times [\text{Solution Matrix}]$$

FUNCTIONAL MODULE DDRMM (DYNAMIC DATA RECOVERY - MATRIX METHOD)

If the modal solutions input to the module are SØRT2, the equation is:

$$[\text{Output Matrix}^2] = [\text{Solution Matrix}]^T \times [\text{Data Matrix}^2]$$

The output matrix formed for each modal input data block is converted to output data block format for processing by module ØFP.

Further details follow in the subroutine descriptions.

In the above equations the solution matrix contains one solution vector (column) of size h (= number of eigenvalues used) for each time or frequency step. Thus,

[Solution Matrix] is of size Rows x Columns

where Rows = number of eigenvalues,
and Columns = number of time or frequency steps.

In the above, the data matrix, if the modal data input is SØRT1, contains one column vector for each eigenvalue used. Each column contains all modal output component results for all elements or points requested. If the modal data input is SØRT2, the data matrix will be the transpose of that for SØRT1 input data, this being due to the data processing requirements. Thus,

[Data Matrix¹] is of size Rows x Columns

where Rows = \sum components of all points or elements requested.
Columns = number of eigenvalues.

$$[\text{Data Matrix}^2] = [\text{Data Matrix}^1]^T$$

The output matrix will then, if the modal data input is SØRT1, contain columns (one for each time or frequency step) containing transient response or frequency response solution components in the same order as those found in the data matrix. Thus,

[Output Matrix¹] is of size Rows x Columns

where Rows = \sum components of all points or elements requested.
Columns = number of time or frequency steps being output.

$$[\text{Output Matrix}^2] = [\text{Output Matrix}^1]^T$$

If the modal solutions input to the module are SØRT2, the equation is:

$$[\text{Output Matrix}^2] = [\text{Solution Matrix}]^T \times [\text{Data Matrix}^2]$$

MODULE FUNCTIONAL DESCRIPTIONS

4.141.8 Subroutines

4.141.8.1 Subroutine Name: DDRMM

1. Entry Point: DDRMM
2. Purpose: Main driving routine of the DDRMM module. Allocates open core, GINØ buffers, reads case control into open core, reads list of frequency or time steps into open core.
3. Calling Sequence: CALL DDRMM
4. Method: If frequency response problem, this routine will partition out solution vectors from the solution matrix based upon the output frequency set (ØFREQ) specified in case control.

If transient response problem, this routine will partition into three scratch data blocks the solution matrix with respect to displacements, velocities, and accelerations.

A loop is executed once for each modal input solution data block to arrive at its respective transient response, or frequency response solution data block. In this loop the appropriate data blocks are opened, the major identification code of the input data is determined, the form of the input (SØRT1 or SØRT2) is determined, and data initialization is made. The SØRT1 or SØRT2 processor (DDRMM1 or DDRMM2, respectively) is then called to form and write in ØFP-input-format the solution data block. On return from the SØRT1 or SØRT2 processor, the data blocks involved are closed and the loop is made again for the next modal input solution.

After the loop has been satisfied for all modal input data blocks possible, the module returns control to the executive monitor.

4.141.8.2 Subroutine Name: DDRMM1

1. Entry Point: DDRMM1
2. Purpose: Performs SØRT1 type processing.
3. Calling Sequence: CALL DDRMM1(n_1, n_2, n_3)

n_1 - FØRTRAN statement number defining return taken in the event of a purged data block.

n_2 - FØRTRAN statement number defining return taken in the event an unexpected end-of-file is encountered during a read operation.

n_3 - FØRTRAN statement number defining return taken in the event an unexpected end-of-

FUNCTIONAL MODULE DDRMM (DYNAMIC DATA RECOVERY - MATRIX METHOD)

record is encountered during a read operation.

./DDRMCI/ - Common block used to communicate local storage between DDRMM and DDRMM1.

4. Method: A data matrix (SCRATCH5) is formed from the input modal solution data block being processed on a given call to this routine. Simultaneously an identification mapping file is formed on another data block (SCRATCH4).

The ØFP-input-format modal data are read and interpreted. For each eigenvalue, the corresponding data are collected and a matrix column output to SCRATCH4.

Component data of first ID

Component data of next ID

⋮

Component data of last ID

} Matrix column of data is of same size for each eigenvalue present, as written to the data-matrix (SCRATCH5)

During the output of the component data for the first column of the data matrix, representing the first eigenvalue for which there is modal input data, the grid point or element-identifications are written to SCRATCH4. This is performed only for the first output eigenvalue as the data are in the same order for all eigenvalues present.

Once the data matrix is complete on SCRATCH5, it is multiplied by the frequency solution vectors via the NASTRAN matrix utility subroutine MPYAD. The product matrix on data block SCRATCH6 will have columns of the same size as that of the data matrix (SCRATCH5), and the number of columns will be equal to the number of time or frequency step solution vectors present.

SCRATCH6 is read and an ØFP-input-type data block is written. The component data of the transient or frequency step solution vectors on SCRATCH6 are reunited with identification data of the mapping file data block (SCRATCH4) and written to the ØFP-input-type data block. Each column as processed to the output data block has its appropriate time or frequency step value placed in the output data block too.

In the event of displacements in a transient response problem, three passes of this routine are made, on one call, with respect to displacements, velocities, and accelerations. This is due to the ability of the NASTRAN user to request different output identifications for each of displacements, velocities, and accelerations.

MODULE FUNCTIONAL DESCRIPTIONS

4.141.8.3 Subroutine Name: DDRMM2

1. Entry Point: DDRMM2
2. Purpose: Performs SØRT2 type processing.
3. Calling Sequence: CALL DDRMM2((\$n₁,\$n₂,\$n₃)
n₁ - FØRTRAN statement number defining return taken in the event of a purged data block.
n₂ - FØRTRAN statement number defining return taken in the event an unexpected end-of-file is encountered during a read operation.
n₃ - FØRTRAN statement number defining return taken in the event an unexpected end-of-record is encountered during a read operation.
/DDRMCI/ - Common block used to communicate local storage between DDRMM and DDRMM2.
4. Method: A data matrix (SCRATCH5) is formed from the input modal solution data block being processed on a given call to this routine. Simultaneously an identification mapping file is formed on another data block (SCRATCH4).

The ØFP-input-format modal data are read and interpreted. For each "Major-ID record" present representing one point-ID or element-ID, the following "Data record" contains component data of the point-ID or element-ID for all eigenvalues used. Each component set of data representing all eigenvalues used, becomes a column of the data matrix (SCRATCH5).

As each "Major-ID record" and "Data record" pair is read and processed, its point-ID or element-ID is placed in the mapping data block (SCRATCH4).

Once the data matrix is complete on SCRATCH5, it is used along with the transient or frequency solution vectors to solve for a product matrix via the NASTRAN matrix utility subroutine MPYAD. The product matrix is placed on data block SCRATCH6 by MPYAD and has a number of columns equal to the number of columns in the data matrix (SCRATCH5). The number of rows equals the output number of time or frequency steps.

SCRATCH6 is read and an ØFP-input-type data block is written. All component data with respect to all time or frequency steps are grouped along with point-ID or element-ID data from the mapping data on SCRATCH4 to form the ØFP type "ID-record" - "Data record" pairs of the output data block.

FUNCTIONAL MODULE DDRMM (DYNAMIC DATA RECOVERY - MATRIX METHOD)

4.141.8.4 Subroutine Name: DDRMMA

1. Entry Point: DDRMMA
2. Purpose: To unpack data from the solution product matrix (SCRATCH6)
3. Calling Sequence: CALL DDRMMA(SETUP)

SETUP - A logical variable. To initialize unpacking of a data block, one call is made with SETUP = .TRUE. . Thereafter, for each ØFP-type line entry called for, SETUP is set equal to .FALSE. .

/DDRMCI/ - Common block used for communication of storage within the DDRMM module.

4. Method: N/A

4.141.8.5 Subroutine Name: DDRMMP

1. Entry Point: DDRMMP
2. Purpose: Builds a list in open core of sorted points for which XY plot requests have been made for file type IXYTYP and for subcase 0 and ICASE.
3. Calling Sequence: CALL DDRMMP(\$N1,2,NCORE,IXYTYP,ICASE,BUFF,ANYXY)

N1 - FØRTRAN statement number defining return to be taken if insufficient core is available

Z - Address of open core - integer

NCØRE - Number of words of open core available - integer

LUSED - Length of open core used by DDRMMP - integer

IXYTYP - The type of XY plot being requested - integer

ICASE - Subcase for which list is being built - integer

BUFF - Buffer for opening XYCDB

ANYXY - Flag for XYPLØT output - logical

/DDRMCI/ - Common block used for communication of errors between DDRMMP and DDRMM2.

4. Method: N/A

MODULE FUNCTIONAL DESCRIPTIONS

4.141.9 Design Requirements

Module design and logic requires that in the case of SØRT2 processing, the component data of any one point-ID or element-ID for all output time or frequency steps must fit in core. That is to say that the largest output ØFP-"ID-record" must fit in core. (Example - if a point has six components of data plus two components for specifications of ID and type, and there were 50 time steps, the ØFP-"ID-record" would be $(6+2) \cdot 50 = 600$ words.)

Common blocks /CLSTRS/ and /GPTA1/ as found in block data subprogram GPTABD are required by DDRMM. These give various element dependent data.

Execution requires that with respect to transient response problems, module SDR2 put out modal displacements (eigenvectors) for all points so that DDRMM can produce the various output selections of displacements, velocities, and accelerations which the user may request in case control. SDR2, while producing modal solutions, will recognize that a transient response problem is being solved for overall.

4.141.10 Diagnostic Messages

Messages 301 thru 303, 391 thru 393, 447 and 448 may be issued by this subroutine.

FUNCTIONAL MODULE ØPTR2

4.142 FUNCTIONAL MODULE ØPTR2 (FULLY STRESSED DESIGN - PHASE 2)

4.142.1 Entry Point: ØPTR2

4.142.2 Purpose

To create new property data for elements on ØPTP1 and ØES1 data blocks. Convergence and print control parameters are set.

4.142.3 DMAP Calling Sequence

ØPTR2 ØPTP1,ØES1,EST / ØPTP2,EST1 / V,N,PRINT / V,N,TSTART / V,N,CØUNT \$

4.142.4 Input Data Blocks

ØPTP1 - Property optimization table

ØES1 - Output element stress requests (SØRT1, real)

EST - Element summary table

Note: The ØPTP1 may be purged in which case ØPTR2 returns.
All other data blocks must exist.

4.142.5 Output Data Blocks

ØPTP2 - Property optimization table (updated)

EST1 - Element summary table (updated)

4.142.6 Parameters

PRINT - Print control parameter, input and output, integer.

TSTART - Initial iteration start time, input, integer.

CØUNT - Iteration counter, input and output, integer.

4.142.7 Method

4.142.7.1 Overview of the Method

This module compares the properties for the last iteration and the calculated properties for the current iteration. Upon convergence the module may be requested to punch the new property data cards.

MODULE FUNCTIONAL DESCRIPTIONS

Convergence of stresses is defined by

$$EPS \geq \frac{|\sigma - \sigma_l|}{\sigma_l}, \quad (1)$$

where EPS = value supplied on the PØPT data card.

σ = stresses to optimize.

σ_l = the appropriate stress limit from the material card.

} See Section 4.120.9.4

Convergence of properties is calculated as follows:

$$A1 = \frac{\sigma}{\sigma_l}, \quad (2)$$

$$ALPH = \text{MAX}(A1, ALPH), \quad (3)$$

where ALPH is initially -1.0. Note that A1 will always be positive or zero. This done for all subcases present on ØES1. Prior to calculating the new property, ALPH is set to .0001 if it is zero. This allows for stress redistribution in subsequent iterations.

The new property is calculated by

$$PNEW = PLST \left(\frac{ALPH}{ALPH + (1 - ALPH)GAMA} \right), \quad (4)$$

where PLST = design property on last iteration

GAMA = iteration factor from PØPT bulk data card.

If maximum and/or minimum property limit(s) exist for the property, PNEW/PØRIG must fall between these limit(s), in which case ALPH is recalculated to the appropriate limit, and PNEW is recalculated (the original property is PØRIG).

Convergence of properties occurs if PNEW is not different from PLST for at least one property.

4.142.7.2 Program Method

A maximum number of iterations, MAX, is defined on the PØPT bulk data card. The parameter CØUNT is increased by one with each entry to ØPTR2. If CØUNT is greater than MAX, the module

exits. This allows MAX recalculations of the properties and one additional check iteration in the case where convergence was not achieved.

Parameters CØUNT and TSTART are used to determine if sufficient time exists for one more loop. If not, CØUNT is set to -1 and an exit is taken. If sufficient time exists for only one additional iteration, CØUNT = MAX and processing continues.

The main routine, ØPTR2, loads file ØTP1 into core. Subroutine ØPT2A is then called to read the stress data, determines if stress convergence is achieved, and calculates the property change ratio, ALPH, for each property. Subroutine ØPT2B is called to calculate the new properties and verify that the new property is different than the last property for at least one element property.

If another iteration will occur, file EST1 is created in ØPT2C and ØPT2 is created in subroutine ØPT2D.

The parameter PRINT is set so the first and last iteration will always be printed by ØFP and by ØPTR2. Intermediate iterations are also printed as specified by the bulk data card PØPT.

4.142.8 Subroutines

4.142.8.1 Subroutine Name: ØPT2A

1. Entry Point: ØPT2A
2. Purpose: To read the ØES1 data block, calculate ALPH and determine if stress convergence occurred.

3. Calling Sequence: CALL ØPT2A (PT,EL,IEL,PR,IPR)

COMMON //

COMMON / OPTPW2 /

COMMON / XXOPT2 /

COMMON / NAMES /

COMMON / SYSTEM /

} See Section 4.142.9.3

See Section 2.5.1.8

See Section 2.4.1.8

where

PT = input integer - element type optimization pointer table.

EL = input real - element section of core.

IEL = input integer - the same array as EL.

MODULE FUNCTIONAL DESCRIPTIONS

PR = input real - property section of core.

IPR = input integer - the same array as PR.

4. Method: For each element type on ØES1, it is determined if any elements reside in core of this type. If not, the records are skipped. If there are any, a sequential search is made for each element ID in core. When and if it is found, ALPH is calculated and convergence checked.
5. Additional Subroutines: READ, FREAD, EJECT, MESSAGE

4.142.8.2 Subroutine Name: ØPT2B

1. Entry Point: ØPT2B
2. Purpose: To create the new property values and determine if property convergence occurred.
3. Calling Sequence: CALL OPT2B (IPR,PR,PL)

CØMMØN //	}	See Section 4.142.9.3
CØMMØN / XXØPT2 /		
CØMMØN / SYSTEM /		

See Section 2.4.1.8

where

IPR = input and output integer - property section of core

PR = input and output real - the same array as IPR

PL = input real - property change ratio section of core

4. Method: The ALPH word of the PR array controls this routine. If negative, no properties are changed. If zero, .0001 is substituted with an appropriate warning message (on "print" iterations) and the properties are recalculated. If positive, the properties are recalculated as in Equation 4 and a property change flag is set. If no properties were changed, property convergence has occurred. Parameter CØUNT is set to zero and CØNV = 0.0 as a flag to ØPTPR2.
5. Additional Subroutines: EJECT

4.142.8.3 Subroutine Name: ØPT2C

1. Entry Point: ØPT2C
2. Purpose: To create EST1 data block. For "print" iterations, to print the new property cards. Upon convergence to punch the new property cards is requested.
3. Calling Sequence: CALL ØPT2C (PT,IEL,IPR,PR)

CØMMØN //	}	See Section 4.142.9.3
CØMMØN / ØPTPW2 /		
CØMMØN / XXØPT2 /		
CØMMØN / NAMES /		See Section 2.5.1.8
CØMMØN / SYSTEM /		See Section 2.4.1.8
CØMMØN / GPTA1 /		See Section 2.5.2.1

where

PT = input integer - element type optimization pointer table.

IEL = input integer - element section of core.

IPR = input integer - property section of core.

PR = input real - the same array as IPR.

4. Method: The EST data block element type is compared to elements to be optimized. If the element type is not to be optimized, it is copied across to EST1. If the element type has entries in the PR array, each element not in the IEL array is copied across to EST1 and elements in IEL are updated before copying to EST1 data block.

The printed and punched properties use a "variable format" and are thus built as needed. Real numbers are converted to 2A4 by subroutine RE2AL and 3 to 6 place accuracy with roundoff. Single field bulk data cards are created.

5. Additional Subroutines: EJECT, MESSAGE, READ, WRITE, EOF, ØRF, RSHIFT, LSHIFT, RE2AL

MODULE FUNCTIONAL DESCRIPTIONS

4.142.8.4 Subroutine Name: ØPT2D

1. Entry Point: ØPT2D
2. Purpose: Create ØPTP2 data block.
3. Calling Sequence: CALL ØPT2D (IPR,PR)

COMMON //	}	See Section 4.142.9.3
COMMON / ØPTPW1 /		
COMMON / SYSTEM /		See Section 2.4.1.8
COMMON / NAMES /		See Section 2.5.1.8

where

IPR = input integer - property section of core.

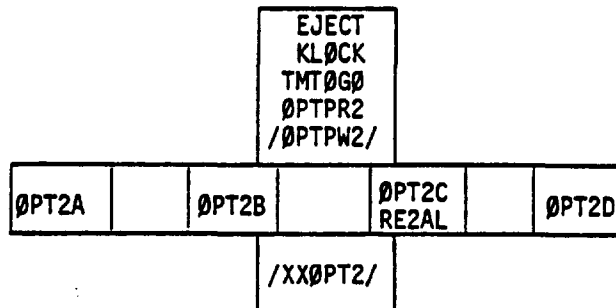
PR = input real - the same array as IPR.

4. Additional Subroutines: READ, FREAD, WRITE, EOF.

4.142.9 Design Requirements

4.142.9.1 Overlay Design

The module in the future may have the following overlay. Normal NASTRAN utility subroutines are not shown.



FUNCTIONAL MODULE ØPTR2

4.142.9.2 Open Core Design (Common Block XXØPT2)

See Section 4.120.9.2 for the contents of each array.

PARM(1)	PØPT(6) - Data from PØPT bulk data Card
IPRNT	Internal print control parameter, integer
Y(1)	ELT(NTYPES) - Element type pointers
Y(PTPTY)	PT(2,NPØW+1) - Element type and element property starting locations (NPØW = 13)
Y(PTELY)	EL(NELW) - Element section of core where NELW is a multiple of 5.
Y(PTPRY)	PR(NPRW) - Property section of core where NPRW is a multiple of 6.
Y(PTPLY)	PL(NKLW) - Property change limits where NKLW is a multiple of 2.

4.142.9.3 Block Data Interface

Value following description is default value.

1. CØMMØN // PRINT,TSTART,CØUNT,NCARD,'SKP',YCØR,B1,NELØP,NWDSE,NWDSP,
ØPTP1,ØES1,EST1,ØPTP2,EST2,NELW,NPRW,NKLW,CØNV

- PRINT - Parameter, print (positive) or not to print (negative), integer.
- TSTART - Parameter, starting time of first iteration loop, integer.
- CØUNT - Parameter, iteration counter. Set negative on last iteration, integer.
- NCARD - Parameter, number of cards punched, integer.
- YCØR - Open core available relative to Y(1), integer.
- B1 - Location of a GINØ buffer relative to PARM(1), integer.
- NELØP - Number of element types that may be optimized, integer.
- NWDSE - Number of words for each element entry, integer (5).
- NWDSP - Number of words for each property entry, integer (6).
- ØPTP1 - GINØ file name of ØPTP1, integer.
- ØES1 - GINØ file name of ØES1, integer.
- EST1 - GINØ file name of EST, integer.
- ØPTP2 - GINØ file name of ØPTP2, integer.

MODULE FUNCTIONAL DESCRIPTIONS

EST2 - GINØ file name of EST1, integer.
NELW - Total number of words in element section of core, integer (0).
NPRW - Total number of words in property section of core, integer (0).
NKLW - Total number of words in property change limit section of core,
integer (0).
CØNV - Convergence parameter, real (0.0).

2. CØMMØN / ØPTPW2 / ZCØR,Z(200)

ZCØR - length of array z, integer (200).
Z - fixed length array that each subroutine may use as scratch space.

3. CØMMØN / XXØPT2 / PARM(6),IPRNT,Y(1)

PARM - parameters from the PØPT bulk data card
IPRNT - internal print flag
Y - open core

4.142.10 Diagnostic Messages

Some messages are written on the output file immediately (numbers 2289, 2295 to 2297, 2302 thru 2304). Others are queued (3001 thru 3003, 3008 and 3061).

4.143 FUNCTIONAL MODULE DIAGØNAL (MATRIX DIAGONAL EXTRACTOR)

4.143.1 Entry Point: DIAGØN

4.143.2 Purpose:

To remove the real part of the diagonal from a matrix, raise each term to a specified power, and output a column vector or square symmetric matrix.

4.143.3 DMAP Calling Sequence

DIAGØNAL A / B / C,Y,ØPT / V,Y,PØWER \$

4.143.4 Input Data Blocks

A - can be any square or diagonal matrix.

4.143.5 Output Data Blocks

B - contains the real diagonal elements of A. It is either a column vector (rectangular) or square (symmetric) matrix, depending on the output option chosen.

4.143.6 Parameters

ØPT - Input-bcd, default = 'CØLUMN'. Output option.
 = 'CØLUMN' produces column vector output labeled as a general rectangular matrix.
 = 'SQUARE' produces square output matrix labeled symmetric.

PØWER - input-real-single precision, default = 1. Exponent to which the real part of each diagonal element is raised.

4.143.7 Method

The input matrix trailer is read and checked for validity. A purged input causes a return with no action. The power parameter is examined and special cases of 0., 0.5, 1.0, and 2.0 are noted. The input data block is then read element by element and the real part of each diagonal term is processed. The processed values are written one at a time into either a column vector or as the diagonal of a square matrix; the choice depending on the input parameter.

4.143.8 Subroutines

None.

MODULE FUNCTIONAL DESCRIPTIONS

4.143.9 Design Requirements

Open core is defined at /DIAGXX/ and must be sufficient to hold two GINØ buffers.

4.143.10 Diagnostic Messages

The following diagnostic messages may occur: 3008, 3016, and 3300.

FUNCTIONAL MODULE SCALAR (MATRIX ELEMENT EXTRACTOR)

4.144 FUNCTIONAL MODULE SCALAR (MATRIX ELEMENT EXTRACTOR)

4.144.1 Entry Point: SCALAR

4.144.2 Purpose

To extract a specified element from a matrix for use as a parameter.

4.144.3 DMAP Calling Sequence

SCALAR A // V,Y,NROW / V,Y,NCØL / V,Y,VALUE \$

4.144.4 Input Data Blocks

A - Any type of matrix

4.144.5 Output Data Blocks

None.

4.144.6 Parameters

NRØW - Input-integer-default = 1. Row number of elements to be extracted from [A].

NCØL - Input-integer-default = 1. Column identification of element.

VALUE - Output-complex-single precision, default = (0.,0.). Contents of element (NRØW, NCØL) in matrix [A].

4.144.7 Method

The input row and column parameters are checked for validity and the input matrix trailer is read. If the input is purged, the module returns with a value of (0.,0.). The cases that can be satisfied without I/Ø are checked next (e.g., request for off-diagonal term of a diagonal matrix). The data block is then opened and unwanted columns are skipped. The desired column is read element by element until the specified element is located.

4.144.8 Subroutines

None.

MODULE FUNCTIONAL DESCRIPTIONS

4.144.9 Design Requirements

Open core is defined at /SCALXX/ and must be sufficient to hold one GINØ buffer.

4.144.10 Diagnostic Messages

The following diagnostic messages may occur: 3007
(invalid row or column number): 3008.

FUNCTIONAL MODULE CURV (STRESS/STRAIN COORDINATE SYSTEM TRANSFORMATION)

4.145 FUNCTIONAL MODULE CURV

4.145.1 Entry Point: CURV

4.145.2 Purpose

The CURV module transforms the element stresses (or strains/curvatures) output by the SDR2 module to a material coordinate system and, on option, interpolates the stresses (or strains/curvatures) at the grid points to which the elements are connected.

4.145.3 DMAP Calling Sequence

CURV $\left\{ \begin{matrix} \emptyset ES1 \\ \emptyset ES1A \end{matrix} \right\}, MPT, CSTM, EST, SIL, GPL, \left\{ \begin{matrix} \emptyset ES1M \\ \emptyset ES1AM \end{matrix} \right\}, \left\{ \begin{matrix} \emptyset ES1G \\ \emptyset ES1AG \end{matrix} \right\} / C, Y, \left\{ \begin{matrix} STRESS \\ STRAIN \end{matrix} \right\} / C, Y, NINTPTS \$$

4.145.4 Input Data Blocks

- $\emptyset ES1$ - Output element stress requests.
- $\emptyset ES1A$ - Output element strain/curvature requests.
- MPT - Material Property Table.
- CSTM - Coordinate System Transformation Matrices.
- EST - Element Summary Table.
- SIL - Scalar Index List.
- GPL - Grid Point List.

Note: If the $\emptyset ES1$ (or $\emptyset ES1A$) input data block is purged, the CURV module returns without performing any stress (or strain/curvature) computations. None of the other input data blocks may be purged.

4.145.5 Output Data Blocks

- $\emptyset ES1M$ - Output element stress requests (in material coordinate system).
- $\emptyset ES1AM$ - Output element strain/curvature requests (in material coordinate system).
- $\emptyset ES1G$ - Output element stress requests (at grid points).
- $\emptyset ES1AG$ - Output element strain/curvature requests (at grid points).

Notes:

1. $\emptyset ES1M$ (or $\emptyset ES1AM$) may not be purged.
2. $\emptyset ES1G$ (or $\emptyset ES1AG$) may be purged if there are no requests for element stresses (or strains/curvatures) at grid points (Parameter $\left\{ \begin{matrix} STRESS \\ STRAIN \end{matrix} \right\} > 0$. See below).

MODULE FUNCTIONAL DESCRIPTIONS

4.145.6 Parameters

$\begin{Bmatrix} \text{STRESS} \\ \text{STRAIN} \end{Bmatrix}$ - Input-integer-default = -1. Processing flag.

$$\begin{cases} < 0 - \text{generate neither } \begin{Bmatrix} \emptyset\text{ES1M} \\ \emptyset\text{ES1AM} \end{Bmatrix} \text{ nor } \begin{Bmatrix} \emptyset\text{ES1G} \\ \emptyset\text{ES1AG} \end{Bmatrix} . \\ 0 - \text{generate both } \begin{Bmatrix} \emptyset\text{ES1M} \\ \emptyset\text{ES1AM} \end{Bmatrix} \text{ and } \begin{Bmatrix} \emptyset\text{ES1G} \\ \emptyset\text{ES1AG} \end{Bmatrix} . \\ > 0 - \text{generate only } \begin{Bmatrix} \emptyset\text{ES1M} \\ \emptyset\text{ES1AM} \end{Bmatrix} . \end{cases}$$

NINTPTS - Input - integer - default = 0. Number of interpolation points.

$$\begin{cases} \leq 0 - \text{use all independent points in the interpolation.} \\ > 0 - \text{Number of closest independent points to be used in the interpolation.} \end{cases}$$

4.145.7 Method

The CURV module computes stresses or strains/curvatures according as $\emptyset\text{ES1}$ or $\emptyset\text{ES1A}$ data block is input. Each subcase is processed independently of all other subcases, one at a time, until the $\emptyset\text{ES1}$ (or $\emptyset\text{ES1A}$) data block is exhausted. The processing occurs in three phases. Phase 1 is executed once, while Phases 2 and 3 are executed once for each subcase. Note, however, that Phase 3 processing occurs only if the $\emptyset\text{ES1G}$ (or $\emptyset\text{ES1AG}$) data block is to be generated, that is, only if the processing flag parameter (STRESS or STRAIN) is 0. The processing in the three phases is described below. A flow chart of the CURV module is shown in Figure 1.

4.145.7.1 Phase 1 - Initialization and Data Collection

1. The MPT data block is read and the material identification number (MID) and the material coordinate system identification number (MCSID) for each MAT1 and MAT2 material are extracted. Entries having MCSID = 0 are ignored. A paired list of MID-MCSID values is built and sorted on the MIDs.
2. The SIL data block and the first record of the GPL data block are read into core if the $\emptyset\text{ES1G}$ (or $\emptyset\text{ES1AG}$) data block is to be formed.
3. The EST data block is read and element types of interest are examined. Elements referencing materials having MCSID = 0 are ignored. Using the EST data and the SIL and GPL data read earlier, an abbreviated subset of the EST data block, called ESTX, is then built containing all elements of potential interest. This data is written onto SCRATCH1.

FUNCTIONAL MODULE CURV (STRESS/STRAIN COORDINATE SYSTEM TRANSFORMATION)

Elements not selected for output in Case Control for a particular subcase, while they will be placed on ESTX, will not be used for that subcase. A master list of element types that exist on ESTX is also built separately.

The ESTX data block consists of one record for each element type. Within each record, there is a three-word header entry followed by a group of 14 (for TRIAi elements) or 18 (for QUADi elements) words. This latter group is repeated for each element of the specified type. The format of the data on ESTX is as follows for each record (or element type):

	<u>Word</u>	<u>Symbol</u>	<u>Item</u>
Header entry	1	ELTYPE	Element type code
	2	M	Coded word equal to $4 \cdot \text{NPTS} + 2$ where NPTS is given by the next word
	3	NPTS	3 for TRIAi elements, 4 for QUADi elements
Group of 14 (for TRIAi) or 18 (for QUADi) words repeats for each element of specified type	4	EID	Element ID
	5	MCSID	Material coordinate system ID
	6	G1	External grid IDs of connected points (see note below)
	7	G2	
	8	G3	
	9	G4 (for QUADi only)	
	10-12	x_1, y_1, z_1	Basic coordinates of connected grid points
	13-15	x_2, y_2, z_2	
	16-18	x_3, y_3, z_3	
	19-21	x_4, y_4, z_4 (for QUADi only)	

Note: Words 6 through 9 are filled with zeroes if the ØESIG (or ØESLAG) data block is not to be formed.

- A sorted list of MCSIDs that are actually referenced by the elements being considered is built. The MCSIDs in this list are paired with count flags (initially set to 0) for use later when subcases are being handled.

MODULE FUNCTIONAL DESCRIPTIONS

5. The CSTM data block is examined and the coordinate system transformation matrices of those MCSIDs that are in the list above are read into core.
6. The second word of the first record of the first input data block is examined to determine if it is the ØES1 or ØES1A data block. Stress or strain/curvature computations are performed subsequently according as ØES1 or ØES1A data block is input.

4.145.7.2 Phase 2 - Transformation to Material Coordinate System

The following steps are repeated for each subcase.

1. The ØES1 (or ØES1A) and ESTX data blocks (the latter data block is on SCRATCH1) are simultaneously read, element by element. For those entries that appear on both data blocks, the transformation matrix $[U]$, designed to transform the stress (or strain/curvature) values from the element coordinate system to the material coordinate system, is generated via subroutine TRANEM.
2. The element stress (or strain/curvature) values are transformed to the material coordinate system by the transformation $\{T\}_{mat} = [U]\{T\}_{ele}$ and the invariant quantities (see Equations 28, 29, 30 and 31 of Section 8.4.6) are recomputed. It should be noted here that the transformation matrix $[U]$ is designed to transform "tensor" components such as stresses. If strains are to be transformed, the third component must first be multiplied by $\frac{1}{2}$. The "engineering" shear strain is later recovered by multiplying the last component by 2.
3. The resultant stress (or strain/curvature) data are written onto ØES1M (or ØES1AM) for subsequent processing by the ØFP module.
4. If the ØES1G (or ØES1AG) data block is to be formed, data required in Phase 3 are generated and written on scratch files. These consist of an ID record written on SCRATCH3 and a modified version of the ESTX data block, called ESTXX, written on SCRATCH2. The ESTXX data block consists of a single record for the entire subcase under consideration. Within this record, groups of data are written for each element in ESTX. The format of the data for each element is as follows:

FUNCTIONAL MODULE CURV (STRESS/STRAIN COORDINATE SYSTEM TRANSFORMATION)

<u>Word</u>	<u>Symbol</u>	<u>Item</u>
1	MCSID	Material coordinate system ID
2-7	VALUES	Stress (or strain/curvature) values
8-10	VEC	Basic coordinates of element center
11	NPTS	3 for TRIA <i>i</i> elements, 4 for QUAD <i>i</i> elements
12	G_1	External grid IDs of connected points
13	G_2	
14	G_3	
15	G_4 (for QUAD <i>i</i> only)	
16-18	x_1, y_1, z_1	Basic coordinates of connected grid points
19-21	x_2, y_2, z_2	
22-24	x_3, y_3, z_3	
25-27	x_4, y_4, z_4 (for QUAD <i>i</i> only)	

4.145.7.3 Phase 3 - Projection and Interpolation

This phase is executed only if the ØESIG (or ØESIAG) data block is to be generated. It involves the following steps which are repeated for each subcase.

1. The ID record is read from SCRATCH3 which is then rewound.
2. The MCSIDs in the list created in Phase 1 are considered one by one. There is a pass on all of the remaining steps of this phase for each MCSID in this list.
3. The ESTXX data created in Phase 2 are read from SCRATCH2, element by element. Only those elements whose MCSIDs match the MCSID of this pass are considered.
4. The stress (or strain/curvature) data read from SCRATCH2 for the applicable elements are saved on SCRATCH3.
5. Tables of coordinates of the independent points (element centers) and the dependent points (connected grid points) of the applicable elements are built. The latter table is sorted on the external grid IDs of the connected points.

MODULE FUNCTIONAL DESCRIPTIONS

6. Using the CSTM data created in Phase 1, the basic coordinates (r_{basic}) of the independent and dependent points are transformed to the material coordinate system of this pass as follows:

$$\{r\}_{\text{mat}} = [T]^T \{r_{\text{basic}} - v_{\text{basic}}\}$$

where $[T]$ is the transformation matrix and $\{v\}_{\text{basic}}$ is the translation offset vector obtained from the CSTM data. (See Sections 3.4.37 and 2.3.3.4 for details.)

7. The coordinates in the material coordinate system computed above are converted to the mapping coordinates as described below.

Let (x, y, z) be the coordinates in the material coordinate system and let (ρ_1, ρ_2, ρ_3) be the mapping coordinates.

Then, for rectangular mapping coordinates,

$$\rho_1 = x,$$

$$\rho_2 = y,$$

$$\rho_3 = z.$$

For cylindrical coordinate systems,

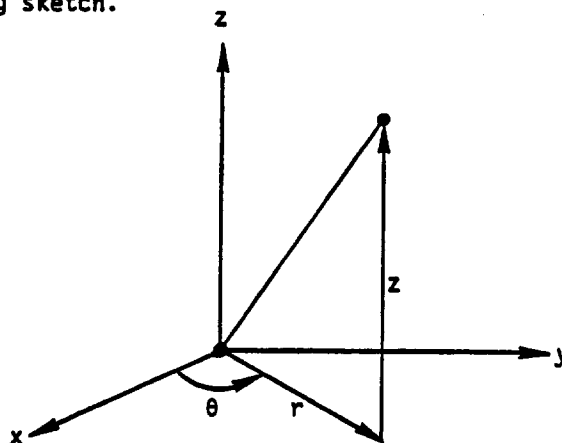
$$\rho_1 = r = \sqrt{x^2 + y^2},$$

$$\rho_2 = \theta = \tan^{-1} \left(\frac{y}{x} \right),$$

and

$$\rho_3 = z,$$

as shown in the following sketch.



FUNCTIONAL MODULE CURV (STRESS/STRAIN COORDINATE SYSTEM TRANSFORMATION)

For spherical coordinate systems, let $\ell = \sqrt{x^2 + y^2}$. Then,

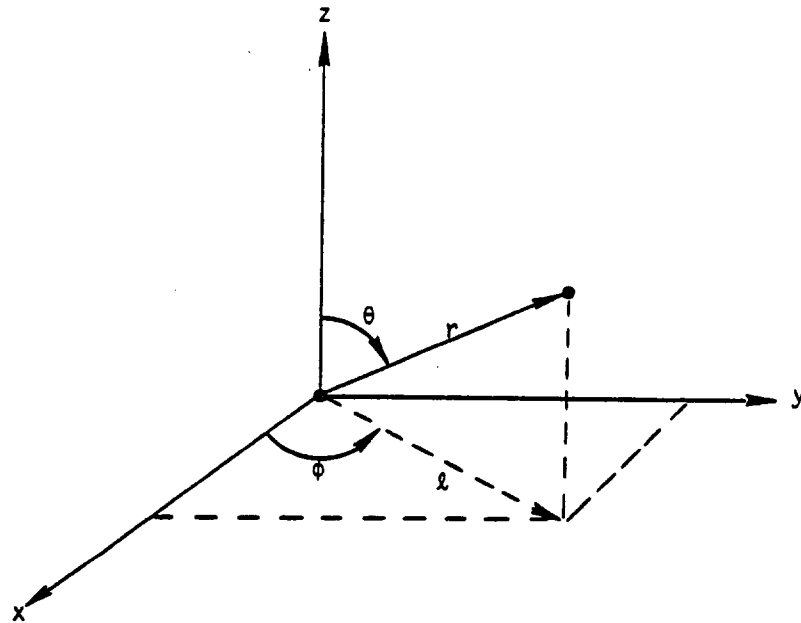
$$\rho_1 = r = \sqrt{x^2 + y^2 + z^2} = \sqrt{\ell^2 + z^2},$$

$$\rho_2 = \theta = \tan^{-1} \left(\frac{\ell}{z} \right),$$

and

$$\rho_3 = \phi = \tan^{-1} \left(\frac{y}{x} \right),$$

as shown in the following sketch.



8. The appropriate projection is selected based on the independent points. To do this, the mapping coordinate which has the smallest range of values needs to be determined. First, define a characteristic length, $\bar{\ell}$ = average r , for cylindrical and spherical coordinate systems.

Then, the desired projection is given by the coordinate which gives

$$\min \{x_{\max} - x_{\min}, y_{\max} - y_{\min}, z_{\max} - z_{\min}\} \quad \text{for rectangular coordinate systems,}$$

$$\min \{r_{\max} - r_{\min}, \bar{\ell}\theta_{\max} - \bar{\ell}\theta_{\min}, z_{\max} - z_{\min}\} \quad \text{for cylindrical coordinate systems,}$$

$$\text{or } \min \{r_{\max} - r_{\min}, \bar{\ell}\theta_{\max} - \bar{\ell}\theta_{\min}, \bar{\ell}\phi_{\max} - \bar{\ell}\phi_{\min}\} \quad \text{for spherical coordinate systems.}$$

The resulting mapping surfaces (for regular mesh spacing) are illustrated in Figures 3, 4 and 5.

MODULE FUNCTIONAL DESCRIPTIONS

9. The independent variables for the interpolation are now selected by discarding the selected projection coordinates and scaling both the remaining two mapping coordinates to have the range -1 to +1. The dependent variables are then arbitrarily reduced and scaled by the same selection and scaling as was used for the independent variables.

The net result of this step is two sets of pair lists,

$$x_i, y_i, i = 1 \text{ to } N_i \quad (N_i = \text{number of independent points})$$

and $x_d, y_d, i = 1 \text{ to } N_d \quad (N_d = \text{number of dependent grid points}),$

which will be used in the next step.

10. The interpolation matrix for the dependent points is computed by subroutine CURVIT via the utility routine SSPLIN. If the NINTPTS parameter is positive, its value is used to determine the number of closest independent points for each dependent point; only these points are used in the SSPLIN interpolation. If $NINTPTS \leq 0$, all independent points are used in the SSPLIN interpolation to obtain all dependent points. In both cases, the points used are the ones shown on the mapping surface sketches in Figures 3, 4 and 5.
11. The stress (or strain/curvature) values at the dependent grid points are computed using the interpolation matrix obtained above. The invariants associated with the stress (or strain/curvature) quantities are computed using Equations 28, 29, 30 and 31 of Section 8.4.6.
12. The resultant stress (or strain/curvature) data are written onto ØESIG (or ØESIAG) for subsequent processing by the ØFP module. This data is generated in external grid point sort.

4.145.8 Subroutines

Utility routines BISLØC, GINØ, GØPEN, GMMATS, ØPEN, PRELØC, PRETRS, SØRT, SSPLIN and WRTTRL are used. See Section 3 for the descriptions of these subroutines. A high level hierarchy diagram for the module is shown in Figure 6.

4.145.8.1 Subroutine Name: CURV1

1. Entry Point: CURV1

FUNCTIONAL MODULE CURV (STRESS/STRAIN COORDINATE SYSTEM TRANSFORMATION)

2. Purpose: To perform the preliminary operations for the module and to set up files and tables for CURV2 and CURV3 subroutines.

3. Calling Sequence: CALL CURV1

4.145.8.2 Subroutine Name: CURV2

1. Entry Point: CURV2

2. Purpose: To form the ØES1M (or ØES1AM) data block and to set up files and tables for CURV3 if ØES1G (or ØES1AG) is to be formed.

3. Calling Sequence: CALL CURV2

4.145.8.3 Subroutine Name: CURV3

1. Entry Point: CURV3

2. Purpose: To form the ØES1G (or ØES1AG) data block.

3. Calling Sequence: CALL CURV3

4.145.8.4 Subroutine Name: CURVIT

1. Entry Point: CURVIT

2. Purpose: To perform local interpolation.

3. Calling Sequence: CALL CURVIT (INDEP, NI, DEP, ND, IFILE, Z, IZ, LZ, MCLØSE, TØLER, MCSID, XSCALE, YSCALE)

INDEP - x, y coordinates of independent points (element centers) - real - input.

NI - Number of independent points - integer - input.

DEP - x, y, coordinates of dependent grid points - real - input.

ND - Number of dependent grid points - integer - input.

IFILE - GINØ file number of scratch file - integer - input.

Z - Pointer to open core, length LZ - real - input.

IZ - Pointer to open core, length LZ - integer - input.

LZ - Length of open core - integer - input.

MCLØSE - Number of closest independent points to use - integer - input.

TØLER - Tolerance for including independent points in the interpolation - real - input.

MODULE FUNCTIONAL DESCRIPTIONS

MCSID - Material coordinate system identification number - integer - input.

XSCALE - Scale factor for x values - real - input.

YSCALE - Scale factor for y values - real - input.

4.185.8.5 Subroutine Name: CURVPS

1. Entry Point: CURVPS

2. Purpose: To compute principal stresses (or strains/curvatures).

3. Calling Sequence: CALL CURVPS (SIGS, PRIN)

SIGS - Stresses (or strains/curvatures) - real - input.

PRIN - Principal stresses (or strains/curvatures) - real - output.

4.185.8.6 Subroutine Name: TRANEM

1. Entry Point: TRANEM

2. Purpose: To compute a transformation matrix [U] for the TRIAi and QUADi elements which converts the stress (or strain/curvature) vectors from the element coordinate system to a material coordinate system projected on the surface of the element.

$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau \end{Bmatrix}_{\text{material coordinates}} = [U] \begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau \end{Bmatrix}_{\text{element coordinates}}$$

3. Calling Sequence: CALL TRANEM (MCSID, NG, R, ICØMP, U, RC)

MCSID - Material coordinate system identification number - integer - input.

NG - 3 for TRIAi and 4 for QUADi - integer - input.

R - Basic coordinates of connection points, length = 3*NG - real - input.

ICØMP - 1 if material x-axis is used, 2 if material y-axis is used - integer - output.

U - Transformation matrix, row stored, length = 9 - real - output.

RC - Basic coordinates of element center, length = 3 - real - output.

Requirement: The calling routine must call PRETRS.

FUNCTIONAL MODULE CURV (STRESS/STRAIN COORDINATE SYSTEM TRANSFORMATION)

4. Method:

- a. Compute the element normal:

$$\begin{aligned}\text{TRIAi} - \vec{n} &= \vec{v}_{13} \times \vec{v}_{23} \\ &= (\vec{v}_3 - \vec{v}_1) \times (\vec{v}_3 - \vec{v}_2)\end{aligned}$$

$$\begin{aligned}\text{QUADI} - \vec{n} &= \vec{v}_{13} \times \vec{v}_{24} \text{ (definition)} \\ &= (\vec{v}_3 - \vec{v}_1) \times (\vec{v}_4 - \vec{v}_2)\end{aligned}$$

- b. Compute the center of the element:

$$\vec{RC} = \frac{1}{NG} \sum \vec{R}$$

- c. Call TRANS to compute the unit vectors of the material coordinate system at the center of the element.

- d. Select the reference axis:

If the element normal is not normal to the material coordinate x-axis projection, use the x-axis and set ICOMP=1. Otherwise, use the material coordinate y-axis and set ICOMP=2. The criterion is arbitrarily chosen to be $\vec{n} \cdot \vec{i}_m > 0.4$ for use of the y-axis rather than the x-axis.

- e. Compute the sine and cosine of the angle θ between the selected material coordinate system axis and the element coordinate system x-axis.

- f. Generate the transformation matrix U:

$$U_{11} = \cos^2 \theta$$

$$U_{12} = \sin^2 \theta$$

$$U_{13} = -\sin \theta \cos \theta$$

$$U_{21} = \sin^2 \theta$$

$$U_{22} = \cos^2 \theta$$

$$U_{23} = \sin \theta \cos \theta$$

$$U_{31} = \sin 2\theta$$

$$U_{32} = -\sin 2\theta$$

$$U_{33} = \cos 2\theta$$

MODULE FUNCTIONAL DESCRIPTIONS

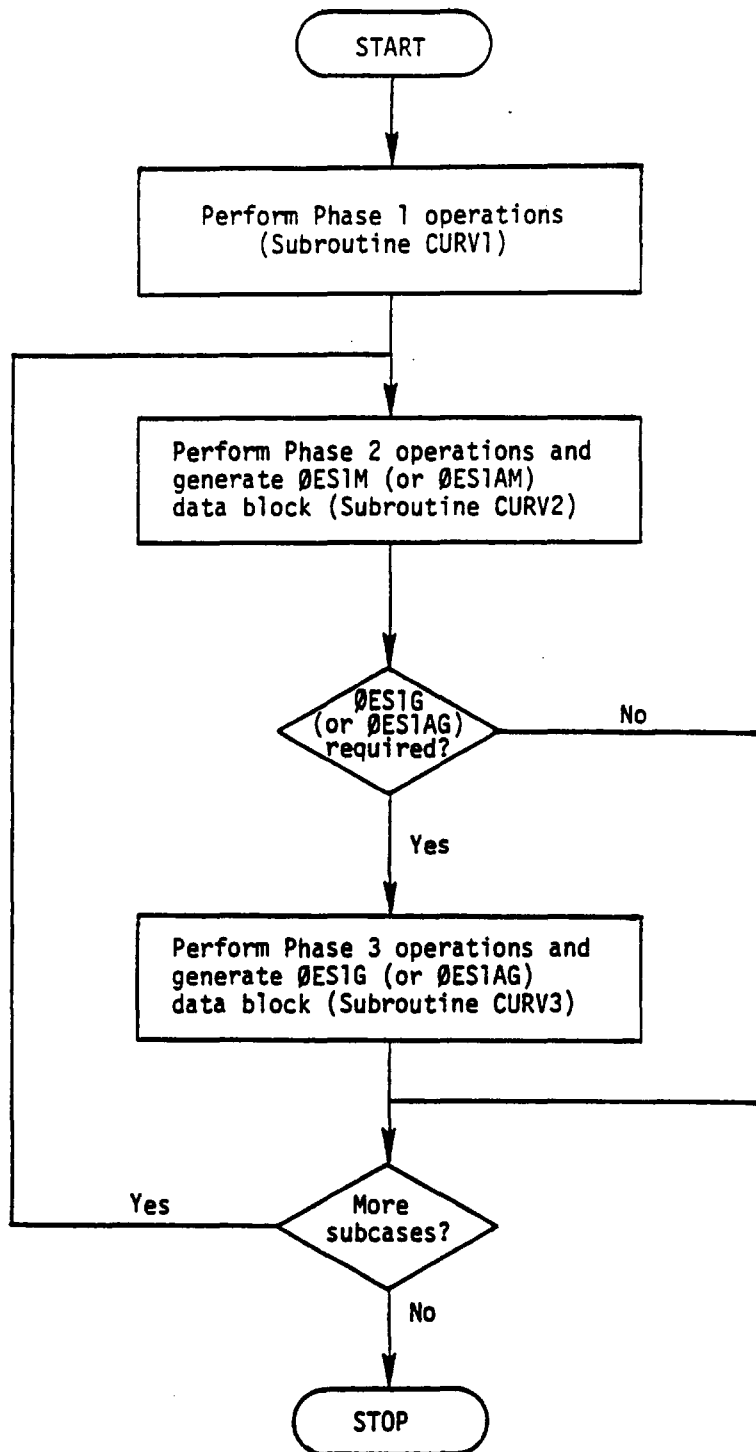
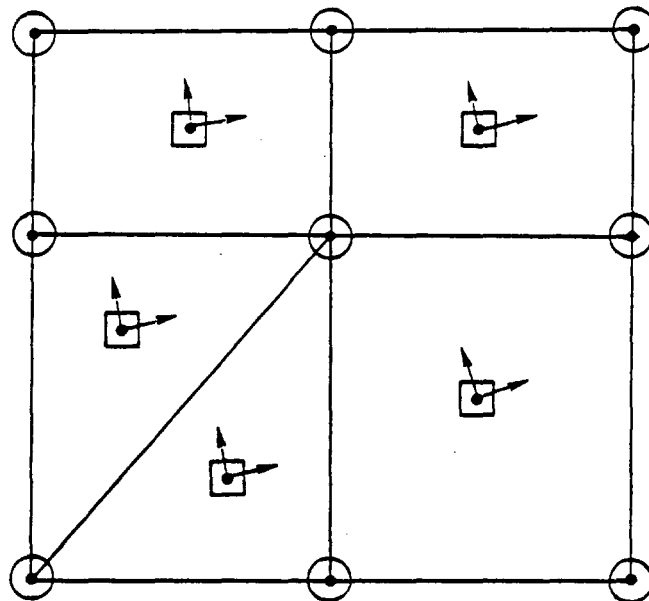


Figure 1. CURV module flow chart

FUNCTIONAL MODULE CURV (STRESS/STRAIN COORDINATE SYSTEM TRANSFORMATION)



● Grid point locations (dependent points)

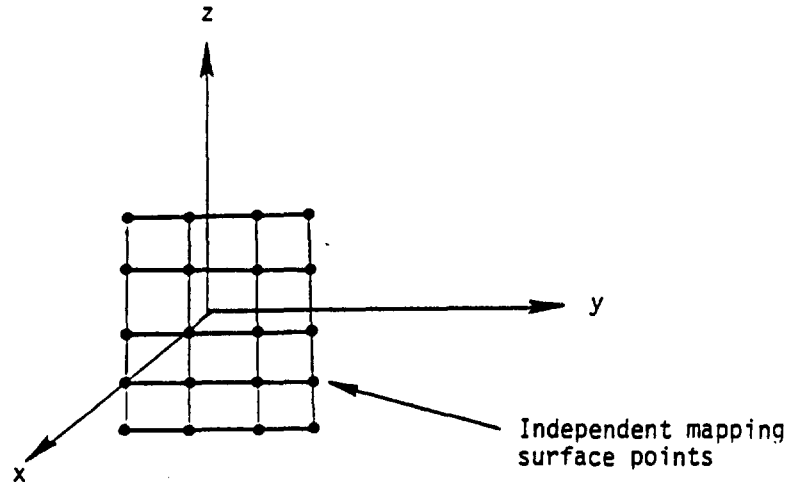
■ Element center locations (independent points)

↖ ↗ Material coordinate system

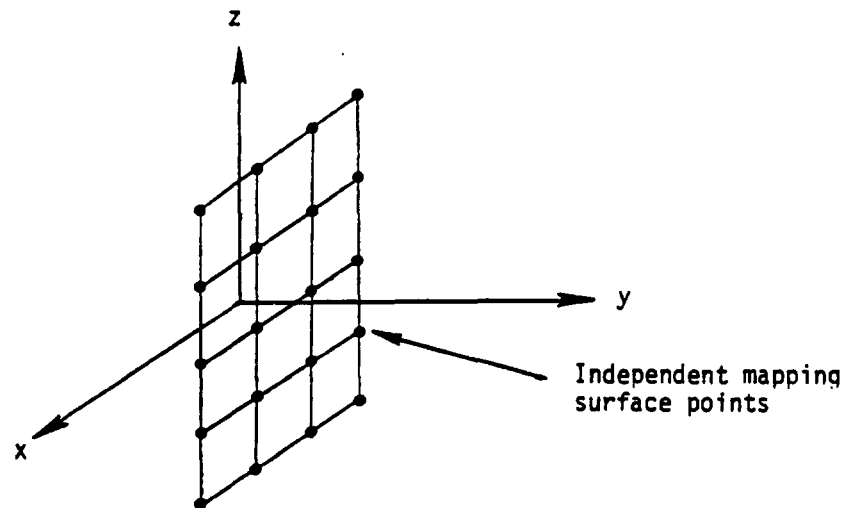
Figure 2. Element geometry

MODULE FUNCTIONAL DESCRIPTIONS

$x = \text{constant}$



$y = \text{constant}$



$z = \text{constant}$

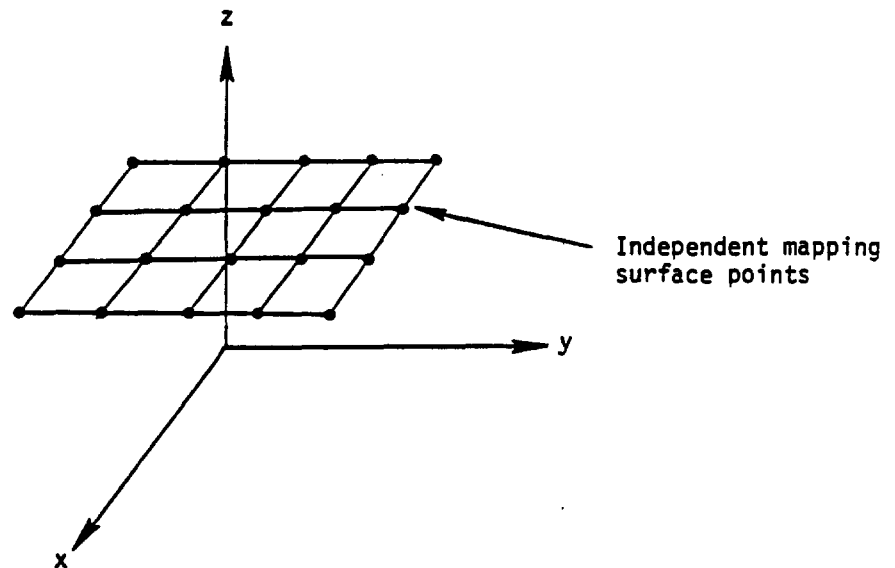


Figure 3. Rectangular mapping surfaces (uniformly spaced meshes are shown for convenience)

FUNCTIONAL MODULE CURV (STRESS/STRAIN COORDINATE SYSTEM TRANSFORMATION)

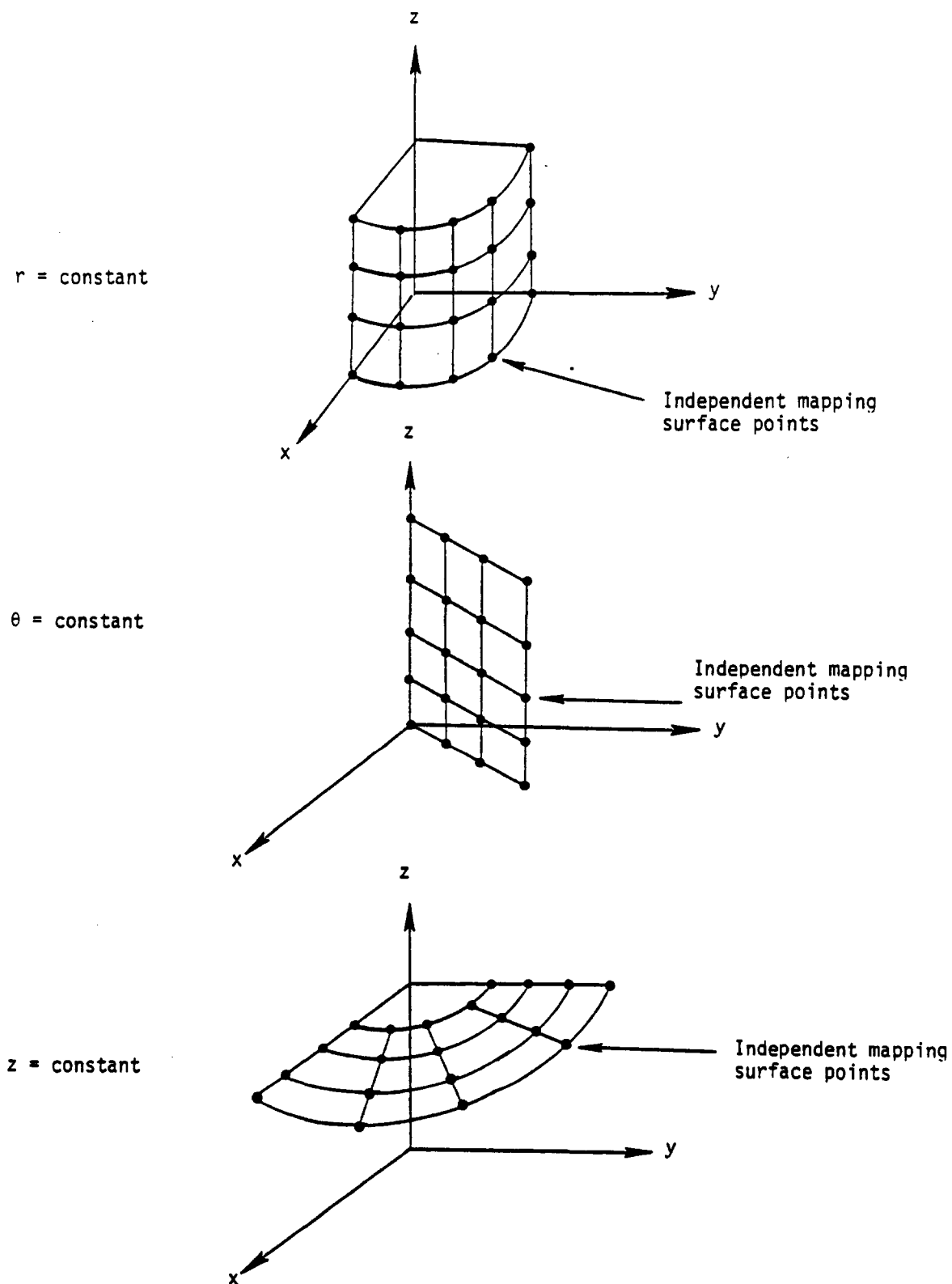


Figure 4. Cylindrical mapping surfaces (uniformly spaced meshes are shown for convenience)

MODULE FUNCTIONAL DESCRIPTIONS

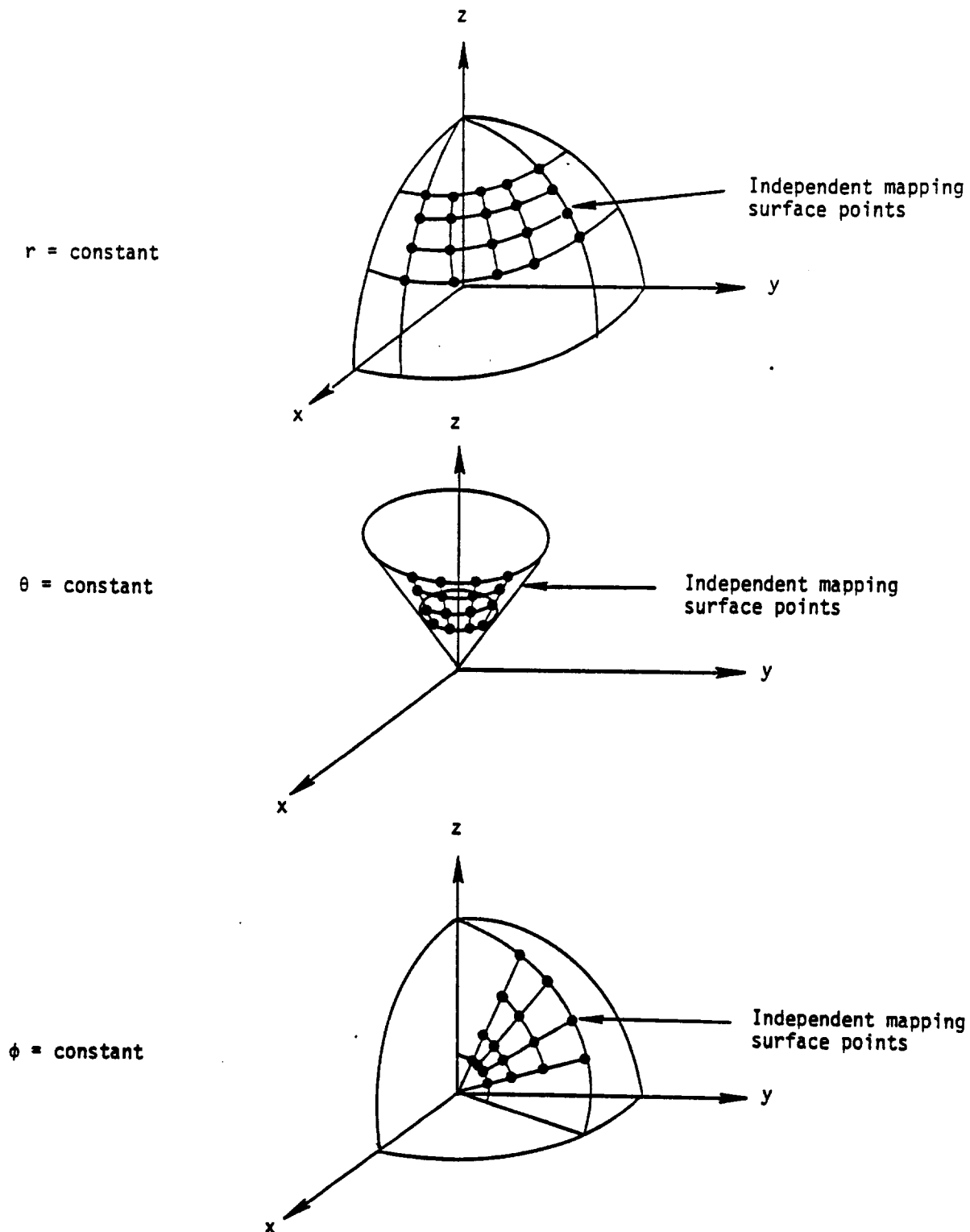


Figure 5. Spherical mapping surfaces (uniformly spaced meshes are shown for convenience)

FUNCTIONAL MODULE CURV (STRESS/STRAIN COORDINATE SYSTEM TRANSFORMATION)

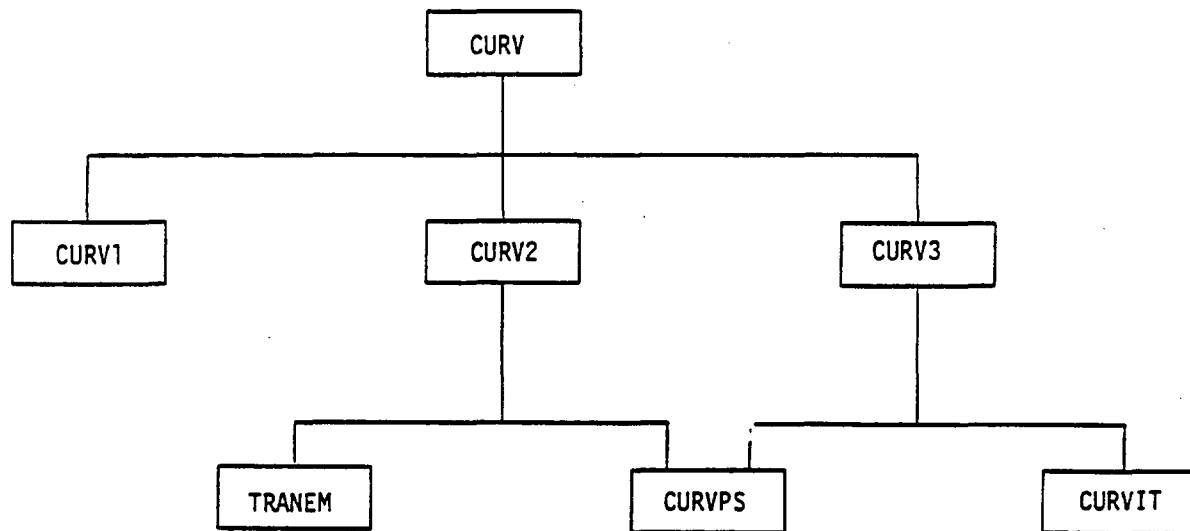


Figure 6. Hierarchy diagram for the CURV module.

FUNCTIONAL MODULE GPFDR (GRID POINT FORCE DATA RECOVERY)

4.146 FUNCTIONAL MODULE GPFDR (GRID POINT FORCE DATA RECOVERY)

4.146.1 Entry Point: GPFDR

4.146.2 Purpose

1. To prepare an element-strain-energy table for a user-selected set of elements. These selected elements are listed by type with their strain energy, and per cent of total strain energy with respect to all elements. The strain energy computed is equal to

$$\frac{1}{2} \{F_e\}^T \{u_e\}$$

The total energy is arrived at by summing the element strain energies of all elements for which stiffness matrices exist.

2. To prepare a grid-point-force-balance-table for a user-selected set of points. This table lists the forces acting at each selected point due to element constraints, single-point constraints, and applied loads. Also listed is the sum total of these forces which represents the balance in an opposite direction due to multipoint constraints, general elements, round-off errors, and other nonlisted sources.

4.146.3 DMAP Calling Sequence

GPFDR CASECC,UGV,KMAT,KDICT,ECT,EQEXIN,GPECT,PG,QG/ØNRGY1,ØGPFB1/C,N,STATICS \$

4.146.4 Input Data Blocks

CASECC Case Control data table

UGV Displacement vector matrix giving displacements in the g set

KMAT Data table containing element stiffness matrices

KDICT Data table relating the element stiffness matrices in table KMAT to the problem being solved

ECT Element Connection Table

EQEXIN Table of equivalences between external and internal point ID's, and also external point ID's and scalar indices

GPECT Table of grid points and their associated element connections

MODULE FUNCTIONAL DESCRIPTIONS

PG Static load vector matrix giving loads in the g set

QG Single point constraint vector matrix giving single point constraints in the g set

4.146.5 Output Data Blocks

ØNRGY1 Table of selected element energies for output by the ØFP module

ØGPFB1 Table of selected grid-point-force-balances for output by the ØFP module

4.146.6 Parameters

STATICS BCD parameter indicating a statics solution

4.146.7 Method

First, elements for strain energy output, and/or connecting grid points requested for grid-point-force-balance outputs, have their element force vectors, and/or strain energies computed and stored via GINØ write comments. The element forces computed are $\{F_e\} = -[K_e]\{u_e\}$. The strain energy computed is

$$\frac{1}{2} \{F_e\}^T \{u_e\}$$

The element strain energies are output by element types after the sum total of all element energies is at hand.

Second, if requested, the grid-point-force-balance outputs are prepared in a series of three steps. The first step, after obtaining element forces for elements connected to grid points scheduled to be output in the balance summary, involves the passing of the GPECT data block and collection of the appropriate element forces acting at those points to be output.

The second step involves the selection of applied loads from PG and single-point constraint forces from QG for those points scheduled to be output.

The final step involves the output of the element forces, applied loads, single-point constraint forces, and sum totals in external point ID order. Further discussion of the method is found in the subroutine description for subroutine GPFDR below.

FUNCTIONAL MODULE GPFDR (GRID POINT FORCE DATA RECOVERY)

4.146.8 Subroutines

4.146.8.1 Subroutine Name: GPFDR

1. Entry Point: GPFDR
2. Purpose: Main driving routine and processor of the GPFDR module.
3. Calling Sequence: CALL GPFDR.
4. Method: A check of the module parameter is made for acceptability. Six GINØ buffers are allocated and a core sufficiency check is made. Subroutine DELSET is called to initialize data in common /GPTA1/ for any "dummy elements" present. CASECC and UGV are opened and positioned to the first non-header record. The order of the problem size g is obtained from the trailer of UGV. Header records are written to the two output data blocks and their existence is noted.

After the above initialization, a large loop is executed once for each subcase record found in data block CASECC. This loop consists of the following:

- a. Read a record of CASECC.
- b. Determine the type of requests for grid-point-force-balance and element strain energy. If requests are for a subset of points or elements, the set desired is found in the CASECC record table and pointers initialized.
- c. Depending on whether this is a REPCASE, symmetry sequence or normal subcase, a record position pointer is modified for UGV, PG and QG to indicate which vector, or which sequence of vectors to use. This positioning is necessary in the case of a REPCASE which uses the vector of the previous subcase or symmetry sequence which uses the previous N subcases of vectors. It is also noted that some subcases may have requests for grid-point-force-balance and/or element-strain-energies while others may not.
- d. The full vector or sequence of vectors is added to core for displacements. The logic to do this is also used as an internal subroutine callable later to obtain a PG vector or QG vector.
- e. Data blocks CASECC and UGV are closed at this point to free GINØ buffers for use by other data blocks.

MODULE FUNCTIONAL DESCRIPTIONS

- f. On the first subcase pass the second record of the EQEXIN data block is brought into core. It contains two-word entries giving,

1 - external point ID

2 - $10 \times \text{SIL} + \text{INDEX}$

where INDEX = 1 if 6 degrees of freedom

= 2 if 1 degree of freedom.

As these two-word entries are in sort on external ID, they are sorted on SIL's to insure that they are not only in sort on SIL's but by consequence are in sort by internal ID (i.e., the position of an entry gives its internal ID). For convenience later, the INDEX associated with each SIL is removed from same and attached to each external ID.

- g. Data blocks KMAT and KDICT are opened and the precision of the element stiffness matrices in KMAT is noted. All calculations of this module are in single precision, although KMAT may supply double-precision inputs.
- h. Data block ECT is opened and at this point a pass is made of KDICT. Stiffness data exists only for elements found listed in KDICT which may be a subset of those found in ECT. Each record of KDICT represents one element type. As elements are identified in KDICT by the so-called element-internal-ID or "ESTID" and the ESTID is based on each element found in ECT of all types, the ECT has to be read and carefully maintained in sync as the ESTID_s found in KDICT while numerically increasing need not be contiguous.
- i. As each new element type is read from KDICT, element type parameters are set and the corresponding element type is found in ECT. If element strain energies are requested for any elements at all, then all element force vectors must be computed regardless of whether grid-point-force-balance for any point is requested. But only those force vectors computed for grid points which are requested to have a force-balance-output will be saved by the GINØ direct access method. In this case they are written expanded to represent degrees of freedom 1 thru 6 to scratch data block 1. A corresponding dictionary entry is prepared for each element having one or more of its grid point force vectors preserved in direct access on scratch data block 1. This entry

FUNCTIONAL MODULE GPFDR (GRID POINT FORCE DATA RECOVERY)

is of the following form:

<u>Word</u>	<u>Description</u>
1	ESTID from KDICT
2	External element ID from ECT
3	GINØ - LØCS as returned by SAVPØS subroutine when writing a six-word point force vector.
:	
:	
LPDICT	

LPDICT = 2 + number of grid points for the element type. Force vectors of only those element points in the output grid-point-balance are preserved and thus zeros may appear in the element-force-vector dictionary entries written to scratch data block 2.

Each element type for which an element grid-point-force vector is preserved results in a record being started on scratch data block 2 of the corresponding dictionaries, and this record is prefaced with a three-word entry, giving

1. the element type number
2. the value LPDIC
3. the number of grid points.

- j. Element strain energies destined for output are written to scratch data block 3 as two-word entries, giving

1. the element ID
2. the strain energy.

Each element type which has an entry written to scratch data block 3 results in the initialization of a record containing a two-word entry, giving

1. the element type name first 4H
2. the element type name second 4H.

- k. After an end of file is encountered on KDICT, data blocks KMAT, KDICT, ECT, SCRATCH1, SCRATCH2, and SCRATCH3 are all closed.

1. If element energies were computed, a running total has been computed and at this time an ØFP type pair of records is written to ONRGY1 for each element type having any strain energy output.

MODULE FUNCTIONAL DESCRIPTIONS

Record 1 is a 146-word ØFP identification type record (see data block descriptions for ONRGY1, Section 2.3.79.1).

Record 2 contains 3-word data entries. To form these, the two-word element-ID and strain energy entries are read from SCRATCH3. Their per cent of total strain energy is computed, and a three-word entry is written to ONRGY1.

- m. For purposes of creating the grid-point-force-balance summary for points requested, the GPECT is passed to collect the force vectors associated with each point to be output.

First, all dictionary information relating to the element-grid-point-force vectors is distributed into core. Two tables are formed in core. The first contains three words for each element type possible (many of which are not in any given problem). This table is set to zero. As each record found on SCRATCH2 is read, its three-word header entry is read, giving

1. Element type
2. Length of entries to follow
3. Number of grid points.

The entries are then blast-read into the second table at the next available position. The number of entries read is computed and for the element type indicated the three words in the first table are set to

1. INDEX into open core to the data of the second table
2. Total length of data
3. Number of entries (number of elements) represented.

- n. At this point it should be noted that the purpose of the following and previous logic is to avoid completely what would be a potentially large sort were the element-point-force vectors merely placed into a large pool for output.

The GPECT permits a very nice method of collecting all element force contributions, however, the points in the GPECT are not necessarily in external point ID order and for purposes of outputting the grid point force balance in external order, and for the ease of merging in the applied-load and forces of single-point-constraint, it is now necessary to build a grid-point-force-balance-output-map, hereafter referred

FUNCTIONAL MODULE GPFDR (GRID POINT FORCE DATA RECOVERY)

to as the GPFBOM (pronounced G-P-F-BOMB). This file of data is placed on SCRATCH2 and contains one GINØ direct access record for each grid point to be output. In these records then are repeating four-word entries, giving

1. External element ID
2. Element type name first 4H
3. Element type name second 4H
4. GINØ-locate code to the 6 x 1 force vector on SCRATCH1.

For each record of the above four-word entries written to SCRATCH2, a corresponding three-word entry is sequentially written to SCRATCH3, giving

1. the external grid point ID
2. the GINØ-locate code to the SCRATCH2 record of four-word entries
3. the number of entries in the record.

Thus, the GPECT is passed. The SIL is converted to external point ID, it is checked for output request in the case control table, and set list if necessary. Then, the elements listed are collected by finding the pointer of their type to the force-vector dictionary data, and by picking out the appropriate entry using BISRCH and the ESTID. A four-word entry is formed and added to the GPFBOM.

- o. When the GPECT has been completely passed, all open data blocks are closed.
- p. At this point applied loads and forces of single-point constraint of requested output points are found and written to SCRATCH4. Only nonzero entries are written. These entries contain the entire ten words to eventually be output in the grid point force balance.

- 1 - External point ID*10 + device code
 - 2 - 0
 - 3 - { 4HAPP- } or { 4HF-OF }
 - 4 - { 4HLOAD } or { 4H-SPC }
 - 5 - T1
 - 6 - T2
 - 7 - T3
 - 8 - R1
- } Load or SPCF contribution (continued next page)

MODULE FUNCTIONAL DESCRIPTIONS

9 - R2	}	Load or SPCF contribution
10 - R3		

In the case of scalar points, only T1 is set. T2 thru R3 will be 0. When all applied-load and force-of-single-point-constraint entries are on SCRATCH4, they are read back into core, sorted on external ID and written back to SCRATCH4 for later merge with the element force contributions.

- q. To reduce the number of sequential passes of the element-grid-point-force vectors on SCRATCH1, open core is used in its entirety except for the EQEXIN table and buffer area to collect in external sort, the grid-point-force-balance outputs.

SCRATCH3 containing three-word entries pointing to the GPFBOM data is blast-read into core. Each three-word entry represents one grid point. Thus, first these three-word entries are sorted on external ID. The amount of core available in the final assembly is divided by 12 to determine the number of line entries which may be held. This number is used to pass the three-word entries now in core and to partition these into groups.

Once the three-word entries are partitioned in groups, the third word of each three-word entry (now containing the number of four-word GPFBOM entries) is converted to an output entry order within the group. Once this has been performed each group of three-word entries can be sorted on their GINØ-locate codes (word 2 of each entry). Each group is then written back out to SCRATCH3.

- r. Core is allocated into two tables at this point, TABLE 1 and TABLE 2. TABLE 1 contains two-word entries, TABLE 2 contains ten-word entries. The number of entries in TABLE 1 and TABLE 2 is the same and greater than or equal to the number of line entries represented by any group of GPFBOM output points to be processed at one time.
- s. SCRATCH3 is read again for each record and the following takes place. A three-word entry is read, the GPFBOM (SCRATCH2) is positioned using the GINØ-locate code, the four-word entries are read and their data placed in core at the appropriate place of TABLE 2 as dictated by word 3 of the SCRATCH3 entry. The GINØ-locate code of each four-word entry is placed in TABLE 1 along with a pointer to the ten-word entry of TABLE 2. This then is accomplished for all three-word entries of one record on SCRATCH2. The GINØ-locate codes of each four-word GPFBOM entry were placed in TABLE

FUNCTIONAL MODULE GPFDR (GRID POINT FORCE DATA RECOVERY)

l so that they may now be sorted on GINØ locate code. This now results in only one pass of the force vectors on SCRATCH1 being necessary for all the data being formed in core. This is now accomplished and the 6 x 1 vectors are added to the ten-word entries.

- t. The ten-word entries now complete and in external grid point order are output, and in the process entries from SCRATCH4 giving applied-load and forces-of-single-point constraint are merged in. These outputs are made to ØGPF1 along with a sum total entry at the conclusion of each point-ID.
- u. Return is made at this point to (r). When no more groups remain, this subcase is complete and the next is started.

4.146.9 Design Requirements

GPFDR being essentially an output data block preparation module, not affecting a problem's later calculations, attempts to produce only that which it is able to, and in the event of a user or system type error, to exit with warning and/or information-type messages in a normal manner.

Open core must be able to hold the second record of EQEXIN, one case control record, one g-size vector.

Additional core, where all grid points will be output in the grid-point-force balance, will not significantly improve running time, but will reduce I/Ø in the ratio of increased core.

4.146.10 Diagnostic Messages

Diagnostic messages 2342 through 2354 may be issued by GPFDR.

FUNCTIONAL MODULE SWITCH

4.147 FUNCTIONAL MODULE SWITCH

4.147.1 Entry Point: SWITCH

4.147.2 Purpose

To interchange two data block names.

4.147.3 DMAP Calling Sequence

SWITCH DB1,DB2 // PARAM \$

4.147.4 Input Data Block

DB1

- Any NASTRAN data block

DB2

4.147.5 Output Data Block

None

4.147.6 Parameters

PARAM - Input, integer, no default value.

4.147.7 Method

If the value of PARAM is not less than zero, a return is made, otherwise the module operates directly on the FIAT interchanging the data block names only. All characteristics other than the name remain with the data.

4.147.8 Subroutine

4.147.8.1 Subroutine Name: SWITCH

1. Entry Point: SWITCH
2. Purpose: To interchange two input data block names in the FIAT.
3. Calling Sequence: CALL SWITCH
4. Method: See Section 4.147.7.

MODULE FUNCTIONAL DESCRIPTIONS

4.147.9 Design Requirements

Module SWITCH requires the common blocks:

/XFIAT/

/XFIST/

/XPFIST/

4.147.10 Error Messages

SWITCH may generate system fatal message 1.

FUNCTIONAL MODULE COPY

4.148 FUNCTIONAL MODULE COPY

4.148.1 Entry Point: COPY

4.148.2 Purpose

To generate a physical copy of a NASTRAN data block.

4.148.3 DMAP Calling Sequence

COPY DB1 / DB2 / PARAM \$

4.148.4 Input Data Blocks

DB1 - Any NASTRAN data block

4.148.5 Output Data Blocks

DB2 - Any NASTRAN data block

4.148.6 Parameters

PARAM - Input, integer, no default value

4.148.7 Method

If the value of PARAM is ≥ 0 a return is made with no action. If $\text{PARAM} < 0$, the input data block is read into open core one logical record at a time and then written into the output data block. Spill logic has been employed to allow functioning in small open core. This logic blast-reads as many words as will fit in core, then dumps them to the output data block and resumes reading. The input data block trailer is read and copied for the output data block.

4.148.8 Subroutine

4.148.8.1 Subroutine Name: COPY

1. Entry Point: COPY
2. Purpose: To generate a physical copy of a NASTRAN data block.
3. Calling Sequence: CALL COPY
4. Method: See Section 4.148.7.

MODULE FUNCTIONAL DESCRIPTIONS

4.148.9 Design Requirements

1. Open core resides in /CØPYZZ/.
2. Open core must be larger than two GINØ buffers.

4.148.10 Diagnostic Messages

CØPY may produce system fatal messages 1 and 8.

FUNCTIONAL MODULE FRLG (FREQUENCY RESPONSE LOAD GENERATOR)

4.149 FUNCTIONAL MODULE FRLG (FREQUENCY RESPONSE LOAD GENERATION)

4.149.1 Entry Point: FRLG

4.149.2 Purpose

To generate frequency dependent loads from RLØAD1 and RLØAD2 cards or via a transform equation from TLØAD1 and TLØAD2 cards.

4.149.3 DMAP Calling Sequence

FRLG CASEXX, USETD, DLT, FRL, GMD, GØD, DIT, PHIDH/PPF1, PSF1, PDF1, FØL, PHF/V, N, FØRM=MØDAL/
V, N, FREQ/V, N, APP \$

4.149.4 Input Data Blocks

CASEXX - Case Control Data Table - dynamics
USETD - Displacement set definition table - dynamics
DLT - Dynamic Loads Table
FRL - Frequency Response List
GMD - Multipoint constraint transformation matrix - dynamics
GØD - Omitted coordinate transformation matrix - dynamics
DIT - Direct Input Tables
PHIDH - Transformation matrix from d-set to modal coordinates

- Notes:
1. CASEXX, USETD, DLT and FRL cannot be purged.
 2. GMD and GØD cannot be purged if multipoint constraints or omitted coordinates are used.
 3. PHIDH cannot be purged if FØRM=MØDAL.
 4. DIT cannot be purged if a load uses tables.

4.149.5 Output Data Blocks

PPF1 - Load vectors p set
PSF1 - Load vectors s set
PDF1 - Load vectors d set
PHF - Load vectors h set
FØL - Frequency response output list

MODULE FUNCTIONAL DESCRIPTIONS

- Notes: 1. PPF, PDF and PHF cannot be purged.
2. PSF cannot be purged if single point constraints exist.

4.149.6 Parameters

FØRM - Input - BCD - no default. FØRM=MØDAL implies a modal solution is being requested.
FREQY - Output - Integer - default = -1. If the problem is a transient problem, FREQY is set +1, otherwise it is set -1.
APP - Input - BCD - default = FREQ. Formulation for loading conditions. If APP = FREQ, RLØAD1 or RLØAD2 cards are used. If APP = TRAN, TLØAD1 or TLØAD2 loads are computed and transformed to the frequency domain.

4.149.7 Method

The method of assembly and reduction of the load vectors is described in Section 4.61.7.3 of the Programmer's Manual.

4.149.8 Subroutines

Utility routines PRETAB, TAB, CALCV, SSG2B and SSG2A are used. In addition, subroutines FRRD1A and FRRD1B are called. These are documented in Section 4.61 of the Programmer's Manual.

4.149.9 Design Requirements

Four scratch files are used. Open core is described in Section 4.61.9 of the Programmer's Manual.

FUNCTIONAL MODULE LAMX (EDIT LAMA DATA BLOCK)

4.150 FUNCTIONAL MODULE LAMX (EDIT LAMA DATA BLOCK)

4.150.1 Entry Point: LAMX

4.150.2 Purpose

Edit or create a real eigenvalue table (LAMA).

4.150.3 DMAP Calling Sequence

LAMX EDIT,LAMA/LAMB/C,Y,NLAM=0 \$

4.150.4 Input Data Blocks

EDIT - A matrix created with DMI

LAMA - Real eigenvalue table

Note: Both EDIT and LAMA may be purged.

4.150.5 Output Data Blocks

LAMB - Created or edited real eigenvalue table.

Note: LAMX may be purged only if EDIT and LAMA are purged.

4.150.6 Parameters

NLAM - Input-Integer-No default. Maximum number of modes in the output data block. If
NLAM = 0, number of records in LAMB is equal to records in LAMA. If NLAM < 0, LAMB will
be a matrix.

4.150.7 Method

Records of the LAMA data block are to be copied, edited or created on the output LAMB data block. The editing information is contained on the EDIT matrix. LAMA is described in Section 2.3.87.1 of the Programmer's Manual. Record one is copied to the output data block (or created) while records 2 to the end are edited (or created).

The EDIT matrix contains three rows for each mode. "A" is the first row, "B" is the second row and "C" is the third row. Note, "B" and "C" will be set to zero if insufficient rows are supplied. If A=B=C=0.0, then no change to the input mode is desired.

MODULE FUNCTIONAL DESCRIPTIONS

If LAMA exists, then the process is editing (or copying). If EDIT does not exist, the NLAM records will be copied to LAMB from LAMA. If EDIT exists then LAMA is edited. If C is negative the record is skipped. If $C \geq 0$ then:

Word 5 - $f = A + (1.0+B)f_0$ where f_0 is the old word 5

Word 4 - $\omega = 2\pi f$

Word 3 - $\lambda = \omega^2$

Word 1 - New sequence number of record; changes only if modes are deleted.

Word 6 - $m = \text{old value if } C = 0$
 $= C \text{ if } C > 0$

Word 7 - $k = \lambda m$

If LAMA is purged, then LAMB is created from EDIT:

Word 1 - column number of EDIT - mode number

Word 2 - Word 1 - extraction order

Word 3 - $(2\pi A)^2$ - λ

Word 4 - $2\pi A$ - ω

Word 5 - A - f

Word 6 - C - generalized mass

Word 7 - $C(2\pi A)^2$ - generalized stiffness

If the parameter NLAM is equal to zero, then the number of eigenvalue records is found from: (1) the number of eigenvalues in LAMA minus the number deleted; or (2) the number of columns of EDIT in LAMA is purged. If NLAM is positive, then the number will not exceed NLAM.

If NLAM is less than zero, a matrix will be built on LAMB. Columns will be built with eigenvalue, Omega, frequency, generalized mass and generalized stiffness until the generalized mass is zero. The number of rows should then match the number of eigenvectors requested.

FUNCTIONAL MODULE LAMX (EDIT LAMA DATA BLOCK)

4.150.8 Subroutines

LAMX is the only subroutine.

4.150.8.1 Subroutine Name: LAMX

1. Entry Point: LAMX
2. Purpose: Edit or create a real eigenvector table.
3. Calling Sequence: CALL LAMX

4.150.9 Design Requirements

Open core is at LAMXXX. Up to three buffers may be used. No error messages are used.

FUNCTIONAL MODULE FRRD2 (FREQUENCY RESPONSE - WITH AEROELASTIC)

4.151 FUNCTIONAL MODULE FRRD2 (Frequency Response - with Aeroelastic)

4.151.1 Entry Point: FRRD2

4.151.2 Purpose

To solve the matrix equation

$$[-\omega^2[M] + i\omega[B] - Q*B\emptyset V[Q^I]] + [-[K] - Q[Q^R]]][U] = [P]$$

at a given set of frequencies, ω_i and loads, P.

4.151.3 DMAP Calling Sequence

FRRD2 KHH, BHH, MHH, QHHL, PHF, F \emptyset L/UHVF/V,N,B \emptyset V/C,Y,Q/C,Y,MACH \$

4.151.4 Input Data Blocks

KHH - modal stiffness matrix - h-set
BHH - modal damping matrix - h-set
MHH - modal mass matrix - h-set
QHHL - aerodynamic matrix list - h-set
PHF - load vectors - h-set
F \emptyset L - frequency response output list

Note: PHF and F \emptyset L cannot be purged.

4.151.5 Output Data Blocks

UHVF - Displacement vector - h-set

Note: UHVF cannot be purged.

4.151.6 Parameters

B \emptyset V - Input-Real-No default. Reference frequency over velocity.
Q - Input-Real-No default. Dynamic presence scale factor.
MACH - Input-Real-No default. Mach number

Note: None of the parameters are used if QHHL is purged.

MODULE FUNCTIONAL DESCRIPTIONS

4.151.7 Method

Before looping over all the frequencies, an interpolated matrix list $[Q_{HIL}]$ is built by subroutine FRD2I. This matrix is used to build the $[Q^R]$ and $[Q^I]$ matrices inside the frequency loop.

A loop is then made over each frequency. During this loop all the loads associated with this frequency are used. FRD2A is called to build $[Q^R]$ and $[Q^I]$, FRD2B is called to add the input h-matrix together (multiplied by their proper constants), FRD2C is called to perform the solve, and FRD2D is called to add the solution to a scratch file during the loop. The loop then repeats for each frequency input. If the number of frequencies is one or the number of loads is one, then the processing is complete. If this is not the case, then FRD2E is called to build UHVF in load frequency order.

If space permits, the NASTRAN core file is used for MHH, BHH, KHH, Q_{HIL} and PHF.

4.151.8 Subroutines

Utility subroutines MINTRP, SADD, INCØRE, CFBSØR and CFACTR may be used.

4.151.8.1 Subroutine Name: FRRD2

1. Entry Point: FRRD2
2. Purpose: Module driver for FRRD2.
3. Calling Sequence: CALL FRRD2
4. Method: See discussion above. Open core is at /FRRD2X/ .

4.151.8.2 Subroutine Name: FRD2A

1. Entry Point: FRD2A
2. Purpose: Find the proper columns of Q_{HIL} and split into two real matrices.
3. Calling Sequence: CALL FRD2A(NQHL,QHR,QNI,IN,NFREQ)
NQHL - GINØ file number for Q_{HIL} matrix
QHR - GINØ file number for real matrix output
QHI - GINØ file number for imaginary matrix output
IH - column six of output matrix
NFREQ - frequency number wanted.

FUNCTIONAL MODULE FRRD2 (FREQUENCY RESPONSE - WITH AEROELASTIC)

4. Method: NQHL is skipped up to the proper frequency and then a column is unpacked into core. This column is split into two $IH \times IH$ matrices with the real part on QHR and the imaginary part on QHI. Open core is at /FRD2AX/ .

4.151.8.3 Subroutine Name: FRD2B

1. Entry Point: FRD2B
2. Purpose: Add five matrices
3. Calling Sequence: CALL FRD2B(A,ALP,B,BET,C,GAM,D,DEL,E,EPS,ØUT)

A, B, C, D, and E are GINØ file numbers for the matrix to add
ØUT is the GINØ file number for the result

ALP, BET, GAM, DEL and EPS are the constants to multiply the matrix by - Complex
4. Method: /SADDX/ is set up and SADD is called. The result is always complex. Open core at /FRD2BX/ .

4.151.8.4 Subroutine Name: FRD2C

1. Entry Point: FRD2C
2. Purpose: Solve $[A][X] = [B]$
3. Calling Sequence: CALL FRD2C(A,B,X,SCR1,SCR2,SCR3,SCR4,SCR5,NLØAD,NFREQ)

A,B,X,SCR1,SCR2,SCR3,SCR4 and SCR5 are GINØ file numbers
NLØAD - number of loads
NFREQ - frequency number wanted
4. Method: FRD2C has two methods of solving the problem. If enough core is available, an incore solve is used (subroutine INCØRE); if the problem is too big, then CFACTR and CFBSØR are used. Open core is at /FRD2CX/ . The B matrix is PHF so the proper columns of it must be found for the loads at the frequency wanted.

4.151.8.5 Subroutine Name: FRD2D

1. Entry Point: FRD2D
2. Purpose: Build columns on to the end of an output matrix.
3. Calling Sequence: CALL FRD2D(IN,ØUT,IP)

IN - input matrix GINØ number
ØUT - output matrix GINØ number

FUNCTIONAL MODULE DESCRIPTIONS

IP - call count - 0 = first add

4. Method: CYCT2B is used to copy column from IN to end of ØUT. Open core at /FRD2DX/ .

4.151.8.6 Subroutine Name: FRD2E

1. Entry Point: FRD2E
2. Purpose: Reorder displacements
3. Calling Sequence: CALL FRD2E(IN,ØUT,NLØAD,NFREQ)
IN - GINØ file number of displacement sorted by Frequency/Load
ØUT - GINØ file number of displacement sorted by Load/Frequency
NLØAD - number of loads
NFREQ - number of frequencies solved.

4.151.8.7 Subroutine Name: FRD2I

1. Entry Point: FRD2I
2. Purpose: Build an aero interpolated matrix load or frequency
3. Calling Sequence: CALL FRD2I(FL,NFREQ,NCØRE,QHHL,QHIL,SCR1,SCR2,SCR3,NRØW)
QHHL,QHIL,SCR1,SCR2,SCR3 are GINØ file numbers
FL - list of frequencies
NFREQ - number of frequencies in FL
NCØRE - length of open core starting at FL(1)
NRØW - h size - output
4. Method: Subroutine MINTRP is used to make the interpolated matrix. The frequencies are changed to reduced frequencies by multiplying each frequency by $2\pi BØV$. These are used as the independent list for MINTRP. The independent list is made from QHHL by picking columns which are closest to MACH. The imaginary parts of these columns are divided by the independent reduced frequency (from QHHL header). The linear spline option of MINTRP is then used to build QHIL.

4.151.9 Design Requirements

Up to 9 scratch files may be used. Each subroutine has its own open core. COMMON/FRRDST/ is used to keep up to 150 frequencies for use during the loop.

FUNCTIONAL MODULE FRRD2 (FREQUENCY RESPONSE - WITH AEROELASTIC)

4.151.10 Diagnostic Messages

FRRD2 may issue 3002, 3003, 3008 and 2271.

FUNCTIONAL MODULE ADR (Aerodynamic Data Recovery)

4.152 FUNCTIONAL MODULE ADR (Aerodynamic Data Recovery)

4.152.1 Entry Point: ADR

4.152.2 Purpose

ADR builds a matrix of aerodynamic forces per frequency for each aerodynamic point. The data is output for a user selected set.

4.152.3 DMAP Calling Sequence

Flutter -

ADR CPHIH1,CASEZZ,QKHL,CLAMAL1,SPLINE,SILA,USETA/PKF/V,N,BØV/C,Y,MACH/C,N,FLUTTER \$

Dynamics -

ADR UHVT1,CASECC,QKHL,TØL1,SPLINE,SILA,USETA/PKF/V,N,BØV/C,Y,MACH/V,N,APP \$

4.152.4 Input Data Blocks

CPHIH1, UHVT1 - Complex modal displacements matrix - h-set

CASEZZ, CASECC- Case Control Data Table

QKHL - Matrix which is a list of matrices for various Mach-reduced reference pairs

CLAMAL1 - Eigenvalue table in ØFP format

TØL1 - List of frequencies

SPLINE - Locate table with k-point triplets

SILA - Scalar Index List with aero points

USETA - Table of set mask for D.O.F.

Note: None of the data blocks may be purged if an AERØF request exists.

4.152.5 Output Data Blocks

PKF - Matrix of forces k-set per frequency

Note: PKF cannot be purged.

FUNCTIONAL MODULE ADR (Aerodynamic Data Recovery)

4.152.6 Parameters

BØV - Conversion factor from circular frequency to reduced frequency (k) - real - input - no default.

MACH - Mach number to select QKHL matrix - real - input - default = 0.0.

APP - Solution - BCD - input - no default.

FUNCTIONAL MODULE ADR (AERODYNAMIC DATA RECOVERY)

THIS PAGE HAS BEEN LEFT BLANK INTENTIONALLY.

MODULE FUNCTIONAL DESCRIPTIONS

4.152.7 Method

Subroutine ADR checks for the existence of an AERØF requests; if one does not exist, ADR returns. If a request exists, ADR checks to see if eigenvalues or frequencies were input. If frequencies were input, they are read into memory from Table TØL1. If eigenvalues were input, the frequencies are found from the eigenvalue table CLAMAL1 and put in memory and subroutine ADRI is then called to build an interpolated matrix based on these frequencies.

Subroutine ADRI converts the frequencies to k's by multiplying by $BØV*2\pi$, which makes the dependent list to send to MINTRP. The independent list for MINTRP is built from the header record of QKHL, using the k's and the matrix associated with the Mach number closest to MACH. The imaginary parts of QKHL are divided by the independent frequencies and the resulting matrix and frequency lists are sent to MINTRP. MINTRP builds a set of interpolated matrices on SCRI and ADRI returns.

ADR now builds each column PKF by multiplying each modal displacement vector by its approximate interpolated matrix (imaginary terms multiplied back by frequency). ADR then calls ADRPRT to print the output for the requested set.

ADRPRT has the frequency list in core, and adds the SPLINE triplets (external ID, SILGA points, row position in PKF). The SILGA pointers are converted to SILA pointers and the SILA associated with aero points is read into core. Then the USETA mask for aero points is read into core, and a consistent set of pointers between them is set up. CASECC and PKF are open, then ADRPRT loops over CASECC records (Loads), over frequencies, and over the points in the requested set.

A CASECC record is read into memory and the set requested found, then a column of PKF is unpacked into memory. Then, for each point requested, its SILA is found. Its SILA points to the USETA and the six D.O.F. associated with it are searched to see which components belong to the k-set. As k-set components are found, the PKF row position is found and the point is output with the forces under their proper component.

4.152.8 Subroutines

4.152.8.1 Subroutine Name: ADR

1. Entry Point: ADR
2. Purpose: Module driver for ADR module.

FUNCTIONAL MODULE ADR (Aerodynamic Data Recovery)

3. Calling Sequence: CALL ADR
4. Method: See method description above.

4.152.8.2 Subroutine Name: ADRI

1. Entry Point: ADRI
2. Purpose: Build interpolated matrix for ADR
3. Calling Sequence: CALL ADRI(FL,NFREQ,NCØRE,QKHL,ØUT,SCR2,SCR3,SCR4,NRØW,NCØL,NØGØ)
FL - independent frequency list - real - input
NFREQ - number of frequencies - integer - input
NCØRE - total core available in ADRX - integer - input
QKHL - GINØ file of input
ØUT - GINØ file for output matrix
SCR2-4- three GINØ scratch files
NRØW - rows of resulting interpolated submatrix - integer - output
NCØL - columns of resulting interpolated submatrix - integer - output
NØGØ - set nonzero if ADRI fails - integer - output
4. Method: See method description above.

4.152.8.3 Subroutine Name: ADRPRT

1. Entry Point: ADRPRT
2. Purpose: Print aero force output request
3. Calling Sequence: CALL ADRPRT(CASECC,PKF,SPLINE,SILA,USETA,FL,NFREQ,NCØRE,NLØAD)
CASECC,PKF,SPLINE,SILA,USETA are GINØ file names of input files - integers - input
FL - frequency list - real - input
NFREQ - number of frequencies - integer - input
NCØRE - total core in common ADRX
NLØAD - number of loads (records on CASECC)
4. Method: See method descriptions above.

MODULE FUNCTIONAL DESCRIPTIONS

4.152.9 Design Requirements

4.152.9.1 Allocation of Open Core

Common /ADRX/ is used by all three subroutines with the frequency list being kept on top.

ADR	ADRI	ADRPRT
Freq. List	Freq. List *2	Freq. List
Column of QKH interpolated	Ind. Freq. List *2	SPLINE triplets
Column of Displacements	MINTRP SPACE	SILA
Column of PKF		USETA
FREE		CASECC
Buffer		Column of PKF
Buffer		FREE
Buffer		Buffer
		Buffer

4.152.9.2 Error Messages

3003, 3007, 3008 and 2271 are messages causing ADR to stop processing, but are not NASTRAN fatal errors. If B0V = 0.0 for flutter analysis, ADR will stop processing.

FUNCTIONAL MODULE GUST (Compute Aerodynamic Loads due to Gusts)

4.153 FUNCTIONAL MODULE GUST (Compute Aerodynamic Loads due to Gusts)

4.153.1 Entry Point: GUST

4.153.2 Purpose

The purpose of this module is to compute loads for aerodynamic analysis, which are associated with aerodynamic flow. Often the shape (time dependence) of the gust is unknown, except for certain statistical information, e.g., power spectral density and RMS value. In these cases the GUST module must create frequency dependent loads. Sometimes the gust shape is specified as a function of time, which will be analyzed by Fourier transform techniques. Then the frequency dependent loads are calculated by Fourier transform.

The value of the load is calculated from the downwash distribution. The calculation involves the aerodynamic formulation. For all methods (except strip theory) the downwash is a part of the aerodynamic theory used in the AMG-AMP modules. The downwash is associated with the j -set, which corresponds to the $[A_{jj}]$ matrix. The loads are computed from the downwash using aerodynamic matrices.

The downwash to be provided comes from a simple model of the atmosphere. The velocity is vertical (in the z -direction of the aerodynamic coordinate system), and appears (to an observer) in the airplane coordinates to sweep back toward the $+x$ direction. This implies that the downwash vector has two properties:

- a. It is proportional to the cosine of the dihedral angle for any panel.
- b. There is a time delay of amount X/U for the arrival at any point. (X is a streamwise coordinate.)

We must allow for the possibility of multiple loading conditions. In the case control deck, these will be specified in separate subcases. The GUST module must (as the load generator FRLG) process adjacent subcases with the same DMIG, constraint, etc., selections. (Note that the aero-elastic dynamic rigid format has no looping.)

The gust loads may appear in conjunction with other loads. One function of the GUST module is to add the loads where appropriate.

MODULE FUNCTIONAL DESCRIPTIONS

4.153.3 DMAP Calling Sequence

GUST CASECC,DLT,FRL,DIT,QHJL,,,ACPT,CSTMA,PHF1/PHF/V,N,NØGUST/V,N,BØV/C,Y,MACH/C,Y,Q \$

4.153.4 Input Data Blocks

CASECC - Case Control Data Table

DLT - Dynamic Loads Table

FRL - Frequency Response List

DIT - Direct Input Tables (and GUST cards)

QHJL - Aerodynamic Interpolation Matrix

ACPT - Information pertaining to each independent group of aerodynamic elements

CSTMA - Coordinate System Transformation Matrices - Aerodynamics

PHF1 - Loads on the modal degrees of freedom due to non-gust effects

Notes: (1) If DIT is purged, GUST will return (setting NØGUST = -1)

(2) CSTMA may be purged.

4.153.5 Output Data Blocks

PHF - Total loads on modal degrees of freedom

4.153.6 Parameters

NØGUST - Integer, output, no default. If no gust loads exist, NØGUST=-1, otherwise NØGUST=1.

BØV - Real, input, default = 0.0

MACH - Real, input, default = 0.0

Q - Real, input, default = 0.0

4.153.7 Method

The task is divided into two basic steps:

- (a) Compute (using code related to the theory) the downwash-lag matrix. Combine this with the results of AMP to prepare the loads due to a delta function.
- (b) Compute the frequency dependent loads (from RLØAD or TLØAD data) for each subcase. Multiply each column of step (a) by the frequency dependent load. Add to the input PHF1 and write on the output.

FUNCTIONAL MODULE GUST (COMPUTE AERODYNAMIC LOADS DUE TO GUSTS)

The formal algebra is as follows:

1. Compute a load function matrix $PP(\omega)$ for the RLØAD/TLØAD cards selected on the GUST cards. Prepare a list of WG, XO, and V for each subcase. This is done by GUST1, which rewrites CASECC and calls FRDD1A to obtain $PP(\omega)$ and FØL (a list of frequencies ($\omega = 2\pi f$)).
2. Compute the downwash matrix

$$[W_j(\omega)] = [W_{j\omega}] = [\cos \gamma_j e^{-i\omega_i(X_j - X_0)/V}] \quad (1)$$

This is computed by GUST2. There is one row for each aerodynamic element j , and one column for each frequency ω_i .

3. QHJL is interpolated to the ω_i 's of the FØL, using the reduced frequencies $k_i = BØV \omega_i$. In the case of multiple Mach values, only the ones nearest MACH are selected. The result $Q_{jh}(\omega)$ is stored with all of the elements of the matrix for a given ω_i packed into a single column. The number of rows is $j \times h$. This is accomplished via a call to ADRI, which in turn calls MINTRP for the interpolation.
4. PDEL(ω) is computed as

$$[PDEL(\omega)] = [Q_{jh}(\omega)][W_j(\omega)] \quad (2)$$

Note that the interpolated $Q_{jh}(\omega)$ must have its imaginary parts multiplied by k_i .

$$[PHF2(\omega)] = [PDEL(\omega)] * Q * WG * [PP(\omega)], \quad (3)$$

where WG is a function of subcase. This is computed by GUST3. Finally,

$$[PHF] = [PHF2] + [PHF1] \quad (4)$$

via a call to SSG2C.

MODULE FUNCTIONAL DESCRIPTIONS

4.153.8 Subroutines

Utility subroutines ZEROC, MINTRP, SSG2C, ADD, MPYAD, DMPFIL, PRELOC/LOCATE and GMMATC are called by GUST. In addition GUST calls FRRD1A and ADRI.

4.153.8.1 Subroutine Name: GUST1

1. Entry Point: GUST1
2. Purpose: To compute FOL, GUSTL and PPW
3. Calling Sequence: CALL GUST1(CASECC,DIT,DLT,FRL,PPW,FOL,GUSTL,NFREQ,NLOAD,X0,V,NOGUST,CASNEW)

CASECC, DIT, DLT, FRL, PPW, FOL, GUSTL and CASNEW are the GINØ file number of their respective data blocks

NFREQ - number of frequencies - output

NLOAD - number of NLOADs (subcases in CASECC) - output

X0, V - taken from the GUST cards

NOGUST - set to -1 if no GUST loads are selected

4. Method: GUST1 puts all GUST cards in memory (5 words per card). It then rewrites CASECC onto CASNEW replacing the DLOAD ID with the load selected on the selected GUST card. GUSTL is generated in order with CASECC with an image of the selected GUST card (SID, DLOAD ID, WG, X0, V). If no GUST card is selected, WG is set to 0. GUST1 then calls FRRD1A to generate PPW and FOL.

4.153.8.2 Subroutine Name: GUST2

1. Entry Point: GUST2
2. Purpose: Compute the downwash matrix $WJ(\omega)$ for GUST
3. Calling Sequence: CALL GUST2(FOL,WJ,ACPT,X0,V,CSTMA,QHJL)

FOL - GINØ number for Frequency List Table

WJ - GINØ number for output matrix

ACPT - GINØ number for ACPT table

X0 - Reference point from GUST card

V - Velocity of airplane from GUST card

CSTMA - GINØ number for CSTMA table

QHJL - GINØ number for QHJL matrix list

FUNCTIONAL MODULE GUST (COMPUTE AERODYNAMIC LOADS DUE TO GUSTS)

4. Method: The input frequencies from FØL are read into memory and multiplied by 2π to get omega's (ω). Two areas of core (Length J) are then set aside to hold AM (Scalar multiplier) and XM (J-point locations). These areas are filled in by reading the ACPT table and branching on the method of the records. AM is defined as follows:

$$AM = \begin{cases} CG & \text{(from ACPT for all panels)} \\ 1 & \text{for all Z slender elements} \\ 0 & \text{for all other elements} \end{cases}$$

XM is defined as follows:

$$XM = \begin{cases} XIC+.5*DELX & \text{for Doublet Lattice without body panels (From ACPT)} \\ X & \text{from ACPT for Doublet Lattice with bodies} \\ .5*(XIS1+XIS2) & \text{for Z slender elements} \\ 0 & \text{for all else} \end{cases}$$

The matrix to build is defined as:

$$WJ_{mn} = AM * e^{-i(\omega_n(XM-XØ))/V}$$

m = J points - Rows

n = omega's - columns

The following defines a complex matrix:

$$\text{Real part} = \cos(\omega_n(XM-XØ)/V) * AM$$

$$\text{Imag part} = -\sin(\omega_n(XM-XØ)/V) * AM$$

The matrix is built with two DØ loops: the outer DØ loop over n (omega) and the inner DØ loop over m (J points). ZBLPKI is called as each term is built.

4.153.8.3 Subroutine Name: GUST3

1. Entry Point: GUST3
2. Purpose: To compute PHF2
3. Calling Sequence: CALL GUST3(QHJK,WJ,PPW,GUSTL,PDEL,PHF2,Q,NFREQ,NLØAD,NRØWJ,NCØLW)

QHJK, WJ, PPW, GUSTL, PDEL, and PHF2 are the GINØ file numbers of their respective data blocks.

Q - real - input supplied by user.

NFREQ, NLØAD are as output from GUST1.

NRØWJ, NCØLW are as output from ADRI.

MODULE FUNCTIONAL DESCRIPTIONS

4. Method: A column of QHJK is unpacked into memory. The imaginary part is divided by K_i . A column of WJ is unpacked into memory. GMMATC is called to multiply $[Q_{hj}(\omega)][W_j(\omega)]$ to form a column of PDEL which is packed out. For each load WG is extracted from GUSTL. PDEL is rewound. For each frequency a column of PDEL is unpacked. A term (column) from PPW is unpacked. Each term of PDEL is multiplied by $Q*WG*PP(\omega)$ and the result is packed onto PHF2.

4.153.9 Design Requirements

GUST uses seven scratch files as follows:

GUST1 generates PP(ω) on SCR1, generates FØL on SCR2, images of the GUST code on SCR4, a new CASECC on SCR3 (for use by FRRD1A)

GUST2 reads SCR2 and writes WJ on SCR3

ADRI writes QJH(ω) and SCR2 and uses SCR5, SCR6 and SCR7 as scratch files

GUST3 reads SCR2, SCR3, SCR1 and SCR4 and writes PDEL on SCR5 and PHIF2 on SCR6

Open core for GUST is GUSTX

Open core for GUST1 is FRRD1A

Open core for GUST2 is GUST2X

Open core for GUST3 is GUSTX

4.153.10 Diagnostic Messages

GUST may issue 3001, 3002, 3003, 3007, 3008.

FUNCTIONAL MODULE IFT (INVERSE FOURIER TRANSFORM)

4.154 FUNCTIONAL MODULE IFT (INVERSE FOURIER TRANSFORM)

4.154.1 Entry Point: IFT

4.154.2 Purpose:

The purpose of this module is to obtain solutions versus time for aeroelastic problems, for which the aerodynamic forces are only known as functions of frequency.

The definition of the inverse transform is

$$u_i(t) = \frac{1}{\pi} \int_0^{\infty} \text{Re}[\tilde{u}_i(\omega) e^{i\omega t}] d\omega \quad (1)$$

where $\tilde{u}(\omega)$ is considered to be a known input (frequency response), and $u(t)$ is the output (time response). Several approximations will be used for evaluation of Equation 1. Special code is to be used for equal $\Delta\omega$'s, which can result in savings of computation times. Special code will be introduced to provide for a cubic spline fit for $\tilde{u}(\omega)$ between the data points.

4.154.3 DMAP Calling Sequence

IFT UHVF,CASECC,TRL,FØL/UHVT,TØL/C,Y,IFTM

4.154.4 Input Data Blocks

UHVF--Complex displacements - h - set

CASECC--Case Control

TRL--Transient Response List

FØL--Frequency Output List

No input data block may be purged.

4.154.5 Output Data Blocks

UHVT--Real displacements - h - set

(Note that velocity and acceleration are zero)

TØL--Time Output List

No output data block may be purged.

4.154.6 Parameters

IFTM--Integer, input, default=1

MODULE FUNCTIONAL DESCRIPTIONS

Method of integration:

0=rectangular

1=trapezoidal

2=spline

4.154.7 Method

4.154.7.1 General Approach

The integral (1) will be expressed as a finite sum. The input data block UHVF has one row for each mode, and a column (complex) for each frequency. The output UHVT will have the same number of rows, but with one column for each time step. Frequencies and times have been selected via standard FREQi and TSTEP Bulk Data cards. Each row of the input matrix is transformed into a row of the output. A single precision version will be written. The basic method is to put the entire UHVF data block in core and to create one column of UHVT at a time.

The basic sum is given by

$$u_i(t_m) = \frac{1}{\pi} \sum_{n=1}^{NFREQ} \operatorname{Re}[(C_{mn} \cdot \tilde{u}_i(\omega_n) + D_{mn} \cdot \tilde{u}_i''(\omega_n)) e^{i\omega_n t_m}] \quad (2)$$

where,

$m = 1, 2, \dots, NTIME$

$\omega_n = 2\pi f_n$ (f = frequency)

$\tilde{u}_i(\omega_n)$ = The complex quantity in the i th row and n th column of UHVF.

$\tilde{u}_i''(\omega_n)$ = The second derivative, found by the method of splines.

$\tilde{u}_i(t_m)$ = The real quantity output on UHVT.

Re = "Real part of." Note for efficiency,

$\operatorname{Re} C_u = \operatorname{Re} C \cdot \operatorname{Re} u - \operatorname{Im} C \cdot \operatorname{Im} u$

$$C_{mn} = \frac{\omega_n - \omega_{n-1}}{2} E_2(-i t_m (\omega_n - \omega_{n-1})) + \frac{\omega_{n+1} - \omega_n}{2} E_2(+i t_m (\omega_{n+1} - \omega_n)) \quad (3)$$

FUNCTIONAL MODULE IFT (INVERSE FOURIER TRANSFORM)

$$D_{mn} = - \frac{(\omega_n - \omega_{n-1})^3}{24} G(-it_m(\omega_n - \omega_{n-1})) - \frac{(\omega_{n+1} - \omega_n)^3}{24} G(+it_m(\omega_{n+1} - \omega_n)) \quad (4)$$

The functions E_2 and G will be discussed later.

For the first ω , only the second term of C and D is used; and, for the last ω , only the first term is used.

4.154.7.2 Values for C and D

If the frequency data has equal $\Delta\omega$'s, $|(\omega_n - \omega_{n-1}) - (\omega_2 - \omega_1)| < 10^{-6}(\omega_2 - \omega_1)$, then the special equal frequency intervals methods are to be used. Otherwise, the general case will be assumed. In the sum, the following assumptions can be made.

Parameter IFTM	Equal $\Delta\omega, \Delta t$		General	
	0	$E_2 = 1$ (for first ω if $\omega=0$, $E_2 = \frac{1}{2}$). $D_{mn} = 0$	1	$E_2 = 1$ ($\frac{1}{2}$ first, last) $D_{mn} = 0$
	1	C_{mn} = function of m , and special for first, last ω . $D_{mn} = 0$	2	$D_{mn} = 0$ C_{mn} -- See Eq. 3
	2	C_{mn}, D_{mn} are functions of m only. Special for first and last ω .		See Equations 3 and 4.

For the equal $\Delta\omega$ case, we will force $\omega_n t_m$ to be an integer multiple of $(2\pi/N)$. Define integer $N = 2\pi/(\Delta\omega \cdot \Delta t)$, rounded up. Then replace the user's Δt by $\Delta t = 2\pi/N\Delta\omega$.

In the equal $\Delta\omega$ case, $\omega_n t_m$ is given by

$$\omega_n t_m = nm\Delta\omega\Delta t = 2\pi\left(\frac{nm}{N}\right) \quad (5)$$

A table for $k = 0, (N-1)$ will be made of

$$\begin{aligned} C_k &= \cos(2\pi k/N) \\ S_k &= \sin(2\pi k/N) \end{aligned} \quad (6)$$

MODULE FUNCTIONAL DESCRIPTIONS

Then,

$$e^{i\omega_n t_m} = C_k + iS_k \quad (7)$$

where,

$$k = \text{Mod}(nm, N) \quad (8)$$

The table can be created with recurrence, such as in module CYCT1. For unequal $\Delta\omega$, then

$$e^{i\omega_n t_m} = \cos(\omega_n t_m) + i \sin(\omega_n t_m) \quad (9)$$

where the COS and SIN subroutines will be called.

The functions E_2 and G will have special formulas depending upon the size of their argument.

$$E_2(i\theta) = \left(\frac{1 - \cos\theta}{\frac{1}{2}\theta^2} \right) + i \left(\frac{\theta - \sin\theta}{\frac{1}{2}\theta^2} \right) \quad |\theta| \geq \theta_0 \quad (10)$$

$$= \left(1 - \frac{\theta^2}{3 \cdot 4} + \frac{\theta^4}{3 \cdot 4 \cdot 5 \cdot 6} - + \dots \right) + i \left(\frac{\theta}{3} - \frac{\theta^3}{3 \cdot 4 \cdot 5} + \frac{\theta^5}{3 \cdot 4 \cdot 5 \cdot 6 \cdot 7} - + \dots \right) \quad (11)$$

$$|\theta| < \theta_0$$

$$G(i\theta) = 2E_2(i\theta) - E_4(i\theta) \quad (12)$$

$$E_4(i\theta) = \frac{\frac{\theta^2}{2} - 1 + \cos\theta}{\frac{\theta^4}{24}} + i \frac{\frac{\theta^3}{6} - \theta + \sin\theta}{\frac{\theta^4}{24}} \quad |\theta| \geq \theta_0 \quad (13)$$

$$= \left(1 - \frac{\theta^2}{5 \cdot 6} + \frac{\theta^4}{5 \cdot 6 \cdot 7 \cdot 8} - + \dots \right) + i \left(\frac{\theta}{5} - \frac{\theta^3}{5 \cdot 6 \cdot 7} + \frac{\theta^5}{5 \cdot 6 \cdot 7 \cdot 8 \cdot 9} - + \dots \right) \quad (14)$$

$$|\theta| < \theta_0$$

$\theta_0 = 0.1$. The alternating series is continued until the last term computed is less than 10^{-9} .

When IFTM = 2, then a spline fit must be made to solve for \bar{u} ". This will be done by a backward and forward pass over the frequencies ω . The same method is used for equal or unequal $\Delta\omega$. On the initializing backward pass, compute real A_n ($n = \text{NFREQ} - 1, \dots, 2$).

$$A_{\text{NFREQ}} = 0.$$

FUNCTIONAL MODULE IFT (INVERSE FOURIER TRANSFORM)

$$A_n = \frac{(\omega_n - \omega_{n-1})}{2(\omega_{n+1} - \omega_{n-1}) - (\omega_{n+1} - \omega_n)A_{n+1}} \quad (15)$$

For each row, do a backward and forward pass. The B_n and $\tilde{u}''(\omega_n)$ can be stored in the same locations. The backward pass:

$$B_{NFREQ} = 0.$$

$$B_n = \frac{6\left(\frac{\tilde{u}_{n+1} - \tilde{u}_n}{\omega_{n+1} - \omega_n} - \frac{\tilde{u}_n - \tilde{u}_{n-1}}{\omega_n - \omega_{n-1}}\right) - (\omega_{n+1} - \omega_n)A_{n+1} \cdot B_{n+1}}{\omega_n - \omega_{n-1}} \quad (16)$$

$$n = (NFREQ-1), \dots, 3, 2.$$

(real and imag.)

The forward pass:

$$\tilde{u}''(\omega_1) = \begin{cases} \frac{6 \frac{\tilde{u}_2 - \tilde{u}_1}{\omega_2 - \omega_1} - (\omega_2 - \omega_1)A_2 B_2}{6\omega_1 + (\omega_2 - \omega_1)(2 - A_2)} & \text{Real Part} \\ 0 & \text{Imag. Part} \end{cases} \quad (17)$$

Then continue for $n = 2, \dots, NFREQ$.

$$\tilde{u}''(\omega_n) = A_n(B_n - \tilde{u}''_{n-1})$$

4.154.8 Subroutines

4.154.8.1 Subroutine Name: IFTE2

1. Entry Point: IFTE2.
 2. Purpose: To compute $E_2(i\theta)$ as given in equations 10 and 11.
 3. Calling sequence: CALL IFTE2(THA,RP,CP).
- THA--Real,input,value of Theta (θ)
 RP --Real,output,real part of E_2
 CP --Real,output,imaginary part of E_2

4. Method

$$\theta_0 = 1.E-9$$

The series is expanded to a maximum of 50 terms.

4.154.8.2 Subroutine Name: IFTE4

MODULE FUNCTIONAL DESCRIPTIONS

1. Entry Point: IFTE4.
2. Purpose: To compute E_4 according to equations 13 and 14.
3. Calling Sequence:

CALL IFTE4(THA,RP,CP)

THA--Real,input,value of Theta (θ)

RP --Real,output,real part of E_4

CP --Real,output,imaginary part of E_4

4. Method

$\theta_0 = 1.E-9$

The series is expanded to a maximum of 50 terms

4.154.8.3 Subroutine Name: IFTG

1. Entry Point: IFTG
2. Purpose: To compute $G(\theta)$ according to equation 12.
3. Calling Sequence:

CALL IFTG(THA,RP,CP)

THA--Real,input,value of Theta (θ)

RP --Real,output,real part of G

CP --Real,output,imaginary part of G

4. Method

IFTG calls IFTE2 and IFTE4.

4.154.8.4 Subroutine Name: ZERØC

1. Entry Point: ZERØC
2. Purpose: To Zero one area of core.
3. Calling Sequence:

CALL ZERØC(Z,LZ)

Z--Array to be zeroed

LZ--Length of array to be zeroed.

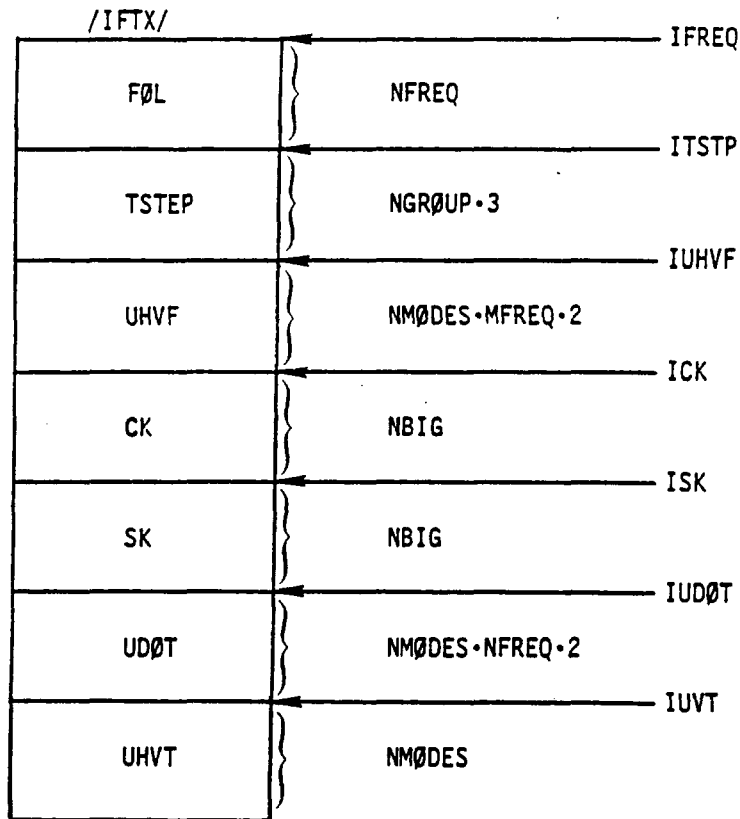
4.154.9.9 Design Requirements

IFT uses no scratch files

Open core is defined at /IFTX/

FUNCTIONAL MODULE IFT (INVERSE FOURIER TRANSFORM)

Open core is used as follows:



IFT uses two GINØ buffers. UDØT is only allocated if IFTM = 2.

4.154.10 Diagnostic Messages

IFT may issue 3001, 3002, 3008.

MODULE FUNCTIONAL DESCRIPTIONS

4.155 MATRIX MODULE SDCMPS (MATRIX DECOMPOSITION - SYMMETRIC WITH CONTROLS AND DIAGNOSTICS)

4.155.1 Entry Point: DDCMPS

4.155.2 Purpose

To decompose a symmetric real matrix $[A]$ into lower $[L]$ and upper $[U]$ triangular factors. Badly conditioned matrix columns encountered during the process are identified with external identification numbers. Various user exit controls for error conditions are available.

4.155.3 DMAP Calling Sequence

SDCMPS USET,GPL,SIL,A / L,U / V,Y,DIAGCK / V,Y,DIAGET / V,Y,PDEFCK / V,N,SING / V,Y,SET /
V,Y,CHØLSKY / V,N,DET / V,N,PØWER \$

4.155.4 Input Data Blocks

USET - Displacement Set Definition Table
GPL - Grid Point List
SIL - Scalar Index List
A - A symmetric real matrix (may not be purged)

Note: Error conditions are identified only by column number if USET, GPL, or SIL are purged.

4.155.5 Output Data Blocks

L - Lower triangular factor of $[A]$
U - Nonstandard upper triangular factor of $[A]$

4.155.6 Parameters

DIAGCK - Input, integer, default = 0. Diagonal singularity exit flag.

0 = nonfatal messages are issued for all $e_s > T_s$ (see DIAGET)

> 0 = maximum number of "fatal" messages for $e_s > T_s$ or for nonconservative columns before aborting decomposition prior to completion

< 0 = no check of e_s (see Method)

DIAGET - Input, integer, default = 20. Diagonal singularity check tolerance, $T_s = 2^{-\text{DIAGET}}$

PDEFCK - Input, integer, default = 0. Positive definiteness exit flag. Decomposed matrix diagonal terms D_{ii} are checked.

MODULE FUNCTIONAL DESCRIPTIONS

- 0 = nonfatal messages are issued for $D_{ij} < 0.0$. Fatal messages occur after completing decomposition for $D_{ij} = 0.0$.
- > 0 = a maximum of PDEFCK "fatal" messages occur for $D_{ij} \leq 0.0$ before aborting decomposition prior to completion.
- < 0 = no check is made for $D_{ij} < 0.0$. If $D_{ij} \equiv 0.0$, -PDEFCK equals the number of "fatal" messages issued before aborting decomposition prior to completion.
- SING - Output, integer, no default. SING set to -1 if [A] is singular, 0 if not positive definite and 1 otherwise.
- SET - Input, BCD, default is L. The displacement set defining the size of the A matrix.
- CHOLSKY - Input, integer, default = 0. 1, use Cholesky decomposition. 0, do not use Cholesky decomposition.
- DET - Output, real single precision, default = 0. The scaled value of the determinant of [A].
- PPOWER - Output, integer, default = 0. Integer PPOWER of ten by which DET should be multiplied to obtain the determinant of [A].

4.155.7 Method

Subroutine single precision SDCMPS or double precision SDCMPD is called by DDCMPS to decompose the real symmetric matrix A. The checks made during decomposition in these routines are singular matrix and negative definite matrix tests. If any diagonal does not pass these tests, the column number is written onto a scratch file along with the associated error. Upon return to DDCMPS if the error flag is set, subroutine SDCMM is called to write the external identification number and component associated with the column. If this is a fatal error, parameter SING is appropriately set before returning.

The algorithm for SDCMPS is the same as that used in SDCOMP except for the conditioning checks. Prior to either check described above, a pass is made over the A matrix to determine the bandwidth. The core and coefficients for M_B , M_C , I, and P in the timing equations are printed if DIAG 11 is on. If any column is null, the program writes these column numbers onto the scratch file as described above, completes the pass, but fatal exits prior to doing the actual decomposition.

MATRIX MODULE SDCMPS (MATRIX DECOMPOSITION - SYMMETRIC WITH CONTROLS AND DIAGNOSTICS)

The bandwidth data and an estimate of the number of calculations to be performed during decomposition and during the forward-backward substitutions (subroutine FBS) are printed prior to decomposition.

The diagonal singularity check, e_s , measures the overall solution matrix singularities. This measure is largest at singularities and is calculated by:

$$e_s = \frac{2^{1-p}}{|D_{ii}/A_{ii}|}$$

where p = Number of bits in the mantissa (machine dependent)

A_{ii} = Diagonal term of A matrix

D_{ii} = Diagonal term of the decomposed matrix

If this exceeds the tolerance $T_s = 2^{-\text{DIAGET}}$ the terms failed the test. Note that $p - \text{DIAGET}$ should be approximately 13 or greater for a valid test. If $D_{ii} > A_{ii}$ the column is labeled as nonconservative.

The positive definite check verifies the decomposed matrix diagonals, D_{ii} , are positive. Note that $D_{ii} = 0.0$ would cause an overflow during decomposition, thus, a 1.0 is substituted, appropriate messages generated, and exit flags (see SDCMQ) set before continuing.

Three scratch files are needed.

4.155.8 Subroutines

Module SDCMPS consists of a main routine DDCMPS and auxiliary subroutines SDCMPR, SDCMPD, SDCMQ, SDCMM, and MATCID. See SDCOMP for a description of the decomposition algorithm of SDCMPR (single precision) and SDCMPD (double precision). These decomposition routines intercept null columns on the input matrix when calculating the B-C-R parameters and test the decomposed matrix diagonals during decomposition. Both routines call SDCMQ to queue the columns in error.

Other fatal messages, such as insufficient time, are queued until SDCMM writes the messages created by SDCMQ.

4.155.8.1 Subroutine Name: SDCMQ

1. Entry Point: SDCMQ

MODULE FUNCTIONAL DESCRIPTIONS

2. Purpose: To write onto a scratch file singularity and positive definite error messages for subsequent processing by SDCMM and to set exit flags.

3. Calling Sequence: CALL SDCMQ (\$n₁,KEY,V1,V,DV1,DV,IC,Z)

n₁ - Nonstandard return if decomposition is to be aborted immediately

KEY - 1 = write onto SCRFIL the column number with a null column on the input matrix found during the prepass on the matrix.

2 = write onto SCRFIL the column number with a zero diagonal on the decomposed matrix.

3 = write onto SCRFIL the column number with a negative diagonal on the decomposed matrix.

4 = write onto SCRFIL the column number that failed the singularity tolerance ($e_s > T_s$).

5 = write onto SCRFIL the column number of a null column found during the decomposition. This message (labeled BAD COLUMN) indicates a system problem.

6 = write onto SCRFIL the column number that is nonconservative (decomposed diagonal greater than 1.001 times the input diagonal).

V, DV - Value of decomposed matrix entry corresponding to KEY. V is single precision, DV is double precision.

V1, DV1 - Value of the corresponding input diagonal

IC - Internal column number

Z - Open core. BUF is relative to Z(1).

COMMON / SDCQ / NERR(2),NØGLEV,BUF,FILSCR,FILCUR,JCURØ,PDEFCK,DIAGCK,IPREC

NERR - Two-word array defining number of errors of each type.

NØGLEV - Error code

0 = no fatal errors

1 = abort after decomp

2 = abort after prepass

3 = abort immediately

BUF - Buffer location relative to Z(1)

MATRIX MODULE SDCMPS (MATRIX DECOMPOSITION - SYMMETRIC WITH CONTROLS AND DIAGNOSTICS)

FILCUR - Current file number using BUF
FILSCR - Scratch file number to queue messages
JCURØ - Read or write status of FILCUR
PDEFCK, DIAGCK - As in input parameters
IPREC - 1 = single precision - use V, V1
 2 = double precision - use DV, DV1

4. Design Requirements: The scratch file, FILSCR, for writing columns in error may share the buffer location, BUF, with another file (FILCUR > 0). If so, the file, FILCUR, is closed without rewind prior to opening the scratch file and then reopened before exiting.

Before returning, the exit condition is checked and a corrected entry for the diagonal placed in V or DV if necessary.

Each error condition on FILSCR has the following format:

<u>Word</u>	<u>Type</u>	<u>Meaning</u>
1	I	Column Number * 10 + KEY
2	R	Value of the input diagonal
3	R	Value of decomposed diagonal

4.155.8.2 Subroutine Name: SDCMM

1. Entry Point: SDCMM
2. Purpose: To print and summarize the data on the scratch file containing the column number and values of the decomposed and input diagonals for matrix columns in error.
3. Calling Sequence: CALL SDCMM (Z,MSET,MSZE,MATRIX,USET,GPL,SIL)
 - Z - Open core. BUFF is relative to Z(1)
 - MSET - BCD name of displacement set matrix belongs to
 - MSZE - Number of columns in matrix
 - MATRIX - GINØ file number of matrix in error
 - USET - GINØ file number of USET data file
 - GPL - GINØ file number of GPL data file
 - SIL - GINØ file number of SIL data file

MODULE FUNCTIONAL DESCRIPTIONS

COMMON / SDCQ / - Same as in subroutine SDCMQ

4. Method: Subroutine MXCID is called to create an external identification number array. This one word/column array contains the external identification number and component number in internal sort. If this array cannot be created then the internal column numbers are used to identify the columns in error.

5. Design Requirements: Two buffers starting at Z(BUF) are required. See MXCID requirements also.

6. Additional Subroutines: EJECT, MXCID.

4.155.8.3 Subroutine Name: MXCID

1. Entry Point: MXCID

2. Purpose: To create an array that can be used to identify columns of a matrix.

3. Calling Sequence: CALL MXCID (\$n₁,Z,SET,MSZE,MCØN,USET,GPL,SIL,BUF1)

n₁ - Nonstandard return if unable to complete the task

Z - Open core

SET - BCD name of displacement set to which matrix belongs.

MSZE - Matrix size

MCØN - Number of words desired per entry (1 minimum)

USET - Displacement Set Definition Table GINØ file number

GPL - Grid Point List GINØ file number

SIL - Scalar Index List GINØ file number

BUF1 - Buffer location for 2 buffers. Defines end of open core.

4. Method: An array is reserved at the start of core for the resultant matrix (length = MSZE * MCØN). The GPL is read into the remaining core. If enough core remains USET and/or SIL are also read into core, otherwise their entries are read one word at a time.

Each internal grid point is examined for members of the requested set. The SIL values determine the number of components the internal point has. These components are then inspected for members of the set. When located, the external ID is taken from the GPL, multiplied by 10 and the component added. Scalar points have a component of zero. This "external ID" is placed in the next available location at the start of open core.

MATRIX MODULE SDCMPS (MATRIX DECOMPOSITION - SYMMETRIC WITH CONTROLS AND DIAGNOSTICS)

If there is insufficient core, purged input files, or the wrong set names are found, the nonstandard return is taken. Messages 3001, 3007, 3008, and 3016 may result.

5. Design Requirements: Two buffers and $MSZE * MCØN + NGP + 7$ words of open core where NGP is the number of grid and scalar points in the model.

4.155.8.4 Subroutine Name: MXCIDS

1. Entry Point: MXCIDS

2. Purpose: To create an array that can be used to identify columns by grid, component, and substructure name. (Yet to be developed.)

3. Calling Sequence: CALL MXCIDS ($n_1, Z, MSET, MSZE, NWDS, USET, NSTRT, SNAM$)

n_1 - Nonstandard return if unable to complete the task

Z - Open core

MSET - BCD name of displacement set to which matrix belongs

MSZE - Matrix size

NWDS - Number of words desired per entry (2 minimum)

USET - Displacement Set Definition Table GINØ file number

NSTRT - On input, pointer to 4 GINO buffers in Z. On output, first word of column identifiers

SNAM - Two-word BCD name of substructure to identify

MODULE FUNCTIONAL DESCRIPTIONS

4.156 FUNCTIONAL MODULE EQMCK

4.156.1 Entry Point: EQMCK

4.156.2 Purpose

EQMCK calculates an overall total of forces and moments on the entire structure to provide an equilibrium check and creates the multipoint constraint force output file.

4.156.3 DMAP Calling Sequence

EQMCK CASECC,EQEXIN,GPL,BGPDT,SIL,USET,KGG,GM, $\left\{ \begin{array}{c} \text{PHIG} \\ \text{UGV} \end{array} \right\}$, $\left\{ \begin{array}{c} \text{LAMA} \\ \text{PGG} \end{array} \right\}$, QG,CSTM / ØQM1 / V,Y,ØPT /
V,Y,GRDPNT / V,N,NSKIP \$

4.156.4 Input Data Blocks

CASECC - Case control data table

EQEXIN - Equivalent external-internal grid point table

GPL - Grid point list table

BGPDT - Basic grid point definition table

SIL - Scalar index list table

USET - Displacement set definition table

KGG - Stiffness matrix - g set

GM - Multipoint constraint transformation matrix - m set

PHIG - Real eigenvector - g set

UGV - Displacement vector matrix - g set

LAMA - Real eigenvalue table

PGG - Static load vector - g set

QG - Single-point constraint force and determinate support force matrix - g set

CSTM - Coordinate system transformation matrix

Note: GM, PGG, QG, and CSTM may be purged.

4.156.5 Output Data Blocks

ØQM1 - Output multipoint constraint force (m set, SØRT1, real)

Note: ØQM may be purged if GM is purged. See also parameter ØPT.

4.156.6 Parameters

\emptyset PT - Input, integer, default = 0. \emptyset PT controls printed output.

- 0 - only create \emptyset QM
- < 0 - calculate equilibrium forces
- > 0 - equilibrium forces and \emptyset QM created

GRDPNT - Input, integer, default = -1. GRDPNT selects the grid point about which equilibrium will be checked. The basic origin is used if GRDPNT is not an external ID of a geometric grid point.

NSKIP - Input, integer, no default. If $NSKIP > 1$, this is a statics problem with $NSKIP$ corresponding to the first CASECC record to use. If $NSKIP \leq 0$, this is a real eigenvalue problem, without DMAP loop capability.

SUBNAM - Input, BCD, default = NONE. Reserved for future use.

4.156.7 Method

All user errors are considered nonfatal. An example is when \emptyset PT = 0 and no MPCs exist in the problem - no execution is required and the module exits. If needed data blocks are required but missing for the selected output, the module exits. A message informs the user of the exit condition.

Overall equilibrium of loads and reactions is calculated from:

1. Directly applied statics loads $\{P_g\}$
2. Forces of single-point constraint and determinant support reactions $\{q_g\}$
3. Forces of multi-point constraints $\{q_m\}$

EQMCK first checks the input files. If GM is purged and \emptyset PT = 0, a message is written and the module exits. If CASECC, EQEXIN, GPL, BGPDT, SIL, or USET is purged the program also exits. If PGG, QG, and GM are purged the module exits.

If GM, UGV, and KGG are available, subroutine EQMCKM is called. Subroutines CALVEC and SSG2A are called to partition KGG.

$$[K_{gg}] \rightarrow \begin{bmatrix} K_{ng} \\ K_{mg} \end{bmatrix}$$

Scratch file 1 is used to store the partitioning vector and scratch file 2 for $[K_{mg}]$.

If parameter NSKIP > 1, NSKIP - 1 columns of UGV and PGG are skipped prior to copying the files to scratch files 3 and 4 respectively in subroutine CURCAS. Thus, further calculations only deal with the subcases in this loop for statics problems.

Using the partitioning vector, the load vector is partitioned to create $[P_m]$ on scratch file 7.

$$[P_g] = \begin{bmatrix} P_n \\ P_m \end{bmatrix}$$

With these matrices as input, the multipoint constraint forces are calculated. Subroutine MPYAD is called to create $[q_m^m]$ on scratch file 5 (using 1 as a scratch file).

$$[q_m^m] = -[P_m] + [K_{mg}][u_g] ,$$

and $[q_m^n]$ on scratch file 6,

$$[q_m^n] = -[G_m]^T [q_m^m]$$

Subroutine SDR1B is called to create $[q_g^m]$ on scratch file 3 if $\emptyset PT \neq 0$ and MPC forces exist.

$$q_g^m = \begin{bmatrix} q_m^n \\ -\frac{q_m^m}{q_m^m} \end{bmatrix}$$

If $\emptyset QM1$ is required, a two-dimensional list of external IDs and components corresponding to the g-size matrix columns is created by subroutine MXCID. The column number is appended and the list sorted on external ID. The CASECC file is read for the current subcases and requested points for each subcase are output on $\emptyset QM1$, unless all constraint forces for the point are zero. Note that only the $S\emptyset RT1$ option is supported. Conical shell and fluid points may be requested by specifying "ALL" on the set requests in Case Control.

If SPC forces exist, subroutine CURCAS is used to create the current subcases on scratch file 5.

The next phase is the equilibrium check, if requested via $\emptyset PT$, as defined by:

$$[S_{T1}] = [D]^T \{ [P_g] + [q_g] + [q_g^m] \} ,$$

MODULE FUNCTIONAL DESCRIPTIONS

where each column is a subcase. This check measures all forces, including the residual forces on the u_ℓ and u_ϕ points. These forces thereby include the effects of matrix reduction, decomposition, grounded scalar springs, and general elements.

Subroutine GPWG1A (see GPWG description) calculates $[D]^T$ as a function of GRDPNT parameter. Scratch file 2 is used. Subroutine EQMCKS is then called.

The equilibrium forces (e.g., $[D]^T[P_g]$) are calculated and output separately and then the sum is output. The format is:

RESULTANT LOADS AT POINT XXX IN BASIC COORDINATE SYSTEM

SUBCASE XX, LOAD XX

-TYPE-	T1	T2	T3	R1	R2	R3
APPLIED	X.XX	X.XX	X.XX	X.XX	X.XX	X.XX
SPCFØRCE	↓	↓	↓	↓	↓	↓
MPCFØRCE						
---TOTAL	↓	↓	↓	↓	↓	↓

SUBCASE XX, LOAD XX

-(repeated for all subcases in the current DMAP loop)-

For real eigenvalue problems (NSKIP < 0) the subheading is:

SUBCASE XX, MODE XX, FREQUENCY X.XX

Subroutine MPYAD calculates the equilibrium forces. Applied loads are saved on scratch file 4, SPC-forces on scratch file 5, and MPC-forces on scratch file 6. Scratch file 1 is available for MPYADs usage. Each column is a 6 x (number of subcases) matrix.

These matrices are read into core. If insufficient core is available, the remaining subcases are ignored (highly unlikely to occur) and a warning message is issued.

For each column CASECC (and LAMA for NSKIP < 0) are read to determine the headings and the output formatted and printed.

4.156.8 Design Requirements

Seven scratch files are needed. Subroutine MPYAD requires four buffers. Open core for EQMCK is at / SSGA2 / and / SSGB2 / . Open core for EQMCKS is at / EQMKS / .

FUNCTIONAL MODULE EQMCK

The core layout for EQMCKM to create the ØQM file is:

/ SSGA2 /

ZZ(2,M) - Row identification data where M is the m-set size (external sort) ZZ(1,i) = external grid ID * 10 + component number ZZ(2,i) = row number
ZZ(2 * M + 1) - 146 words for ØQM odd-numbered records
ZZ(2 * M + 147) - N words of set data for MPCFØRCES
ZZ(2 * M + N + 147) - unpacked column (subcase) of multipoint constraint forces
- unused space -
GINØ buffer - LAMA file. Not required for statics problems
GINØ buffer - MPCFØRCES file
GINØ buffer - ØQM file
GINØ buffer - CASECC file

4.156.9 Subroutines

Calls are made to EQMCKM, EQMCKS, GPWG1A, and EJECT.

4.156.9.1 Subroutine Name: EQMCKM

1. Entry Point: EQMCKM
2. Calling Sequence: CALL EQMCKM

COMMON / EQMK1 / - see Section X.Y.Z.10
3. Purpose: To create an MPC-forces of constraint scratch file, an MPC-forces of constraint output file, and scratch files with only the current subcases for UGV, PGG, and QG.
4. Additional Subroutines: CALCV, EJECT, MPYAD, SDR1B, SSG2A, BISHEL, CURCAS, MXCID

4.156.9.2 Subroutine Name: CURCAS

1. Entry Point: CURCAS
2. Calling Sequence: CALL CURCAS (*,NSKIP,TRL,MCB,ZZ,IBUF)

where

- * - Nonstandard return if unable to process the request due to undefined TRL or MCB, insufficient core, or incorrect value of NSKIP.

NSKIP - If greater than one, the first NSKIP - 1 columns of TRL are not copied to MCB, otherwise, MCB(1) is set to TRL(1).

TRL - Seven-word array, the first word containing the GINØ file number of the input matrix.

MCB - Seven-word array, the first word containing the GINØ file number of the output matrix.

ZZ - Open core

IBUF - Location of two GINØ buffers with respect to ZZ(1).

3. Purpose: If NSKIP > 1, to create a matrix without the first NSKIP - 1 columns, otherwise, to set MCB(1) to TRL(1).

4. Additional Subroutines: CYPFIL

4.156.9.3 Subroutine Name: EQMCKS

1. Entry Point: EQMCKS

2. Calling Sequence: CALL EQMCKS

COMMON / EQMK1 / - see Section X.Y.Z.10

3. Purpose: To calculate and output the overall total of forces and moments on the structure.

4. Additional Subroutines: EJECT, MPYAD, PAGE1

4.156.10 Common Block Interface

COMMON / EQMK1 / CASECC,EQEXIN,GPL,BGPDT,SIL,USET,KGG,GM,UGV,PGG,QG,CSTM,LAMA,ØQM,KSCR(7),
KMPC,KLOAD,KSPC,PARM(4),TRL(7)

where:

CASECC,..., LAMA - GINØ file number of input data files with the same names. Set negative in EQMCK if not available.

ØQM - GINØ file number of output data file. Set negative if necessary GINØ files are not available.

FUNCTIONAL MODULE EQMCK

KSCR - Seven scratch files
KMPC - Positive if MPC-forces are to be calculated
KLØAD - Positive if static load vector exists
KSPC - Positive if SPC-force vector exists
PARM - Error message data
TRL - Trailer array for use by subroutines

FUNCTIONAL MODULE MPY3 (TRIPLE MATRIX MULTIPLY)

4.157 FUNCTIONAL MODULE MPY3 (TRIPLE MATRIX MULTIPLY)

4.157.1 Entry Point: MPY3

4.157.2 Purpose

To perform the matrix product $A^T B + E$ or $BA + E$ for sparse A matrix and full B matrix.

4.157.3 DMAP Calling Sequence

MPY3 FILEA,FILEB,FILEE/FILEC/V,N,CODE/V,N,PREC \$

4.157.4 Input Data Blocks

FILEA Matrix A

FILEB Matrix B

FILEE Matrix E

4.157.5 Output Data Blocks

FILEC Matrix C = $A^T B + E$, $A^T B + E$ or $BA + E$

4.157.6 Parameters

CODE Input, integer, default = 0. If value = 0, $A^T B + E$ is performed. If value = 1, $A^T B + E$ is performed via MPYAD. If value = 2, $BA + E$ is performed.

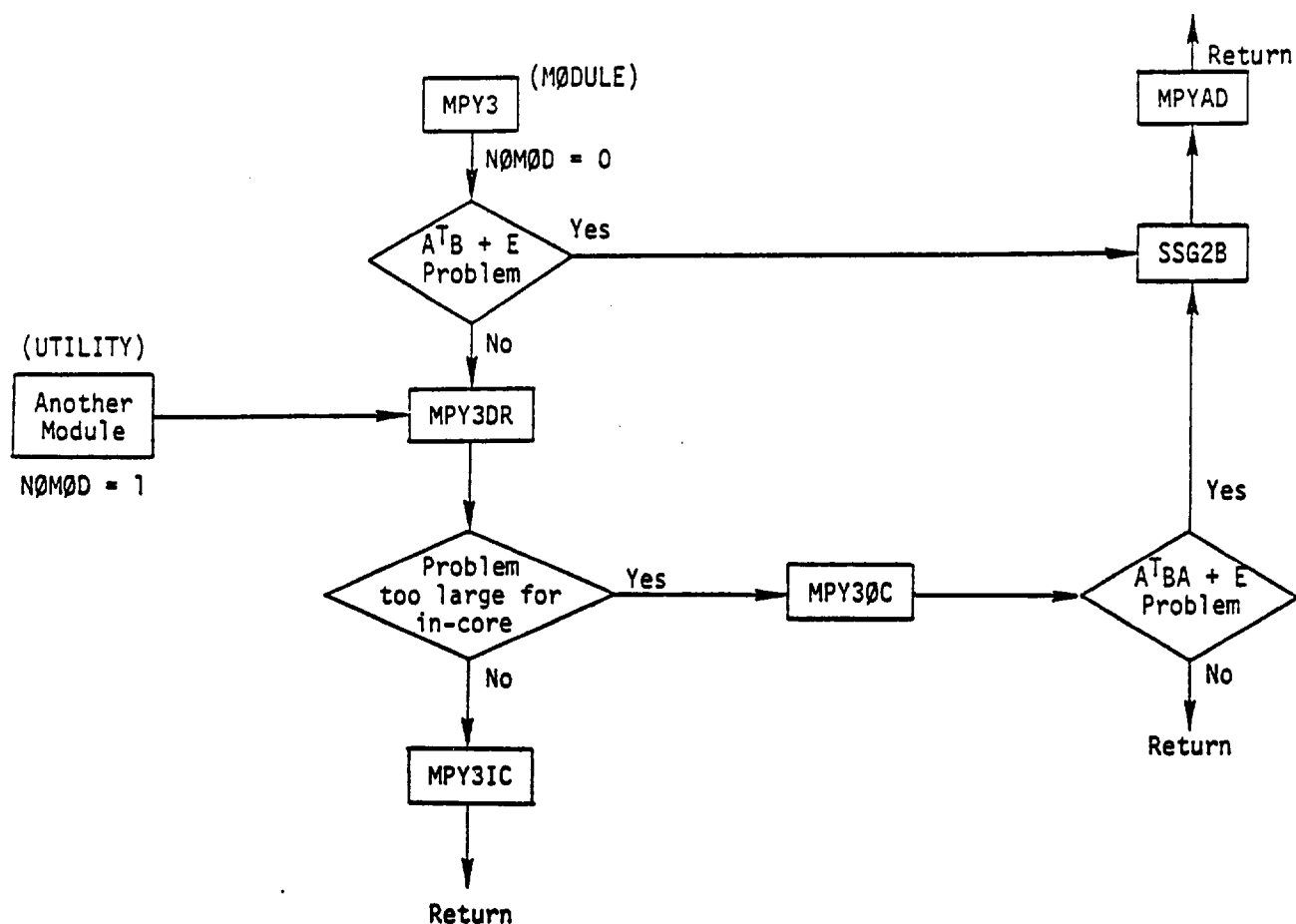
PREC Input, integer, default = 0. If value = 0, output precision is calculated from input matrices. If value = 1, output is in real single precision. If value = 2, output is in real double precision.

4.157.7 Method

As outlined on the following page, the module is subdivided into a primary driver subroutine, a secondary driver subroutine, two major subroutines, and five utility subroutines.

4.157.7.1 Initialization

During module execution the primary driver, subroutine MPY3, assigns file numbers, determines the precision of the output matrix, determines the length of open core, and calls the secondary driver subroutine. If the problem is $A^T B + E$, SSG2B is called to perform the computation via MPYAD.



Subroutine MPY3DR, the secondary driver, calculates the number of columns (NK) of the B matrix able to fit in core in unpacked form in addition to the packed A matrix. If this value is less than 3, the program branches to the out-of-core processor MPY30C. Otherwise, processing is done via MPY3IC, the in-core processor. During non-module execution, MPY3DR is called by the current module.

4.157.7.2 In-Core Processing

Subroutine MPY3IC oversees the in-core computation. Associated with MPY3IC are utility subroutines MPY3A, MPY3B, MPY3C, MPY3P, and MPY3NU.

1. MPY3A is called to perform two functions. It first unpacks the B matrix one column at a time and repacks it onto a scratch file. The file position of each column is stored in array SAVP. Secondly, MPY3A constructs an in-core packed form on the matrix A^T . Three

arrays are used to store the necessary information concerning A^T . Array ATRANS contains the nonzero terms of matrix A row-by-row. ACOLS contains the column numbers of the terms in ATRANS. APPOINT contains pointers identifying the position in ACOLS and ATRANS where each row of A appears.

2. The product matrix $C = A^T B A + E$ or $BA + E$ is then computed column-by-column.
 - a) At the outset of the computation of each column of C, the column is initialized to 0, and the corresponding column of matrix E is then summed to it, if E exists.
 - b) Then, subroutine MPY3B is called to perform a number of functions. It first reads the corresponding column of A and stores its nonzero terms in array AKJ and the row positions of these terms in array KTBP. If the first column of C is the one being processed, the columns of B whose column numbers match with the entries in array KTBP are then read into array BCOLS, up to NK such columns, and the column numbers are stored in array BCID. Otherwise, the columns of B already resident in core are the ones considered. Subroutine MPY3P is then called for each term in AKJ having a matching column of B in core to perform the computation (multiplication and sum accumulation).

Equations for Computing a Column of $C = E + A^T B A$

$$\begin{bmatrix} C_{1j} \\ \vdots \\ C_{ij} \\ \vdots \\ C_{mj} \end{bmatrix} = \begin{bmatrix} E_{1j} \\ \vdots \\ E_{ij} \\ \vdots \\ E_{mj} \end{bmatrix} + \begin{bmatrix} \sum_{k=1}^n \sum_{l=1}^n a_{1l} b_{lk} a_{kj} \\ \vdots \\ \sum_{k=1}^n \sum_{l=1}^n a_{il} b_{lk} a_{kj} \\ \vdots \\ \sum_{k=1}^n \sum_{l=1}^n a_{ml} b_{lk} a_{kj} \end{bmatrix}$$

The sums are accumulated in core in array C. Initially, the corresponding column of E is summed to array C. In the innermost loop, i is varied; then l, then k, then j.

- c) If each term in AKJ has been processed, the current column of C is packed and processing begins on the next column. Otherwise, a test is made to see whether NK columns of B reside in core. If this is the case, the "next-to-be-used" tables are

then constructed. This is done using the utility subroutine MPY3MU (see part d).

If less than NK columns of B reside in core, calls are made to MPY3C to fill additional columns into BCØLS and to process additional terms in AKJ until BCØLS is full.

- d) Subroutine MPY3NU makes use of arrays APOINT and AOCLS to determine for each column of B in core and for each unprocessed row of the current column of A the corresponding "next-time-used" values. The "next-time-used" value is precisely the column number of the next column of matrix A needing the particular column of B, or containing a non-zero term in the particular row of A in question. Array BNTU contains the next-to-be-used values for columns of B in core and ANTU, the next-to-be-used values for the unprocessed rows in the current column of A. These tables are continually updated throughout the remainder of the computation.
- e) In subsequent processing, when the needed column of B is not in core, MPY3C is called and substitutes the column of B in core having the highest next-to-be-used value with the column of B corresponding to the unprocessed term in AKJ having the highest next-to-be-used value. This ensures maximum efficiency in column substitution, as columns of B to be used frequently during this stage will remain in core.
- f) After all terms in AKJ have been processed, the computed column of C is packed and the next column is processed.

4.157.7.3 Out-of-Core Processing

Subroutine MPY3OC oversees the out-of-core computation. MPY3OC is similar to MPY3IC, but differs in two ways:

1. Only the product BA is performed within MPY3OC, while SSG2B is called to multiply the product BA by A^T , if necessary.
2. The management of the next-to-be-used tables is handled entirely within the subroutine MPY3OC itself. Use is made of the arrays NTBU, LAST, and NTBU2 for this purpose.

Thus, the arrays APØINT, ACØLS, and ATRANS are not used, and hence are not processed in MPY3A.

4.159.8 Subroutines

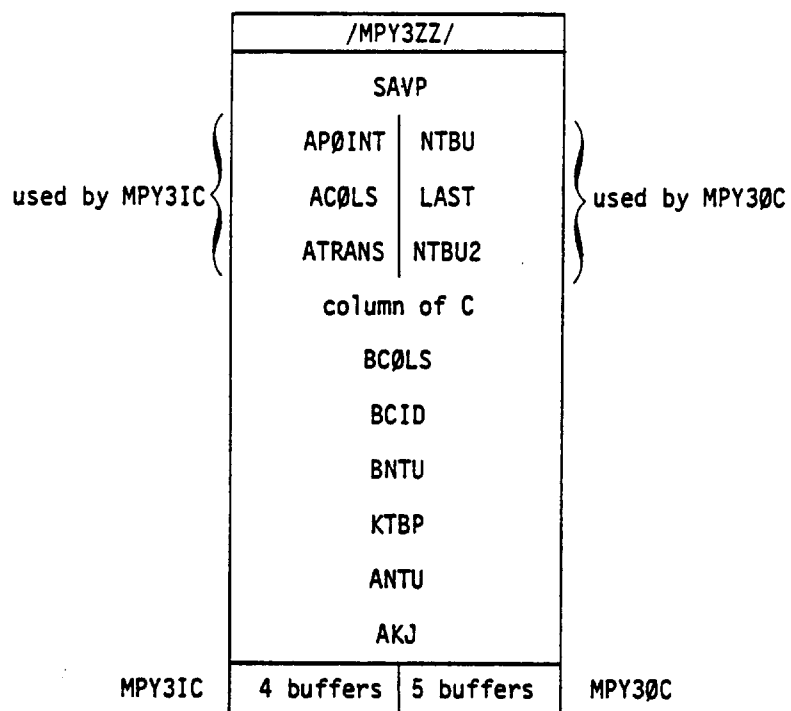
See Section 3.5.31.5.

4.157.9 Common Blocks

1. / / DMAP parameters
2. /MPY3BC/ Type of solution; precision
3. /MPY3TL/ Matrix trailers, open core positions of buffers, and a logical variable denoting the existence or non-existence of matrix E.
4. /MPY3CP/ Various parameters needed by more than one subroutine, including pointers to positions in open core.
5. /MPY3ZZ/ Open core.

4.157.10 Design Requirements

Open core for the entire module is illustrated as follows:



4.157.11 Diagnostic Messages

Fatal messages 6551, 6552, 6553, 6554, 6555, 6556, and 6557 may be issued.

MODULE FUNCTIONAL DESCRIPTIONS

4.158 FUNCTIONAL MODULE MRED1 (MODAL SYNTHESIS REDUCTION CALCULATION INITIALIZATION)

4.158.1 Entry Point: MRED1

4.158.2 Purpose

Initialization of the modal synthesis reduction calculations.

4.158.3 DMAP Calling Sequence

MRED1 CASECC,GEOM4,DYNAMICS/USCTX,EEDX,EQST,DMR/*NAMEA*/S,N,DRY/STEP/S,N,NØUS/S,N,SKIPM/
TYPE \$

4.158.4 Input Data

4.158.4.1 GINØ Data Blocks

CASECC - The Case Control data block from which the reduction instructions are obtained.

GEOM4 - NASTRAN data block of displacement set definitions and substructuring bulk data.

Extracted from this block is the BDYC data corresponding to the fixed and
boundary sets given in CASECC and the BDYS and BDYS1 data specified on BDYC.

DYNAMICS - Eigenvalue extraction method data block

4.158.4.2 SØF Items

EQSS - Substructure equivalence table

BGSS - Basic grid point definition table

CSTM - Coordinate system transformation matrices

4.158.5 Output Data

4.158.5.1 GINØ Data Blocks

USCTX - Identifies s, r, and b degrees of freedom

EEDX - Contains eigenvalue extraction data

EQST - Temporary EQSS table

DMR - Rigid body matrix

4.158.5.2 SØF Items

None

MODULE FUNCTIONAL DESCRIPTIONS

4.158.6 Parameters

NAMEA - Input, BCD, name of input substructure
DRY - Input, output, integer - defines mode of operations
STEP - Input, integer, identifies CASECC record for control data
NØUS - Øutput, equals -1 if no "fixed" points are defined
SKIPM - Øutput, equals -1 if modes are already present
TYPE - Input, BCD, either "REAL" or "COMPLEX"

4.158.7 Method

The following logical sequence is performed by the MRED1 MODULE:

- a. Process fixed sets (Subroutines MRED1A, MRED1B)
- b. Process boundary sets (Subroutines MRED1A, MRED1B)
- c. Convert EQSS data to U_p data (Subroutine MRED1C)
- d. Process eigenvalue data (Subroutine MRED1D)
- e. Process free body modes (Subroutine MRED1E)

4.158.8 Subroutines

4.158.8.1 Subroutine Name: MRED1

1. Entry Point: MRED1
2. Purpose: Driver program to begin initialization of the modal synthesis reduction for both real and complex.
3. Calling sequence: CALL MRED1

COMMON /BLANK/ ØLDNAM(2), DRY, STEP, NØUS, SKIPM, TYPE(2), GPRM, GBUF1, GBUF2, SBUF1, SBUF2,
SBUF3, KØRLN, NEWNAM(2), BNDSET, FIXSET, IEIG, IØ, RGRID(2), RNAME(2), IIRSAVE, KØRBGN,
NCSUBS, NAMEBS, EQSIND, NSLBGN, NSIL, ØDYCS, NBDYCC, USETL, USTLØC, RGRIDX, RGRIDY,
RGRIDZ, USRMØD, ØØUNDS

ØLDNAM - Name of substructure being reduced - BCD - input
DRY - Module operation flag - integer - input
STEP - Control data CASECC record - integer - input
NØUS - Fixed points flag (integer)
.EQ. 1 if fixed points defined
.EQ. -1 if no fixed points defined

FUNCTIONAL MODULE MRED1 (MODAL SYNTHESIS REDUCTION CALCULATION INITIALIZATION)

SKIPM - Modes flag (integer)
 .EQ. 0 if modes not present
 .EQ. -1 if modes present

TYPE - Reduction type - BCD - input

GPRM - G parameter value - real - input

GBUF - GINØ buffers

SBUF - SØF buffers

KØRLEN - Core length

NEWNAM - New substructure name

BNDSET - Boundary set identification number

FIXSET - Fixed set identification number

IEIG - Eigenvalue set identification number

IØ - Øutput flags

RGRID - Freebody reference GRID point for freebody calculations

RNAME - Freebody substructure component name and freebody modes flag

IRSAVE - RSAVE flag

KØRBGN - Beginning address of open core

NCSUBS - Number of contributing substructures

NAMEBS - Beginning address of contributing substructure names

EQSING - Beginning address of EQSS group addresses

NSLBGN - Beginning address of SIL data

NSIL - Number of SIL groups

BDYCS - Beginning address of BDYC data

NBDYCC - Number of BDYC data groups

USETL - Length of USET array

USTLØC - Beginning address of USET array

RGRIDX - Freebody mode relative X coordinate

RGRIDY - Freebody mode relative Y coordinate

RGRIDZ - Freebody mode relative Z coordinate

USRMØD - User mode Type 2 option flag

BØUNDS - Old bounds option flag

MODULE FUNCTIONAL DESCRIPTIONS

COMMON/SYSTEM/SYSBUF,IPRNTR

SYSBUF - System buffer size

IPRNTR - Printer logical output device number

4. Method:

- a. Initialize GINØ and SØF files and buffers.
- b. Initialize Case Control parameters.
- c. Process the Case Control data block.

The allowable case control mnemonics for the MRED1 operation are:

NAMB - Name given to resultant pseudostructure

BØUN - Boundary set ID number

FIXE - Fixed set ID number

METH - Real eigenvalue extraction method ID number

CMET - Complex eigenvalue extraction method ID number

ØUTP - Selective output control flag

RGRI - Rigid body grid point ID number

ØLDM - ØLDMØDES option flag

ØLDB - ØLDBØUNDS option flag

RSAB - Save reduction products option flag

RNAM - Rigid body substructure name

RANG - Eigenvalue mode frequency range

NMAX - Lowest number of modes within frequency range to be used

USER - User modes option flag

NAMA - Name of substructure to be reduced

These values are extracted from the CASECC input block.

- d. The problem summary is printed if requested.
- e. If user modes is not type 2 and the modes are not already present, the following error checks are performed:

FUNCTIONAL MODULE MRED1 (MODAL SYNTHESIS REDUCTION CALCULATION INITIALIZATION)

If \emptyset LDM \emptyset DES is set, PHIS and LAMS must exist on the S \emptyset F.

If \emptyset LDM \emptyset DES is not set, the PHIS and LAMS items must be deleted.

If \emptyset LDB \emptyset UND is set, GIMS and UPRT must exist on the S \emptyset F.

If \emptyset LDB \emptyset UND is not set, the GIMS and LMTX items must be deleted.

f. The EQSS group 0 data for substructure NAMEA is read into core and the names of all component substructures contained in the psedo structure being reduced are placed in the first (NWDSRD-4) of open core.

```
Z (NAMEBS): NAME11  NAME12
              NAME21  NAME22
              .
              .
              .
```

g. A two word entry array NCSUBS long is generated for each EQSS basic substructure. The first word contains the beginning address of the EQSS basic substructure data and the second holds the length in words of the basic substructure.

```
Z (EQSIND): BSUB1  address
              BSUB1  length
              BSUB2  address
              BSUB2  length
              .
              .
              .
```

} 2* NCSUBS long

The remaining EQSS groups are read starting at K \emptyset RBGN of open core.

```
Z (K $\emptyset$ RBGN): EQSS  group 1 data
              .
              .
              .
              EQSS  group NCSUBS data
```

The EQSS SIL data is read to end-of-item.

```
Z (NSLBGN): SIL1  C $\emptyset$ MP1
              SIL2  C $\emptyset$ MP2
              .
              .
              .
```

} NSIL entries

MODULE FUNCTIONAL DESCRIPTIONS

- h. The length of the USET array is determined.
- i. The fixed and boundary sets are processed (See 4.158.8.2).
- j. The EQSS data is converted to UB data (See 4.158.8.3).
- k. If the `OLDMODES` option was not specified, the eigenvalue data is processed (See 4.158.8.4).
- l. Free body modes are processed (See 4.158.8.5).

4.158.8.2 Subroutine Name: MRED1A

- 1. Entry Point: MRED1A
- 2. Purpose: To process the BDYC data for the fixed identification set (FIXSET) and the boundary identification set (BNDSET) for the MRED1 module.
- 3. Calling Sequence: CALL MRED1A (MODE)

MODE - Processing operation flag

.EQ. 1, Process fixed ID set

.EQ. 2, Process boundary ID set

COMMON /BLANK/ - See /BLANK/ description in section 4.158.8.1

COMMON /SYSTEM/ IDUM8,IPRINTR,IDUM9(6),NLPP,IDUM10(2),LINE

IPRINTR - Printer logical output device number

NLPP - Maximum number of lines per page

LINE - Current number of lines per page

COMMON /TWQ/ ITWQ(32)

ITWQ - Powers of two

COMMON /BITPOS/ - See /BITPOS/ description in section 4.158.8.3.

- 4. Method: If no fixed set is defined, or the `OLDMODES` option was specified, the parameter `NQUS` is set to -1. Otherwise the UL,UA,UF,UN and UI bits are turned on in the USET array.

FUNCTIONAL MODULE MRED1 (MODAL SYNTHESIS REDUCTION CALCULATION INITIALIZATION)

```
Z (KORUST): LAFNI
            LAFNI
            LAFNI
            .
            .
            .
            } USCTL long
```

The BDYC boundary set bulk data for the requested fixed set or boundary set ID is read from the GEOM4 data block.

```
Z (KBDYC) = NAME1  SID1
            NAME2  SID2
            .      .
            .      .
            .      .
            } NBDYCC long
```

The substructure specified on the BDYC bulk data card is checked against the list of components. If the substructure is not a component of the structure being reduced, a fatal message is printed and processing continues. After all BDYC data has been processed, a check for duplicate BDYC substructure names is made. If duplicates are present, a fatal message is generated and processing continues. The BDYC data is then sorted on set ID.

4.158.8.3 Subroutine Name: MRED1B

1. Entry Point: MRED1B
2. Purpose: To process the BDYS and BDYS1 data for the fixed identification set (FIXSET) and the boundary identification set (BNDSET) for the MRED1 module.
3. Calling Sequence: CALL MRED1B (MODE)

MODE - Subroutine processing flag
 .EQ. 1, Process fixed ID set
 .EQ. 2, Process boundary ID set

COMMON /BLANK/ - See /BLANK/ description in section 4.158.8.1.

COMMON /SYSTEM/ - See /SYSTEM/ description in section 4.158.8.2.

COMMON /TWO/ - See /TWO/ description in section 4.158.8.2.

MODULE FUNCTIONAL DESCRIPTIONS

COMMON /BITPOS/ IDUM10(5), UL,UA,UF,US,UN,IDUM11(10),UB,U1

UL - L bit position

UA - A bit position

UF - F bit position

US - S bit position

UN - N bit position

UB - B bit position

UI - I bit position

COMMON /PATX/ LCORE,NSUB(3),FUSET

LCØRE - Length of open core

NSUB - Number of columns in the 0,1,2 subsets

FUSET - USET array

COMMON /UNPAKX/ - See /UNPAKX/ description in section 4.158.8.6.

4. Method: The BDYS and BDYS1 data is read from the GEOM4 data block for each BDYC SID specified in the fixed set or boundary set.

$$\begin{array}{l} \text{BDYS} \left\{ \begin{array}{lll} \text{SID}_1 & G_b & C_b \\ & G_b & C_b \\ & \vdots & \\ & \vdots & \\ \text{SID}_2 & G_b & C_b \\ & \vdots & \\ & \vdots & \\ & \vdots & \\ & \vdots & \\ & \vdots & \\ & \vdots & \end{array} \right. \\ \\ \text{BDYS1} \left\{ \begin{array}{lll} \text{SID}_1 & G_b & C_b \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \end{array} \right. \end{array}$$

The EQSS data for the substructure is located. If it cannot be located, a fatal error is

FUNCTIONAL MODULE MRED1 (MODAL SYNTHESIS REDUCTION CALCULATION INITIALIZATION)

issued and processing continues. The correct IP is located for the external ID and the grid point ID and component is converted to a SIL value. The USET array is filled by turning off the UL, UA, and UF bits and turning on the US bit if the fixed set is being processed, or turning off the UI bit and turning on the UB bit if the boundary set is being processed. A check is then made to see if all IPs have been found. If not, a warning message is generated. If boundary data is not requested, a fatal message is issued. Non-existing data causes the generation of a fatal message and the setting of DRY = -2. The USET data block is then written. If the \emptyset LDB \emptyset UND option was specified, a check is made to verify that the old boundary points have not been changed.

4.158.8.4 Subroutine Name: MRED1C

1. Entry Point: MRED1C
2. Purpose: To convert the EQSS data and BGSS data to correspond to the boundary degrees of freedom (UB) for the MRED1 Module.
3. Calling Sequence: CALL MRED1C

COMMON / / - See / / description in section 4.158.8.1.

COMMON /TW \emptyset / - See /TW \emptyset / description in section 4.158.8.2.

COMMON /BITP \emptyset S/ See /BITP \emptyset S/ description in section 4.158.8.3.
4. Method: The SIL DOF is decoded. Its corresponding entry in the USET array is determined and a test is made to check for the UB bit being set. If the UB bit is set, the new component code is set by using the logical " \emptyset R" function and the number of DOF for the SIL is counted. If the SIL DOF was kept, the new IP is counted and the new SIL DOF is saved at location (SILDOF-1) of open core with the new component stored at SILD \emptyset F of open core. The new SIL value is stored as $(8 * \text{NEWIPS}) + \text{number of DOFs}$. If the SIL DOF was not kept, a -1 is saved at (SILD \emptyset F -1).

MODULE FUNCTIONAL DESCRIPTIONS

Z (NSLBGN): SIL1 COMP1

SIL2 COMP2

·
·
·

Z (SILDØF-1): (8* NEWIPS) + NDØR NEW COMP

-1

0

The EQSS group 0 data is written onto the temporary EQST table. The grid point data corresponding to each basic substructure is processed one entry at a time. The new internal point number for the SIL location is found for the old number. The new IP and component code are located. If the component code exists, the grid point ID, new IP number, and component is written onto the EQST. Each basic substructure occupies one logical record on the EQST.

The old SIL entries are reduced. If the old (SILDØF-1) is not equal to -1, it and its component is saved at (2*NSIL) of open core. The number of DØF is counted by the logical "AND" function using the new SIL DØF. The new SIL data is written onto the EQST table. The results of this will be, for example,

	<u>IP_{old}</u>	<u>C_{old}</u>		<u>SIL_{new}</u>	<u>C_{new}</u>
Z (NSLBGN):	-1	0	Z (NEWSIL):	1	111
	-1	0		4	1011
	1	111		7	01
	-1	0		8	1101
	2	1011		·	·
	3	01		·	·
	4	1101		·	·
	-1	0			
	·	·			
	·	·			
	·	·			

FUNCTIONAL MODULE MRED1 (MODAL SYNTHESIS REDUCTION CALCULATION INITIALIZATION)

The BGSS group 0 data is read from the SØF and written onto the EQST table. Each BGSS data entry is read. If the old SIL DØF is not -1, the data is copied onto the EQST. After all BGSS data has been read, the EQST table is closed.

4.158.8.5 Subroutine Name: MRED1D

1. Entry Point: MRED1D
2. Purpose: To generate the EEDX data block using the EED data block format from the EIGR or EIGC and EIGP bulk data for the MRED1 module.

3. Calling Sequence: CALL MRED1D

COMMON /BLANK/ - See /BLANK/ description in section 4.158.8.1.

COMMON /TWØ/ - See /TWØ/ description in section 4.158.8.2.

COMMON /SYSTEM/ - See /SYSTEM/ description in section 4.158.8.1.

4. Method: If TYPE = REAL, the EIGR data is located. If TYPE = COMPLEX, the EIGC and EIGP data is processed. The data is read from the DYNAMICS data block until the data with METHOD = ID is found. The data is then written onto the EEDX data block in the same format as read from the DYNAMICS data block. In the case of TYPE = REAL, only one record is written onto the EEDX data block. In the case of TYPE = COMPLEX, EIGC data and maybe EIGP data is written onto the EEDX data block if they exist on the DYNAMICS data block. Missing EIGR or EIGC data causes a fatal message. If EIGP data is specified without EIGC data, a fatal message is issued.

4.158.8.6 Subroutine Name: MRED1E

1. Entry Point: MRED1E
2. Purpose: To generate the rigid body matrix DMX if freebody modes are requested for the MRED1 module.
3. Calling Sequence: CALL MRED1E

COMMON /PACKX/ TYPIN,TYPPCK,IRØWP,LRØWP,INCRP

TYPIN - Input data type

TYPPCK - Output data type

IRØWP - Row position of first variable to pack

MODULE FUNCTIONAL DESCRIPTIONS

LRØWP - Row position of last variable to pack

INCRP - Increment amount between variables to pack

COMMON /UNPAKX/ TYPUNP,IRØWUP,LRØWUP,INCRUP

TYPUNP - Output data type

IRØWUP - Row position of first variable to unpack

LRØWUP - Row position of last variable to unpack

INCRUP - Increment amount between variables unpacked

4. Method: The BGSS item is read into core, followed by the CSTM item. The freebody basic coordinates are determined. If no internal grid point ID number has been set, the basic coordinates are set to zero. Otherwise the basic coordinates are set to the X, Y, Z of the BGSS item for substructure NAMEA. For each point of the BGSS, the (3x3) matrix

$$[d] = \begin{bmatrix} 0 & \Delta Z & -\Delta Y \\ -\Delta Z & 0 & \Delta X \\ \Delta Y & -\Delta X & 0 \end{bmatrix}$$

is generated with $\Delta X = X_i - X_0$, $\Delta Y = Y_i - Y_0$, and $\Delta Z = Z_i - Z_0$. If CID_i of the BGSS item for substructure NAMEA is positive, the (3x3) transformation matrix $[T_i]$ is generated using TRANS. The matrix (3x3)

$$[TTD] = [T_i]^T [d]$$

is generated by a call to GMMATS. If CID_i is zero, $[T_i]$ is the identity matrix and $[TTD] = [d]$. If CID_i is negative (a scalar point), a null column is added to $[D]$.

The (6x6) matrix

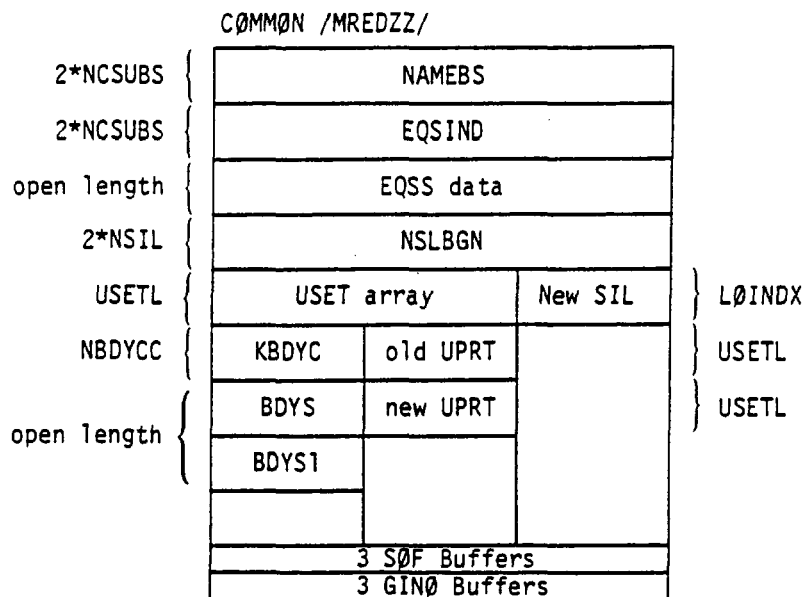
$$[D] = \begin{bmatrix} T^T & | & T^T d \\ \hline 0 & | & T^T \end{bmatrix}$$

is generated. A SIL entry (2 words) is read and the component code decoded. The rows of $[D]$ are extracted corresponding to the active SIL components. The extracted rows of $[D]$ are then saved onto a scratch file. After the BGSS has been processed, the scratch file is read into open core into transposed form. The transposed form is then written onto the DMR data block.

FUNCTIONAL MODULE MRED1 (MODAL SYNTHESIS REDUCTION CALCULATION INITIALIZATION)

4.158.9 Design Requirements

One scratch file is needed. Open core resides in common block /MREDZZ/. Open core is defined as follows:



4.158.10 Diagnostic Messages

Diagnostic messages 3001-3003, 3008, 6101-6103, 6367, 6603, 6604, 6606, 6610, 6611, 6617-6630, 6636, and 6637 are issued.

MODULE FUNCTIONAL DESCRIPTIONS

4.159 FUNCTIONAL MODULE MRED2 (MODAL SYNTHESIS REAL REDUCTION CALCULATIONS)

4.159.1 Entry Point: MRED2

4.159.2 Purpose:

To perform the computations for the modal reduce command.

4.159.3 DMAP Calling Sequence

MRED2 CASECC,LAMAMR,PHISS,EQST,USETMR,KAA,MAA,BAA,K4AA,PAA,DMR,QSM/KHH,MHH,BHH,K4HH,PHH,
PØVE/STEP/S,N,DRY/PØPT \$

4.159.4 Input Data

4.159.4.1 GINØ Data Blocks

CASECC - Case Control Data

LAMAMR - Eigenvalue table for substructure being reduced

PHISS - Eigenvectors for substructure being reduced

EQST - EQSS data for boundary set for substructure being reduced

USETMR - USET table for reduced substructure

KAA - Substructure stiffness matrix

MAA - Substructure Mass matrix

BAA - Substructure viscous damping matrix

K4AA - Substructure structure damping matrix

PAA - Substructure load matrix

DMR - Free body matrix

QSM - Model reaction matrix

- Notes:
1. LAMAMR and PHISS may be purged if the modes exist on the SOF.
 2. KAA,...,PAA may be purged depending on the substructure OPTION command.
 3. DMR may be purged if no SUPPORT command is used.
 4. QSM cannot be purged if type 2 user modes are requested.

MODULE FUNCTIONAL DESCRIPTIONS

4.159.4.2 SØF Items

LAMS - Eigenvalue table for original substructure
PHIS - Eigenvector table for original substructure
LMTX - Stiffness decomposition product for original substructure
GIMS - G transformation matrix for boundary points for original substructure
HØRG - H transformation matrix for original substructure

4.159.5 Output Data

4.159.5.1 GINØ Data Blocks

KHH - Reduced stiffness matrix
MHH - Reduced mass matrix
BHH - Reduced viscous damping matrix
K4HH - Reduced structure damping matrix
PHH - Reduced load matrix
PØVE - Interior point load matrix

Note: KHH,...,PHH may be purged depending on input data blocks provided.

4.159.5.2 SØF Items

LAMS - Eigenvalue table for original substructure
PHIS - Eigenvector table for original substructure
LMTX - Stiffness decomposition product for original substructure
GIMS - G transformation matrix for boundary points for original substructure
HØRG - H transformation matrix for original substructure
UPRT - Partitioning vector for MREDUCE for original substructure
PØVE - Internal point loads for original substructure
PØAP - Internal points appended loads for original substructure
EQSS - Substructure equivalence table for reduced substructure
BGSS - Basic grid point definition table for reduced substructure
CSTM - Coordinate system transformation matrices for reduced substructure
LØDS - Load set data for reduced substructure
LØAP - Appended load SET data for reduced substructure
PLTS - Plot set data for reduced substructure

FUNCTIONAL MODULE MRED2 (MODAL SYNTHESIS REDUCTION CALCULATIONS)

KMTX - Stiffness matrix for reduced substructure

MMTX - Mass matrix for reduced substructure

PVEC - Load matrix for reduced substructure

PAPP - Appended load matrix for reduced substructure

BMTX - Viscous damping matrix for reduced substructure

K4MX - Structure damping matrix for reduced substructure

4.159.6 Parameters

STEP - input - integer - no default. Identifies the Case Control record which contains the MREDUCE data.

DRY - output - integer. A flag which is set during module execution if error conditions prevent a requested GO operation.

P0PT - input - BCD - no default. Set to PVEC or PAPP for options P and PA respectively.

4.159.7 Method:

The following logical sequence is performed by the MRED2 module:

- a. Process Case Control (Subroutine MRED2)
- b. Process Stiffness Matrix (Subroutine MRED2A)
- c. Process 0LDBOUND Option (Subroutine MRED2B)
- d. Process 0LDM0DE Option (Subroutine MRED2C)
- e. Process Usermodes Option (Subroutine MRED2D, MRED2I-P)
- f. Calculate Modal Transformation Matrix (Subroutine MRED2E)
- g. Calculate Freebody Effects (Subroutine MRED2F)
- h. Calculate Structural Matrices (Subroutine MRED2G)
- i. Process New 00F Table Items (Subroutine MRED2H)

4.159.8 Subroutines

4.159.8.1 Subroutine Name: MRED2

1. Entry Point: MRED2
2. Purpose: Driver which performs the major computations for the modal reduce command.
3. Calling Sequence: CALL MRED2

MODULE FUNCTIONAL DESCRIPTIONS

COMMON /BLANK/ STEP, DRY, PØPT, GBUF1, GBUF2, GBUF3, SBUF1, SBUF2, SBUF3, INFILE(12),
ØTFILE(6), ISCR(10), KØRLEN, KØRBGN, ØLDNAM(2), NEWNAM(2), FREBDY, RANGE(2), NMAX,
USRMØD, IØ, BØUNDS, MØDES, RSAVE, LAMSAP, MØDPTS, MØDLEN

STEP - Control Data CASECC Record (Integer)

PØPT - PVEC or PAPP option flag (BCD)

DRY - Module operation flag (integer)

GBUF - GINØ Buffers

SBUF - SØF Buffers

INFILE - Input file numbers

ØTFILE - Øutput file numbers

ISCR - Array of scratch file numbers

KØRLEN - Length of open core

KØRBGN - Beginning address of open core

ØLDNAM - Name of substructure being reduced

NEWNAM - Name of reduced substructure

FREBDY - Free body modes calculation flag

RANGE - Range of frequencies to be used

NMAX - Maximum number of frequencies to be used

USRMØD - USERMØDE calculation flag

IØ - IØ options flag

BØUNDS - ØLDBØUNDS option flag

MØDES - ØLDMØDES option flag

RSAP - Save reduction product flag

LAMSAP - Beginning address of mode use description array

MØDPTS - Number of modal points

MØDLEN - Length of mode use array

COMMON /SYSTEM/ SYSBUF, IPRNTR

SYSBUF - System buffer size

IPRNTR - Printer logical output device number

FUNCTIONAL MODULE MRED2 (MODAL SYNTHESIS REDUCTION CALCULATIONS)

4. Method:

- a. Initialize GINØ and SØF files and buffers.
- b. Initialize Case Control parameters
- c. Process Case Control data block.

The allowable case control mnemonics for the MRED2 operation are:

NAMA - Name of substructure to be reduced
NAMB - Name of resultant reduced substructure
FREE - Freebody modes calculation flag
RANG - Eigenvalue mode frequency range
NMAX - Maximum number of mode frequencies with range to be used
USER - User modes calculation flag
ØUTP - Selective output control flag
ØLDB - Øldbounds option flag
ØLDM - Øldmodes option flag
RSAV - Save reduction products flag

These values are extracted from the CASECC input block.

- d. If RUN = GØ, calculate the structural matrices (see Section 4.159.8.3) and end modal reduction.
- e. If not RUN = GØ, and not a User mode type 2 reduction then
 - process stiffness matrix (see Section 4.159.8.2),
 - process oldbounds option (see Section 4.159.8.3),
 - process oldmodes option (see Section 4.159.8.4),
 - calculate modal transformation matrix (see Section 4.159.8.5),
 - calculate free body effects (see Section 4.159.8.7)
 - calculate the structural matrices (see Section 4.159.8.8), and
 - process the new SØF table items (see Section 4.159.8.9).
- f. If not RUN = GØ and it is a User mode type 2 reduction, then
 - process user mode option (see Section 4.159.8.5)
 - process the new SØF table items (see Section 4.159.8.9).

MODULE FUNCTIONAL DESCRIPTIONS

4.159.8.2 Subroutine Name: MRED2A

1. Entry Point: MRED2A
2. Purpose: To partition the stiffness matrix into boundary and interior points and save the partitioning vector on the SØF as the UPRT item for substructure NAMA.
3. Calling Sequence: CALL MRED2A

COMMON /BLANK/ - See Section 4.159.8.1

COMMON /BITPOS/ - See Section 4.159.8.5

COMMON /PATX/ LCØRE, NSUB(3), FUSET

LCØRE - Length of open core

NSUB - Number of rows in 0,1, and neither 0,1 subset

FUSET - USET array

COMMON /SYSTEM/ - See Section 4.159.8.9

4. Method: The stiffness matrix is partitioned to boundary and interior points

$$[KAA] = \begin{bmatrix} KBB & KBI \\ \hline KIB & KII \end{bmatrix}$$

using Subroutines CALCV and PARTN.

The partitioning vector calculated is saved on the SØF as the UPRT item for substructure ØLDNAM.

4.159.8.3 Subroutine Name: MRED2B

1. Entry Point: MRED2B
2. Purpose: To perform the GUYAN reduction on the structure points.
3. Calling Sequence: CALL MRED2B

COMMON /BLANK/ - See Section 4.159.8.1

COMMON /SFACT/ KIIT(7), LIIT(7), ISCRQ(7), IS CRA, IS CRB, NZSF, DETR, DETI, POWER, IS CRC, MINDIA, CHLSKY

KIIT - Matrix control block for KII

FUNCTIONAL MODULE MRED2 (MODAL SYNTHESIS REDUCTION CALCULATIONS)

LIIT - Matrix control block for LII
ISCRQ - Matrix control block for a scratch file
ISCRA - Scratch file A
ISCRB - Scratch file B
NZSF - Number of open core words
DETR - Determinate of LII (real)
DETI - Determinate of LII (imaginary)
POWER - Power of LII
ISCRC - Scratch file C
MINDIA - Minimum diagonal element value
CHLSKY - Cholesky option flag

COMMON /FBSX/ LIIFBS(7),U(7),KIBT(7),GIBT(7),NZFBS,PREC,SIGN

LIIFBS - LII matrix control block
U - U matrix control block
KIBT - KIB matrix control block
GIBT - GIB matrix control block
NZFBS - Number of open core words
PREC - Precision of FBS solution
SIGN - Sign of FBS computation

4. Method: The interior stiffness matrix is decomposed using subroutine SDCOMP

$$[KII] = [LII][LII]^T$$

If the RSAVE flag is set, the decomposition results are saved on the SØF under the LMTX item for substructure ØLDNAM.

Using subroutine FBS, the structure reduction transformation matrix is solved.

$$[LII][LII]^T[GIB] = -[KIB]$$

The G matrix is then saved on the SØF as the GIMS item for substructure OLDNAM.

MODULE FUNCTIONAL DESCRIPTIONS

4.159.8.4 Subroutine Name: MRED2C

1. Entry Point: MRED2C
2. Purpose: To process the \emptyset LDM \emptyset DES option flag.
3. Calling Sequence: CALL MRED2C (K \emptyset DE)

K \emptyset DE - Subroutine operation flag

C \emptyset MM \emptyset N /BLANK/ - See Section 4.159.8.1

C \emptyset MM \emptyset N /SYSTEM/ - See Section 4.159.8.9

4. Method: If \emptyset LDM \emptyset DES is not set, the LAMAMR and PHISS data block are stored on the S \emptyset F as the LAMS and PHIS item for substructure \emptyset LDNAM.

If the \emptyset LDM \emptyset DES option is set, the LAMS and PHIS items are read from the S \emptyset F and stored on scratch files. The scratch files are then switched with the LAMAMR and PHISS input data blocks.

4.159.8.5 Subroutine Name: MRED2D

1. Entry Point: MRED2D
2. Purpose: To calculate the modal mass and stiffness matrices if USERMODE = TYPE2.
3. Calling Sequence: CALL MRED2D

C \emptyset MM \emptyset N /BLANK/ - See Section 4.159.8.1

C \emptyset MM \emptyset N /TW \emptyset / ITW \emptyset (32)

ITW \emptyset - Powers of two

C \emptyset MM \emptyset N /BITPOS/ IDUM5(5),UL,UA,UF,US,UN,IDUM6(10),UB

UL - L bit position

UA - A bit position

UF - F bit position

US - S bit position

UN - N bit position

UB - B bit position

FUNCTIONAL MODULE MRED2 (MODAL SYNTHESIS REDUCTION CALCULATIONS)

COMMON /PACKX/ TYPIN,TYP0UT,IR0W,NR0W,INCR

TYPIN - Type of input variable to pack

TYP0UT - Type of input variable to pack out

IR0W - Row position of first variable to pack

NR0W - Row position of last variable to pack

INCR - Increment amount between variables

COMMON /SYSTEM/ - See Section 4.159.8.9

4. Method:

- a. Count the number of free and fixed points within the boundary set.
- b. If a fixed set exists, process the fixed set (See Section 4.159.8.10).
- c. If a free set exists, partition the PHISS matrix (See Section 4.159.8.11).
- d. Form the HK matrix (See Sections 4.159.8.12-13).
- e. Compute K matrix (See Section 4.159.8.14).
- f. Compute HM matrix (See Section 4.159.8.15).
- g. Output the HORG item to the S0F (See Section 4.159.8.16).
- h. Form the stiffness and mass matrices

$$[(K,M)HH] = [(K,M)] + \left[\begin{array}{c|c} (K,M)BB & 0 \\ \hline 0 & 0 \end{array} \right]$$

and store as the KMTX and MMTX items on the S0F for substructure NEWNAM.

- i. If load data exists, then expand the load data for the modal D0F.

$$[PHH] = \left[\begin{array}{c} PBB \\ \hline 0 \end{array} \right]$$

Save the expanded load data as the PVEC or PAPP item on the S0F for substructure NEWNAM. Store a null matrix as the P0VE or P0AP item on the S0F for substructure 0LDNAM.

MODULE FUNCTIONAL DESCRIPTIONS

4.159.8.6 Subroutine Name: MRED2E

1. Entry Point: MRED2E
2. Purpose: To calculate the modal transformation matrix.
3. Calling Sequence: CALL MRED2E

COMMON /BLANK/ - See Section 4.159.8.1

COMMON /PACKX/ - See Section 4.159.8.5

COMMON /PATX/ - See Section 4.159.8.2

COMMON /MPYADX/ ITRLRA(7),ITRLRB(7),ITRLRC(7),ITRLRD(7),NZ,T,SIGNAB,SIGNC,PREC,SCR

ITRLRA - Matrix A control block

ITRLRB - Matrix B control block

ITRLRC - Matrix C control block

ITRLRD - Matrix D control block

NZ - Number of open core words

T - Transpose flag

SIGNAB - Sign of (A*B)

SIGNC - Sign of C

PREC - Precision of matrix multiplication

SCR - Scratch file

COMMON /BITPOS/ - See Section 4.159.8.5

COMMON /PARMEG/ IA(7),IA11(7),IA21(7),IA12(7),IA22(7),LCR,RULF

IA - A matrix control block

IA11 - A11 matrix control block

IA21 - A21 matrix control block

IA22 - A22 matrix control block

LCR - Number of open core words

RULE - Matrix merge rule

COMMON /UNPAKX/ TYPEU,IR0WU,NR0WU,INCRU

TYPEU - Type of variable to be unpacked

FUNCTIONAL MODULE MRED2 (MODAL SYNTHESIS REDUCTION CALCULATIONS)

IROWU - Row position of first variable unpacked

NROWU - Row position of last variable unpacked

INCRU - Incremental value between unpacked variables

COMMON /SYSTEM/ - See Section 4.159.8.9

4. Method:

a. The frequency data is read from the LAMMR data block and desired modes are selected based on the RANGE and NMAX commands provided by the user such that

for $RANGE = f_1, f_2$ the frequencies f_i are selected with

$$f_1 < f_i < f_2$$

where

$$i = 1, 2, \dots, NMAX.$$

b. A record is built to be appended to the LAMS item to describe the manner in which each mode is used. The usage codes are

0 - rigid body point,

1 - included in modal set,

2 - excluded from modal set because of non-participation, and

3 - excluded from modal set because of RANGE or NMAX.

c. The PHISS data block is read and the desired eigenvectors from Step (a) are retrieved.

$$[PHISS] = [0; PHIAM]$$

d. These vectors are then partitioned.

$$[PHIAM] = \begin{bmatrix} PHIBM \\ \hline PHIIM \end{bmatrix}$$

e. From this, the modal transformation matrix is computed as

$$[HIM] = [PHIIM] - [GIB][PHIBM]$$

MODULE FUNCTIONAL DESCRIPTIONS

f. The columns of the H_{im} matrix are then processed. For each column of the H_{im} and PHI_{im} matrix, if

$$\frac{\left| \left\{ H_{im} \right\} \right|}{\left| \left\{ PHI_{im} \right\} \right|} < \epsilon = 10^{-3}$$

the mode (column) is rejected. An included column is scaled so that the largest term has a magnitude of 1.0.

4.159.8.7 Subroutine Name: MRED2F

1. Entry Point: MRED2F
2. Purpose: To compute freebody effects.
3. Calling Sequence: CALL MRED2F

COMMON /BLANK/ - See Section 4.159.8.1

COMMON /MPYADX/ - See Section 4.159.8.6

COMMON /BITPOS/ - See Section 4.159.8.5

COMMON /PATX/ - See Section 4.159.8.2

COMMON /FBSX/ - See Section 4.159.8.5

COMMON /PACKX/ - See Section 4.159.8.5

COMMON /UNPAKX/ - See Section 4.159.8.6

COMMON /SYSTEM/ - See Section 4.159.8.9

4. Method: The freebody matrix is computed

$$[FAR] = [MAA] [DMR]$$

where $r = 6$.

The F matrix is partitioned to interior and boundary points.

$$[FAR] = \begin{bmatrix} FBR \\ \hline FIR \end{bmatrix}$$

FUNCTIONAL MODULE MRED2 (MODAL SYNTHESIS REDUCTION CALCULATIONS)

The freebody transformation matrix is solved by using subroutine FBS.

$$[LII] \quad [LII]^T [HIR] = - [FIR]$$

Each column of H_{ir} is then scaled so the largest magnitude is 1.0. This is then merged with the modal transformation matrix

$$[HIE] = [HIR \mid HIM]$$

The final H matrix is computed by merging the previously-computed structure reduction transformation matrix (See Section 4.159.8.3).

$$[HGH] = \left[\begin{array}{c|c} I & 0 \\ \hline GIB & HIE \end{array} \right]$$

This matrix is then stored on the SØF as the HØRG item for substructure ØLDNAM.

4.159.8.8 Subroutine Name: MRED2G

1. Entry Point: MRED2G
2. Purpose: To calculate the final structural matrices.
3. Calling Sequence: CALL MRED2G (KODE)

KODE - HØRG matrix input flag

CØMMØN /BLANK/ - See Section 4.159.8.1

CØMMØN /SYSTEM/ - See Section 4.159.8.9

CØMMØN /MPYADX/ - See Section 4.159.8.6

CØMMØN /PACKX/ - See Section 4.159.8.5

CØMMØN /BITPØS/ - See Section 4.159.8.5

CØMMØN /PATX/ - See Section 4.159.8.2

CØMMØN /MPY3TL/ JTRLRA(7),JTRLRB(7),JTRLRE(7),JTRLREC(7),JSCR(3),LKØRE,ICØDE,PREC3

JTRLRA - A matrix control block

JTRLRB - B matrix control block

MODULE FUNCTIONAL DESCRIPTIONS

JTRLRE - E matrix control block
 JTRLRC - C matrix control block
 JSCR - Scratch files
 LKQRE - Length of open core
 ICQDE - Matrix multiplication type
 PREC3 - Precision of matrix multiplication

4. Method:

- a. Set up new substructure on the SØF.
- b. If a stiffness matrix has been provided, the following computations are performed,

$$[KBARBB] = [KBB] + [GIB]^T [KIB],$$

$$[KEE] = [HIE]^T [KII] [HIE],$$

and merged

$$[KHH] = \left[\begin{array}{c|c} KBARBB & 0 \\ \hline 0 & KEE \end{array} \right]$$

The K_{hh} matrix is stored on the SØF as the KMTX item for substructure NEWNAM.

- c. The remaining structural matrices are calculated and stored on the SØF as the MMTX, BMTX, K4MX, and PVEC or PAPP items for substructure NEWNAM.

$$[(M,B,K4)HH] = [HGH]^T [(M,B,K4)AA] [HGH]$$

$$[PHH] = [HGH]^T [PAA]$$

- d. If the load matrix is present, the loads on the interior points, P_i , are partitioned out and stored on the SØF as the PØVE or PØAP item for substructure OLDNAM.

$$[PAA] = \left[\begin{array}{c} PB \\ \hline PII \end{array} \right]$$

FUNCTIONAL MODULE MRED2 (MODAL SYNTHESIS REDUCTION CALCULATIONS)

4.159.8.9 Subroutine Name: MRED2H

1. Entry Point: MRED2H
2. Purpose: To create the reduced substructure new SØF table items.
3. Calling Sequence: CALL MRED2H

CØMMØN /BLANK/ - See Section 4.159.8.1

CØMMØN /SYSTEM/ IDUM5,IPRNTR,IDUM7(6),NLPP,IDUM8(2),LINE

IPRNTR - Printer logical output device number

NLPP - Maximum number of lines per page

LINE - Current number of lines on a page

4. Method: The EQSS item is created by adding the modal degrees of freedom to the boundary data which is on the EQST data block. The new dof will be for the modal points and the freebody points if they exist. They will be stored under a new component basic substructure with the name NEWNAM. The grid point ID's are 1-6 for rigid points and 101, 102, 103,...for modal points. All component codes are 1.

The BGSS item for substructure NEWNAM is created in a similar manner with the new points having a -1 coordinate system ID (i.e., scalar points) and zero coordinates.

The LODS or LØAP, PLT and CSTM items are then created and any output requests are listed.

4.159.8.10 Subroutine Name: MRED2I

1. Entry Point: MRED2I
2. Purpose: To compute the GS matrix for the User mode type 2 option.
3. Calling Sequence: CALL MRED2I (KØDE,NUF,N2)

KØDE - Not used

NUF - Number of free points in the boundary set.

N2 - Not used

MODULE FUNCTIONAL DESCRIPTIONS

COMMON /BLANK/ - See Section 4.159.8.1

COMMON /SYSTEM/ - See Section 4.159.8.9

COMMON /CONDA3/ IDUM5(4),FORPI2

FORPI2 - The value of $4.0 \times \pi^2$

COMMON /PACKX/ - See Section 4.159.8.5

COMMON /UNPAKX/ - See Section 4.159.8.6

4. Method: The mass and frequency are read from the LAMAMR table and the GS matrix is calculated.

$$[GS] = - \begin{bmatrix} & & 0 \\ & 1/K & \\ 0 & & \end{bmatrix} [QSM]^T \quad \text{where } K = M_I W_I^2$$

$$= m_i (2\pi f_i)^2$$

$$(i=1,2,\dots,NMODES)$$

4.159.8.11 Subroutine Name: MRED2J

1. Entry Point: MRED2J

2. Purpose: To partition the PHISS matrix User mode type 2 option.

3. Calling Sequence: CALL MRED2J(NUF,N2)

NUF - Number of free points in the boundary set

N2 - Difference between the number of modes and the number of free points in the boundary set

COMMON /BLANK/ - See Section 4.159.8.1

COMMON /PACKX/ - See Section 4.159.8.5

4. Method: The PHIS matrix is partitioned into free points (PHISS1) and non-free points (PHISS2).

$$[PHISS] = [PHISS1 \mid PHISS2]$$

4.159.8.12 Subroutine Name: MRED2L

1. Entry Point: MRED2L

2. Purpose: To perform preliminary calculations and merges of the HK matrix for the User mode type 2 option.

FUNCTIONAL MODULE MRED2 (MODAL SYNTHESIS REDUCTION CALCULATIONS)

3. Calling Sequence: CALL MRED2L (NUF,N2,NUS,UFBITS)

NUF - Number of free points in the boundary set.

N2 - Difference between number of modes and the number of free points in the boundary set

NUS - Number of fixed points in the boundary set

UFBITS - Not used

COMMON /BLANK/ - See Section 4.159.8.1

COMMON /MPYADX/ - See Section 4.159.8.6

COMMON /PACKX/ - See Section 4.159.8.5

4. Method: The HK matrix components are formed by the following steps.

$$[HK] = \left[\begin{array}{cc|c|cc} -\phi_1^{-1} & \phi & GS & \phi_1^{-1} & -\phi_1^{-1} & \phi_2 \\ \hline 0 & & & 0 & & I \end{array} \right]$$

a. The inverse of the PHISS1 matrix is computed.

b. The PHIGS matrix is calculated.

$$[PHIGS] = - [PHISS1]^{-1} [PHISS] [GS]$$

c. The PHIGS matrix is then merged with two null matrices to form the HKPG matrix.

$$[HKPG] = \left[\begin{array}{c|c} PHIGS & PHISS^{-1} \\ \hline 0 & 0 \end{array} \right]$$

d. The PHIS12 matrix is calculated next.

$$[PHIS12] = - [PHISS1]^{-1} [PHISS2]$$

e. An identify matrix is generated next and merged with the PHIS12 matrix.

$$[PHI12I] = \left[\begin{array}{c} PHIS12 \\ \hline I \end{array} \right]$$

MODULE FUNCTIONAL DESCRIPTIONS

4.159.8.13 Subroutine Name: MRED2M

1. Entry Point: MRED2M
2. Purpose: To form the HK matrix for the User mode type 2 option.
3. Calling Sequence: CALL MRED2M (NUF,N2,NUS)

NUF - Number of free points in the boundary set

N2 - Difference between the number of modes and the number of free points in the boundary set

NUS - Number of fixed points in the boundary

COMMON /BLANK/ - See Section 4.159.8.1

COMMON /PACKX/ - See Section 4.159.8.5

4. Method: If free points exists, HK is formed by merging the HKPG and PHI12I matrices.

$$[HK] = [HKPG \mid PHI12I]$$

If no free points exist, HK is formed by merging a zero and an identify matrix.

$$[HK] = [0 \mid I]$$

4.159.8.14 Subroutine Name: MRED2N

1. Entry Point: MRED2N
2. Purpose: To calculate the K matrix for the User mode type 2 option.
3. Calling Sequence: CALL MRED2N

COMMON /BLANK/ - See Section 4.159.8.1

COMMON /CONDAS / - See Section 4.159.8.10

COMMON /MPY3TL/ - See Section 4.159.8.8

COMMON /PACKX/ - See Section 4.159.8.5

4. Method: The mass and frequency are read from the LAMAMR table and the K matrix is generated.

$$[K] = [HK]^T \begin{bmatrix} & 0 \\ K & \\ 0 & \end{bmatrix} [HK] \quad \text{where } K = M_I W_I^2$$

$$= M_i (2 f_i)^2$$

$$(i=1,2,\dots,NMODES)$$

FUNCTIONAL MODULE MRED2 (MODAL SYNTHESIS REDUCTION CALCULATIONS)

4.159.8.15 Subroutine Name: MRED2Q

1. Entry Point: MRED2Q
2. Purpose: To form the M matrix for the User mode type 2 option.
3. Calling Sequence: CALL MRED2Q (NUS)

NUS - Number of fixed points in the boundary set.

COMMON /BLANK/ - See Section 4.159.8.1

COMMON /MPY3TL/ - See Section 4.159.8.8

COMMON /PACKX/ - See Section 4.159.8.5

4. Method:

- a. If fixed points exist, the GS matrix is merged with a zero matrix and added to the HK matrix to form the HM matrix.

$$[HM] = [HK] + [GS \mid 0 \mid 0]$$

- b. If no fixed points exist, the HM matrix is equal to the HK matrix.

$$[HM] = [HK]$$

- c. The mass data is read from the LAMAMR table and the M matrix is formed.

$$[M] = [HM]^T \begin{bmatrix} & & 0 \\ & M & \\ 0 & & \end{bmatrix} [HM] \quad (i=1,2,\dots,NMODES)$$

where $M = M_i$

4.159.8.16 Subroutine Name: MRED2P

1. Entry Point: MRED2P
2. Purpose: To output the HAB matrix to the SOF as the HORG item for the User modes type 2 option.
3. Calling Sequence: CALL MRED2P (NUS,NUF,N2)

NUS - Number of fixed points in the boundary set

NUF - Number of free points in the boundary set

MODULE FUNCTIONAL DESCRIPTIONS

N2 - Not used

COMMON /BLANK/ - See Section 4.159.8.1

COMMON /PACKX/ - See Section 4.159.8.5

COMMON /SYSTEM/ - See Section 4.159.8.9

4. Method: The HAB matrix is formed and output to the SØF as the HORG item for substructure ØLDNAM.

$$[HAB] = [I \ ; \ 0]$$

4.159.8.17 Subroutine Name: GMMERG

1. Entry Point: GMMERG
2. Purpose: To perform a general matrix merge
3. Calling Sequence: CALL GMMERG (FILEA,FILE11,FILE21,FILE12,FILE22,RPART,CPART,NSUB,MRGTYP,CØRE,LCØRE)

FILEA - Resulting merged file number

FILE11 - Subset 0,0 file number

FILE21 - Subset 0,1 file number

FILE12 - Subset 1,0 file number

FILE22 - Subset 1,1 file number

RPART - Row merge vector file number (maybe 0 if no row merge needed)

CPART - Column merge vector file number (may be 0 if no column merge needed)

NSUB - Array of dimension 4 holding the number of columns in RPART 0 and 1 subsets and the number of rows in the CPART 0 and 1 subset respectively.

MRGTYP - Merge type (equals 1 for a resulting square matrix and 2 for a resulting rectangular matrix).

CØRE - Beginning address of open core

LCØRE - Number of words in open core

COMMON /PARMEG/ - See Section 4.159.8.6

4. Method: The matrix control blocks are set up and the type of merge is called to MERGE based on whether or not RPART and CPART are provided.

FUNCTIONAL MODULE MRED2 (MODAL SYNTHESIS REDUCTION CALCULATIONS)

$$[FILEA] = \left[\begin{array}{c|c} FILE11 & FILE12 \\ \hline FILE21 & FILE22 \end{array} \right]$$

4.159.8.18 Subroutine Name: GMPRTN

1. Entry Point: GMPRTN
2. Purpose: To perform a general matrix partition.
3. Calling Sequence: CALL GMPRTN (FILEA,FILE11,FILE21,FILE12,FILE22,RPART,CPART,NSUB0,NSUB1,CØRE,LCØRE)

FILEA - File number to be partitioned

FILE11 - 0,0 subset partition file number

FILE21 - 1,0 subset partition file number

FILE12 - 0,1 subset partition file number

FILE22 - 1,1 subset partition file number

RPART - Row partition vector filename (may be 0 if no row partition needed)

CPART - Column partition vector file number (may be 0 if no column partition needed)

NSUB0 - Number of entries in 0 subset

NSUB1 - Number of entries in 1 subset

CØRE - Beginning address of open core

LCØRE - Length of open core

COMMON /PARMEG/ - See Section 4.159.8.6

4. Method: The matrix control blocks are set up and the type of partition is called to PARTN based on whether or not RPART and CPART are provided.

$$[FILEA] = \left[\begin{array}{c|c} FILE11 & FILE12 \\ \hline FILE21 & FILE22 \end{array} \right]$$

4.159.9 Design Requirements

Ten scratch files are needed. They are allocated in the following manner.

MODULE FUNCTIONAL DESCRIPTIONS

FILE ALLOCATION WITH NO USER MODES

File Number	File Number Used in Subroutine MRED--							
	2	2A	2B	2C	2E	2F	2G	2H
CASECC	101							
USETMR		105			105	105	105	
KAA		106						
KBB		301					301	
KIB		302	302				302	
KII		303	303				303	
UPRT		305					301	
LII			305			305		
GIB			306		308	304	304	
LAMAMR				102	102			
PHIS				103				
LAMS				305				
PHISS				306	306			
PAA							110	
KHH							201	
PHIIM					307			
PHIAM					308			
HIM					308	308		
PHIBM					309			
MAA						107		
DMR						111		
HGH						308	308	
HIE						307		
HIR						309		
FAR						309		
FIR						310		
PJVE							206	
KBARBB							305	
KEE							306	
HIE							307	
EQST								104

FUNCTIONAL MODULE MRED2 (MODAL SYNTHESIS REDUCTION CALCULATIONS)

FILE ALLOCATION WITH USER MODES TYPE 2

File Name	File Number Used in Subroutine MRED--									
	2	2D	2I	2J	2L	2M	2N	2O	2P	2H
CASECC	101									
USETMR		105								
KBB		106								
LAMAMR			102				102	102		
PHISS				303	306					
MBB		107								
PAA		110								
KHH		201								
MHH		202								
PHH		205								
K		303					303			
M		310						310		
QSM			112							
GS			307		307			307		
PHISS1				308	308					
PHISS2				309	309					
PHIGS					302					
PHIS12					302					
PHI12I					308	308				
HK						302	302	302		
HKPG						303				
HM								309		
GSZERØ								310		
HAB									302	
EQST										104

MODULE FUNCTIONAL DESCRIPTIONS

Open core resides in common block /MRED27/ and is defined as follows:

During modal transformation matrix calculation		All other times	
COMMON /MRED2Z/		COMMON /MRED2Z/	
NMØDES	Mode index array		
NMØDES	Mode usage array		
NRØW	HIM matrix column		
NRØW	PHIIM matrix column		
		3 SØF buffers	
		3 GINØ buffers	

4.159.10 Diagnostic Messages

Diagnostic messages 3001-3003, 3008, 3037, 6101-6103, 6211, 6215, 6311, 6518, and 6632-6634 are issued.

MODULE FUNCTIONAL DESCRIPTIONS

4.160 FUNCTIONAL MODULE CMRED2 (MODAL SYNTHESIS COMPLEX REDUCTION CALCULATIONS)

4.160.1 Entry Point: CMRED2

4.160.2 Purpose:

To perform the computations for the complex modal reduce command.

4.160.3 DMAP Calling Sequence:

CMRED2 CASECC,LAMAMR,PHISSR,PHISSL,EQST,USETMR,KAA,MAA,BAA,K4AA,PAA/KHH,MHH,BHH,K4HH,PHH,
PØVE/STEP/S,N,DRY/PØPT \$

4.160.4 Input Data

4.160.4.1 GINØ Data Blocks

CASECC - Case Control Data

LAMAMR - Eigenvalue table for substructure being reduced

PHISSR - Right hand eigenvectors for substructure being reduced

PHISSL - Left hand eigenvectors for substructure being reduced

EQST - EQSS data for boundary set for substructure being reduced

USETMR - USET table for reduced substructure

KAA - Substructure stiffness matrix

MAA - Substructure mass matrix

BAA - Substructure viscous damping matrix

K4AA - Substructure structure damping matrix

PAA - Substructure load matrix

- Notes:
1. LAMAMR, PHISSR and PHISSL may be purged if the modes exist on the SØF.
 2. KAA,...,PAA may be purged depending on the substructure ØPTION command.
 3. PHISSL may be purged if the matrices are symmetric.

4.160.4.2 SØF Items

LAMS - Eigenvalue table for original substructure

PHIS - Eigenvector table for original substructure

LMTX - Stiffness decomposition product for original substructure

MODULE FUNCTIONAL DESCRIPTIONS

GIMS - G transformation matrix for boundary points for original substructure

HØRG - H transformation matrix for original substructure

4.160.5 Output Data

4.160.5.1 GINØ Data Blocks

KHH - Reduced stiffness matrix

MHH - Reduced mass matrix

BHH - Reduced viscous damping matrix

K4HH - Reduced structure damping matrix

PHH - Reduced load matrix

PØVE - Interior point load matrix

Note: KHH,...,PHH may be purged depending on input data blocks provided.

4.160.5.2 SØF Items

LAMS - Eigenvalue table for original substructure

PHIS - Right hand eigenvector table for original substructure

PHIL - Left hand eigenvector table for original substructure

GIMS - G transformation matrix for boundary points for original substructure

HØRG - Right hand H transformation matrix for original substructure

HLFT - Left hand H transformation matrix for original substructure

UPRT - Partitioning vector for CREDUCE for original substructure

PØVE - Internal point loads for original substructure

PØAP - Internal points appended loads for original substructure

EQSS - substructure equivalence table for reduced substructure

BGSS - Basic grid point definition table for reduced substructure

CSTM - Coordinate system transformation matrices for reduced substructure

LØDS - Load set data for reduced substructure

LØAP - Appended load set data for reduced substructure

PLTS - Plot set data for reduced substructure

KMTX - Stiffness matrix for reduced substructure

MMTX - Mass matrix for reduced substructure

PVEC - Load matrix for reduced substructure

FUNCTIONAL MODULE CMRED2 (MODAL SYNTHESIS COMPLEX REDUCTION CALCULATIONS)

PAPP - Appended load matrix for reduced substructure

BMTX - Viscous damping matrix for reduced substructure

K4MX - Structure damping matrix for reduced substructure

4.160.6 Parameters

STEP - input - integer - no default. Identifies the Case Control record which contains the CREDUCE data.

DRY - output - integer. A flag which is set during module execution if error conditions prevent a requested GO operation.

PØPT - input - BCD - no default. Set to PVEC or PAPP for options P and PA respectively.

4.160.7 Method

The following logical sequence is performed by the CMRED2 module.

- a. Process Case Control (Subroutine CMRD2)
- b. Process stiffness matrix (Subroutine CMRD2A)
- c. The following steps are performed once for a symmetric reduction and twice for an unsymmetric reduction.

Process Guyan reduction (Subroutine CMRD2C)

Process ØLDMØDES option (Subroutine CMRD2B)

Calculate modal transformation matrix (Subroutine CMRD2D), and

Calculate the H transformation matrix (Subroutine CMRD2E).

- d. Calculate structural matrices (Subroutine CMRD2F).
- e. Generate new SØF table items (Subroutine CMRD2G).

4.160.8 Subroutines

4.160.8.1 Subroutine Name: CMRD2

1. Entry Point: CMRD2
2. Purpose: Driver which performs the computations for the complex modal reduce command.
3. Calling Sequence: CALL CMRD2

MODULE FUNCTIONAL DESCRIPTIONS

COMMON /BLANK/ STEP, DRY, PØPT, GBUF1, GBUF2, GBUF3, SBUF1, SBUF2, SBUF3, INFILE(11), ØTFILE(6),
ISCR(11), KØRLEN, KØRBGN, ØLDNAM(2), NEWNAM(2), SYMTRY, RANGE(2), NMAX, IØ, MØDES,
RSAVE, LAMSAP, MØDPTS, MØDLEN

STEP - Control Data CASECC Record (integer)
PØPT - PVEC or PAPP option flag (BCD)
DRY - Module operation flag (integer)
GBUF - GINO buffers
SBUF - SØF buffers
INFILE - Input file numbers
ØTFILE - Output file numbers
ISCR - Array of scratch file numbers
KØRLEN - Length of open core
KØRBGN - Beginning address of open core
ØLDNAM - Name of substructure being reduced
NEWNAM - Name of reduced substructure
SYMTRY - Symmetry flag
RANGE - Range of frequencies to be used
NMAX - Maximum number of frequencies to be used
IØ - IØ options flag
MØDES - ØLDMØDES option flag
RSAVE - Save reduction product flag
LAMSAP - Beginning address of mode use description array
MØDLEN - Length of mode use array
MØDPTS - Number of modal points

COMMON /SYSTEM/ SYSBUF, IPRNTR

SYSBUF - System buffer size
IPRNTR - Printer logical output device number

4. Method:

- a. Initialize GINØ and SØF files and buffers.
- b. Initialize Case Control parameters.
- c. Process Case Control data block.

FUNCTIONAL MODULE CMRED2 (MODAL SYNTHESIS COMPLEX REDUCTION CALCULATIONS)

The allowable case control mnemonics for the CMRD2 operation are:

NAMA - Name of substructure to be reduced

NAMB - Name of resultant reduced substructure

RANG - Eigenvalue mode frequency range

NMAX - Maximum number of mode frequencies with range to be used

ØUTP - Selective output control flag

ØLDM - ØLDMØDES option flag

RSAY - Save reduction products flag

These values are extracted from the CASECC input block.

d. Process stiffness matrix (See Section 4.160.8.2).

e. Determine whether reduction is symmetric or unsymmetric.

f. If the corresponding H transformation matrix is not present, the following steps are performed once for a symmetric reduction and twice for an unsummetric reduction.

Process ØLDMØDES option (See Section 4.160.8.3)

Perform Guyan reduction (See Section 4.160.8.4)

Calculate the modal transformation matrix (See Section 4.160.8.5)

Calculate the corresponding H transformation matrix (See Section 4.160.8.6)

g. Calculate the structural matrices (See Section 4.160.8.7).

h. Process the new SØF table items (See Section 4.160.8.8).

4.160.8.2 Subroutine Name: CMRD2A

1. Entry Point: CMRD2A

2. Purpose: To partition the stiffness matrix into boundary and interior points and save the partitioning vector on the SØF as the UPRT item.

3. Calling Sequence: CALL CMRD2A

CØMMØN /BLANK/ - See Section 4.160.8.1

CØMMØN /BITPØS/ - IDUM4(9),UN,IDUM5(10),UB,UI

UN - N bit position

UB - B bit position

UI - I bit position

MODULE FUNCTIONAL DESCRIPTIONS

COMMON /PATX/ LCORE, NSUB(3), FUSET

LCORE - Length of open core

NSUB - Number of rows in 0.1, and neither 0.1 subset

FUSET - USET array

COMMON /SYSTEM/ - See Section 4.160.8.8

4. Method: The stiffness matrix is partitioned to boundary and interior points

$$[KAA] = \begin{bmatrix} KBB & KBI \\ KIB & KII \end{bmatrix}$$

using subroutines CALCV and PARTN.

The partitioning vector calculated is saved on the SØF as the UPRT item for substructure ØLDNAM.

4.160.8.3 Subroutine Name: CMRD2B

1. Entry Point: CMRD2B
2. Purpose: To process the ØLDMØDES option flag.
3. Calling Sequence: CALL CMRD2B (KØDE)

KØDE - Subroutine operation flag

COMMON /BLANK/ - See Section 4.160.8.1

COMMON /SYSTEM/ - See Section 4.160.8.8

4. Method: If ØLDMØDES is not set, the LAMAMR and PHISSR or PHISSL data block are stored on the SØF as the LAMS and PHIS or PHIL item for substructure ØLDNAM.

If the ØLDMØDES option is set, the LAMS and PHIS or PHIL items are read from the SØF and stored on scratchfiles. The scratch files are then switched with the LAMAMR and PHISSR or PHISSL input data blocks.

4.160.8.4 Subroutine Name: CMRD2C

1. Entry Point: CMRD2C
2. Purpose: To perform the Guyan reduction on the structure points.

FUNCTIONAL MODULE CMRED2 (MODAL SYNTHESIS COMPLEX REDUCTION CALCULATIONS)

3. Calling Sequence: CALL CMRD2C (ITER)

ITER - Reduction type flag

CØMMØN /BLANK/ - See Section 4.160.8.1

CØMMØN /SFACT/ KIIT(7),LIIT(7),ISCRQ(7),ISCRA,ISCRB,NZSF,DETR,DETI,PØWER,ISCRØ,MINDIA,
CHLSKY

KIIT - Matrix control block for KII

LIIT - Matrix control block for LII

ISCRQ - Matrix control block for a scratch file

ISCRA - Scratch file A

ISCRB - Scratch file B

NZSF - Number of open core words

DETR - Determinant of LII (real)

DETI - Determinate of LII (imaginary)

PØWER - Power of LII

ISCRØ - Scratch file C

MINDIA - Minimum diagonal element value

CHLSKY - Cholesky option flag

CØMMØN /CDCMPX/ KIITC(7),LIITC(7),UIITC(7),SCR(3),DET(2),PØWERC,NX,MINDC,B,BBAR

KIITC - Matrix control block for KII

LIITC - Matrix control block for LII

UIITC - Matrix control block for UII

SCR - Scratch file numbers

DET - Determinate of LII

PØWERC - Power of LII

NX - Number of open core words

MINDC - Minimum diagonal element value

B - B calculation flag

BBAR - \bar{B} calculation flag

CØMMØN /FBSX/ LIIFBS(7),U,(7),KIBT(7),GIBT(7),NZFBS,PREC,SIGN

LIIFBS - LII matrix control block

MODULE FUNCTIONAL DESCRIPTIONS

U - U matrix control block
 KIBT - KIB matrix control block
 GIBT - GIB matrix control block
 NZFBS - Number of open core words
 PREC - Precision of FBS solution
 SIGN - Sign of FBS computation

COMMON /GFBSX/ LIGFBS(7),UGFBS(7),KIGFBS(7),GIBFBS(7),NZGFBS,PREC1,ISIGN

LIGFBS - Matrix control block for LII
 UGFBS - Matrix control block for UII
 KIGFBS - Matrix control block for KII
 GIBFBS - Matrix control block for GIB
 NZGFBS - Number of open core words
 PREC1 - Precision of resulting solution
 ISIGN - Sign of solution

COMMON /TRNSPX/ ATRLR(7),ATTRLR(7),LCØRE,NSCRTH,ISCRTH(8)

ATRLR - Matrix control block of matrix to be transposed
 ATTRLR - Matrix control block of transposed matrix
 LCØRE - Length of open core
 NSCRTH - Number of scratch files
 ISCRTH - Scratch file numbers

COMMON /SYSTEM/ - See Section 4.160.8.8

4. Method: The interior stiffness matrix is decomposed using either subroutine SDCØMP for symmetric matrix decompositions or CDCØMP for unsymmetric matrix decompositions.

$$\text{SYMMETRIC: } [KII] = [LII][LII]^T$$

$$\text{UNSYMMETRIC: } [KII] = [LII][UII]$$

If the RSAVE flag is set, the decomposition results L_{ij} are saved on the SØF under the LMTX item for substructure ØLDNAM.

FUNCTIONAL MODULE CMRED2 (MODAL SYNTHESIS COMPLEX REDUCTION CALCULATIONS)

Using subroutine FBS or GFBS, the structure reduction transformation matrix is solved.

$$\text{SYMMETRIC: } [LII][LII]^T[GIB] = -[KIB]$$

$$\text{UNSYMMETRIC: } [LII][UII][GIB] = -[KIB]$$

If the reduction is unsymmetric (unrelated to whether KII is symmetric or not), the above steps are repeated using KII^T and solving for GIBBAR with KBI^T . The GIB matrix is then saved on the SØF as the GIMS item for substructure ØLDNAM.

4.160.8.5 Subroutine Name: CMRD2D

1. Entry Point: CMRD2D
2. Purpose: To calculate the modal transformation matrix.
3. Calling Sequence: CALL CMRD2D (ITER)

ITER - Reduction type flag

CØMMØN /BLANK/ - See Section 4.160.8.1

CØMMØN /PACKX/ TYPIN,TYPØUT,IRØW,NRØW,INCR

TYPIN - Type of input variable to pack

TYPØUT - Type of input variable to pack out

IRØW - Row position of first variable to pack

NRØW - Row position of test variable to pack

INCR - Increment amount between variables

CØMMØN /PATX/ - See Section 4.160.8.2

CØMMØN /MPYADX/ ITRLRA(7),ITRLRB(7),ITRLRC(7),ITRLRD(7),NZ,T,SIGNAB,SIGNC,PREC,SCR

ITRLRA - Matrix A control block

ITRLRB - Matrix B control block

ITRLRC - Matrix C control block

ITRLRD - Matrix D control block

NZ - Number of open core words

MODULE FUNCTIONAL DESCRIPTIONS

T - Transpose flag
SIGNAB - Sign of (A*B)
SIGNC - Sign of C
PREC - Precision of matrix multiplication
SCR - Scratch file

COMMON /BITPOS/ - See Section 4.160.8.2

COMMON /PARMEG/ IA(7),IA11(7),IA21(7),IA22(7),LCR,RULE

IA - A matrix control block
IA11 - A11 matrix control block
IA21 - A21 matrix control block
IA12 - A12 matrix control block
IA22 - A22 matrix control block
LCR - Number of open core words
RULE - Matrix merge rule

COMMON /UNPAKX/ TYPEU,IR0WU,NR0WU,INCRU

TYPEU - Type of variable to be unpacked
IR0WU - Row position of first variable unpacked
NR0WU - Row position of last variable unpacked
INCRU - Incremental value between unpacked variables

COMMON /SYSTEM/ - See Section 4.160.8.8

4. Method:

a. The eigenvalues P are read from the LAMAMR data block and desired modes are selected based on the RANGE and NMAX commands provided by the user, such that

for RANGE = f_1, f_2 the frequencies f_i are selected with $f_1 < I_m(P) < f_2$
where $i = 1, 2, \dots, NMAX$.

All tests are performed on the imaginary part of the root.

b. A record is built to be appended to the LAMS item to describe the manner in which each mode is used. The usage codes are

1 - included in modal set,

FUNCTIONAL MODULE CMRED2 (MODAL SYNTHESIS COMPLEX REDUCTION CALCULATIONS)

2 - excluded from modal set because of non-participation, and

3 - excluded from modal set because of RANGE or NAMX.

c. The PHISSR data block is read and the desired eigenvectors from Step (a) are retrieved.

$$[PHISSR] = [0 \mid PHIAM]$$

d. These vectors are then partitioned.

$$[PHIAM] = \begin{bmatrix} PHIBM \\ \text{-----} \\ PHIIM \end{bmatrix}$$

e. From this, the modal transformation matrix is computed as

$$[HIM] = [PHIIM] - [GIB][PHIBM].$$

f. The columns of the H_{im} matrix are then processed. For each column of the H_{im} and PHI_{im} matrix, if

$$\frac{\| \{ H_{im} \} \|}{\| \{ PHI_{im} \} \|} < \epsilon = 10^{-3}$$

the mode (column) is rejected. An included column is scaled so that the largest term has a magnitude of 1.0. If the reduction is unsymmetric, the above steps are repeated using the PHISSL matrix and solving for HIMBAR using GIBBAR.

4.160.8.6 Subroutine Name: CMRD2E

1. Entry Point: CMRD2E

2. Purpose: To calculate the H transformation matrix.

3. Calling Sequence: CALL CMRD2E(ITER)

ITER - Reduction type flag

COMMON /BLANK/ - See Section 4.160.8.1

MODULE FUNCTIONAL DESCRIPTIONS

COMMON /PACKX/ - See Section 4.160.8.5

COMMON /UNPAKX/ - See Section 4.160.8.5

COMMON /SYSTEM/ - See Section 4.160.8.8

4. Method: The final H matrix is computed by merging the previously-computed structure reduction transformation matrix (See Sections 4.160.8.4-5).

$$[HGH] = \begin{bmatrix} I & 0 \\ \hline GIB & HIM \end{bmatrix}$$

This matrix is then stored on the SØF as the HØRG item for substructure ØLDNAM.

If the reduction is unsymmetric, the HIMBAR and GIBBAR are merged to generate the HGHBAR matrix which is stored on the SØF as the HLFT item for substructure ØLDNAM.

4.160.8.7 Subroutine Name: CMRD2F

1. Entry Point: CMRD2F
2. Purpose: To calculate the final structural matrices.
3. Calling Sequence: CALL CMRD2F (KØDE)

KØDE - H transformation matrix flag

COMMON /BLANK/ - See Section 4.160.8.1

COMMON /SYSTEM/ - See Section 4.160.8.8

COMMON /MPYADX/ - See Section 4.160.8.5

COMMON /PACKX/ - See Section 4.160.8.5

COMMON /BLTPØS/ - See Section 4.160.8.2

COMMON /PATX/ - See Section 4.160.8.2

COMMON /MPY3TL/ JTRLRJ(7),JTRLRB(7),JTRLRE(7),JTRLRC(7),JSCR(3),LKØRG,ICØDE,PREC2

JTRLRA - A matrix control block

JTRLRB - B matrix control block

JTRLRE - E matrix control block

JTRLRC - C matrix control block

JSCR - Scratch files

LKØRE - Length of open core

FUNCTIONAL MODULE CMRED2 (MODAL SYNTHESIS COMPLEX REDUCTION CALCULATIONS)

ICODE - Matrix multiplication type

PREC3 - Precision of matrix multiplication

4. Method:

a. Set up new substructure on the SØF.

b. If a stiffness matrix has been provided, the following computations are performed,

$$[KBARBB] = [KBB] + [GIB]^T[KIB],$$

$$[KMM] = [HIM]^T[KII][HIM],$$

and merged

$$[KHH] = \left[\begin{array}{c|c} KBARBB & 0 \\ \hline 0 & KMM \end{array} \right].$$

The K_{hh} matrix is stored on the SØF as the KMTX item for substructure NEWNAM.

c. The remaining structural matrices are calculated and stored on the SØF as the MMTX, BMTX, K4MX, and PVEC or PAPP items for substructure NEWNAM.

$$[(M,B,K4)HH] = [HGH(BAR)]^T[(M,B,K4)AA][HGH]$$

$$[PHH] = [HGH(BAR)]^T[PAA]$$

d. If the load matrix is present, the loads on the interior points P_i are partitioned out and stored on the SØF as the PØVE or PØAP item for substructure ØLDNAM.

$$[PAA] = \left[\begin{array}{c} PB \\ \hline PII \end{array} \right]$$

If the reduction is unsymmetric, the GIBBAR matrix is used in place of the GIB, the HIMBAR instead of the HIM matrix, and the left hand side HGH is replaced with the HGHBAR matrix.

4.160.8.8 Subroutine Name: CMRD2G

1. Entry Point: CMRD2G

MODULE FUNCTIONAL DESCRIPTIONS

2. Purpose: To create the reduced substructure new SØF table items.
3. Calling Sequence: CALL CMRD2G

CØMMØN /BLANK/ - See Section 4.160.8.1

CØMMØN /SYSTEM/ IDUM5,IPRNTR,IDUM7(6),NLPP,IDUM8(2),LINE

IPRNTR - Printer logical output device number

NLPP - Maximum number of lines per page

LINE - Current number of lines on a page

4. Method: The EQSS item is created by adding the modal degrees of freedom to the boundary data which is on the EQST data block. The new dof will be for the modal points and they will be stored under a new component basic substructure with the name NEWNAM. The grid point ID's are 101, 102, 103,...and the component codes are 1.

The BGSS item is created in a similar manner with the new points having a -1 coordinate system ID (i.e., scalar points) and zero coordinates.

The LØDS or LØAP, PLT and CSTM items are then created and any output requests are listed.

4.160.8.9 Subroutine Name: GMMERG

See Section 4.160.8.17

4.160.8.10 Subroutine Name: GMPRTN

See Section 4.160.8.18

FUNCTIONAL MODULE CMRED2 (MODAL SYNTHESIS COMPLEX REDUCTION CALCULATIONS)

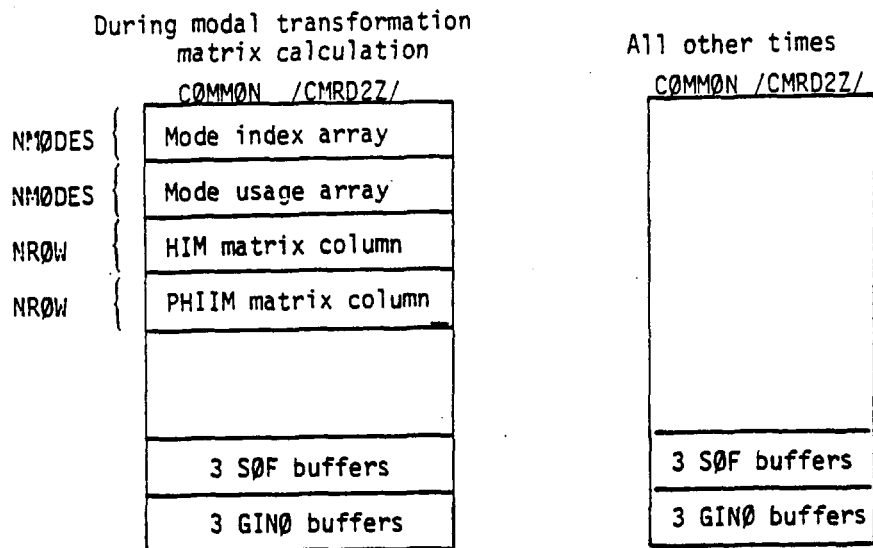
4.160.9 Design Requirements

Eleven scratch files are needed. They are allocated in the following manner.

File Number	File Number Used in Subroutine CMRD--							
	2	2A	2B	2C	2D	2E	2F	2G
CASECC	101							
PHISSL	104		104		104			
USETMR		106			106		106	
KAA		107						
KBB		301					301	
KIB		302		302			302	
KBI		303		303				
KII		304		304			304	
UPRT		305				307	301	
LAMAMR			102		102			
PHISSR			103		103			
LAMS			305		305			
PHISL			306		306			
LII				308				
UII				309				
GIB				311	308	306	303	
GIBBAR				311	311	311	311	
PHIAM					308			
PHIBM					309			
PHIIM					306			
HIMBAR					308	308	308	
HIM					310	310	310	
HGHBAR						309	309	
HGH						309	308	
PAA							111	
KHH							201	
PØVE							206	
KBARBB							305	
KMM							306	
EQST								105

MODULE FUNCTIONAL DESCRIPTIONS

Open core resides in common block /CMRD2Z/ and is defined as follows:



4.160.10 Diagnostic Messages

Diagnostic messages 3001-3003, 3008, 3037, 6101-6103, 6211, 6215, 6311, 6518, 6632, and 6635 are issued.

FUNCTIONAL MODULE EMA (ELEMENT MATRIX ASSEMBLER)

4.161 FUNCTIONAL MODULE EMA1 (ELEMENT MATRIX ASSEMBLER)

4.161.1 Entry Point: EMA1

4.161.2 Purpose

The purpose of this module is to superimpose matrices corresponding to elements into a structural matrix corresponding to all degrees of freedom at all grid points.

4.161.3 DMAP Calling Sequence

EMA1 GPECT,KDICT,KELM,SIL,ECT/KGG,GPST/C,N,NØK4/C,N,WTMASS \$

4.161.4 Input Data Blocks

GPECT - Grid Point Element Connection Table

KDICT - Element Matrix Dictionaries

KELM - Element Matrix Partitions

SIL - Scalar Index List

ECT - Element Connection Table

4.161.5 Output Data Blocks

KGG - Assembled Structural Matrix

GPST - Grid Point Singularity Table

Note: GPST may be purged.

4.161.6 Parameters

NØK4 - Input, integer, default = -1. Flag which specifies whether damping factor is to be used in assembling matrix (-1 ignores factor).

WTMASS - Input, real, default = 1.0. Constant by which all element matrix terms are multiplied.

MODULE FUNCTIONAL DESCRIPTIONS

4.161.7 Method

EMA1 is divided into three phases. The first phase of EMA1 consists of building a modified element dictionary file on SCRATCH2. This is accomplished in the following steps:

1. The contents of the first record of the SIL data block is read into an area at the end of working storage.
2. The ECT and KDICT data blocks are read in synchronously. For each element type on KDICT, a 3-word record header is written on SCRATCH2:
 1. Element type
 2. Number of words per entry (n)
 3. Number of grid points per entry (m)
3. For each element within an element type, an n-word entry is written on SCRATCH2:
 1. Element ID (internal number)
 2. Form of column partitions (1 = rectangle, 2 = diagonal)
 3. Number of terms per column partition
 4. Scalar code defining DØF per grid point
 5. g_e (real damping factor)
 6. Internal index of 1st grid point
 7. GINØ address of 1st column partition

...

 - n-1. Internal index of last grid point
 - n. GINØ address of last column partition

Notes:

- a. Grid points are in sort by internal index
- b. A zero indicates a missing grid point
- c. Any zeros are last in list

The second phase of EMA1 consists of building a data block on SCRATCH3 which contains the defined grid point connection information. This data block is derived from the GPECT data block and has the following format:

1. For each grid/scalar point in the problem, two records are written. The first record contains 6 words:

FUNCTIONAL MODULE EMA (ELEMENT MATRIX ASSEMBLER)

1. internal index of grid/scalar point
 2. DOF of point (1 = scalar, 6 = grid)
 3. maximum DOF of connected points (0 = no connections)
 4. number of connected points
 5. index of first connected point
 6. index of last connected point
2. The second record of each pair is a packed column (written by PACK) which contains a non-zero term for each connected point.

For the assembly (last) phase, EMA1 makes one pass of the SCRATCH2 data block and assembles as many columns of KGG as can be allocated in open core. Allocation of open core for the assembly phase is shown in Figure 1.

MODULE FUNCTIONAL DESCRIPTIONS

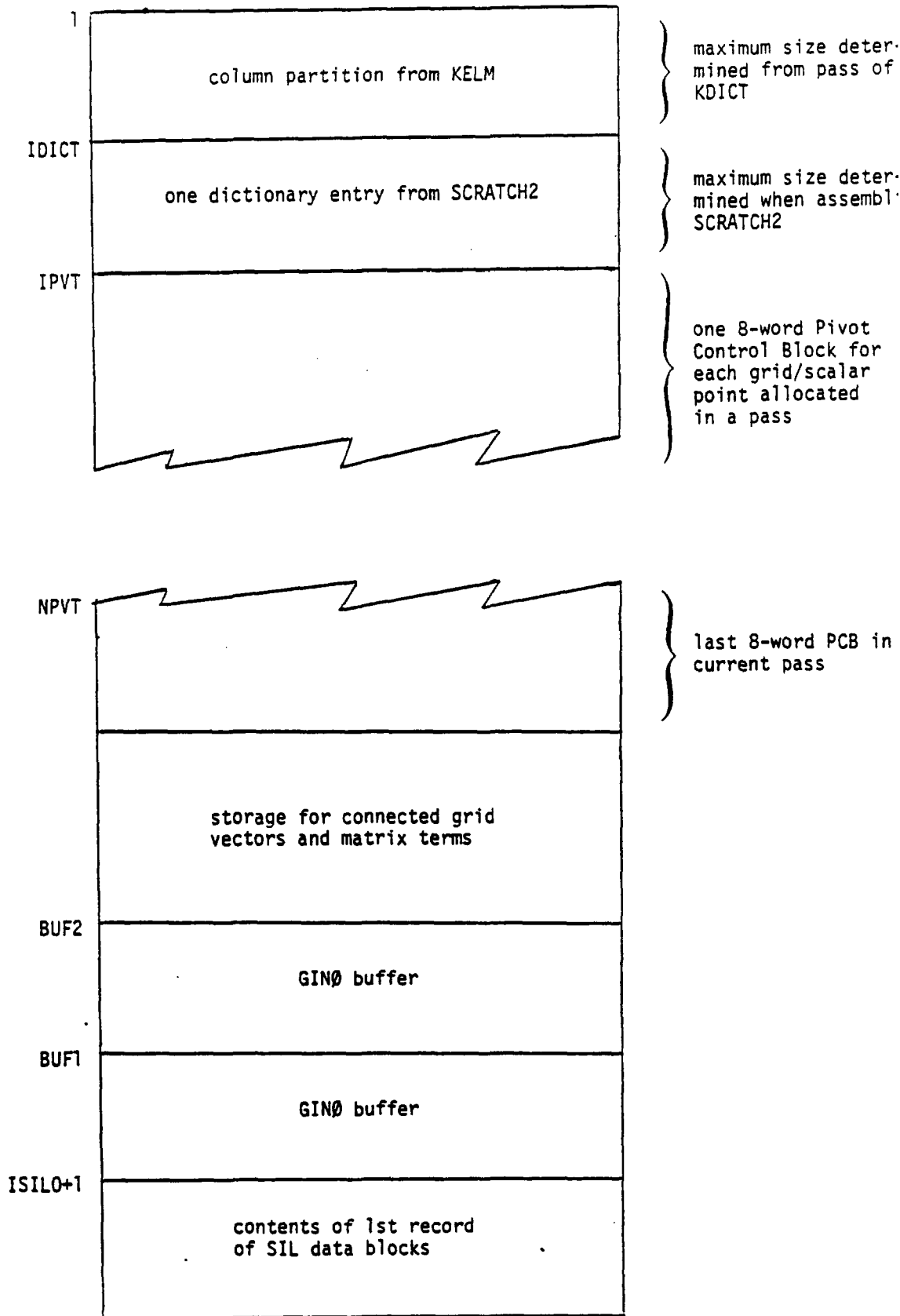


Figure 1. Allocation of Open Core for the Assembly Phase of EMA1

FUNCTIONAL MODULE EMA (ELEMENT MATRIX ASSEMBLER)

The first six words of an 8-word Pivot Control Block (PCB) are read from the first of the 2-record pair on SCRATCH3. The 7th word of the PCB is a pointer to the Connected Grid Vector and the 8th word is a pointer to the matrix terms.

The Connected Grid Vector (CGV) contains one word for each grid or scalar point between the first and last points connected to the pivot point. The contents of each word in the CGV is zero if the point is not connected to the pivot, or the relative position in the column of matrix terms of the connected point if the point is connected to the pivot.

For example, assume point 5 has points 3, 5, 6 and 8 connected to it. The DDF of each connected point is 6. The matrix terms are real single precision. Twenty-four words are required for each column of KGG corresponding to point 5 (144 words for all 6 columns). The CGV for point 5 will contain 6 words as follows:

relative		corresponding point
1	1	3
2	0	4
3	7	5
4	13	6
5	0	7
6	19	8

When the allocation of working storage for a pass is complete, the SCRATCH2 data block is read one element at a time. If the element references a grid point which is within the current allocation, the column partitions corresponding to the point are read from KELM and added to the terms in working storage. The latter step is performed by subroutine EMA1S (single precision) or EMA1D (double precision).

When the SCRATCH2 data block has been read, the columns of KGG currently in working storage are complete. They are packed to the KGG data block by calling subroutine BLDPK.

If the GPST data block is to be generated (GPST is not purged), the matrix terms associated with the pivot (2 diagonal 3 x 3s) are written on SCRATCH1.

MODULE FUNCTIONAL DESCRIPTIONS

When all columns of KGG are complete, EMA1 is complete unless the GPST is to be generated. In this case, the SCRATCH1 data block is read and subroutine DETCKX is called to generate the singularity information.

4.161.8 Subroutines

4.161.8.1 Subroutine Name: DETCKX (see Section 4.123.8)

4.161.8.2 Subroutine Name: EMA1 $\begin{Bmatrix} S \\ D \end{Bmatrix}$

1. Entry Point: EMA1 $\begin{Bmatrix} S \\ D \end{Bmatrix}$

2. Purpose: To add a column vector partition to the corresponding assembled matrix. S is the single precision version. D is the double precision version.

3. Calling Sequence:

CALL EMA1 $\begin{Bmatrix} S \\ D \end{Bmatrix}$ (J,NSCA,SCALAS,PIVØT,DICT,CGV,KGG,CP,F)

where:

- J - index in SCALAS to current relative column number
- NSCA - number of terms per grid point in CP
- SCALAS - array of relative row/column numbers
- PIVØT - 1st 6 words of 8-word Pivot Control Block
- DICT - dictionary entry
- CGV - Connected Grid Vector
- KGG - matrix terms for pivot
- CP - column partition
- F - Factor (real single precision) to be applied to each term

4.161.9 Design Requirements

EMA1 is open ended. At least one complete pivot point must be allocated. Fatal message 3008 is generated if this is not the case.

4.161.10 Diagnostic messages

SYSTEM FATAL MESSAGE 3012, EMA1 LØGIC ERRØR nnn.

If this diagnostic occurs, nnn points to a FØRTRAN statement number at or near the logic test which has failed.

5. NASTRAN - OPERATING SYSTEM INTERFACES

5.1 INTRODUCTION

NASTRAN operates on: 1) the IBM Systems 360/370 under Operating System/Virtual Storage 1 (OS/VS1), Operating System/Virtual Storage 2 (OS/VS2), Operating System/Multiprogramming with a Variable Number of Tasks (OS/MVT), Operating System/Fixed Number of Tasks (OS/MFT), and Operating System/Multiprogramming with a Variable Number of Tasks (OS/MVT); 2) the UNIVAC 1108 and 1110 Series under the Exec 8 operating system; and 3) the CDC 6000/CYBER under the Netword Operating System (NOS) and the Netword Operating System Batch Environment (NOS-BE). This section discusses the interfaces between NASTRAN and these operating systems with respect to: 1) input/output; 2) link switching; 3) overlay considerations and implementation of the open core concept; 4) the setup of the operating system control cards for the NASTRAN data decks; 5) generation of an executable NASTRAN system; and 6) machine dependent routines.

The vocabulary used in each subsection is the one used by systems programmers familiar with the particular operating system being discussed. It is to these system programmers that each subsection is addressed.

THE MATERIAL FOR SECTION 5.2
HAS BEEN DELETED

5.3 NASTRAN ON THE IBM SYSTEM 360-370*

5.3.1 Introduction

The NASTRAN executable consists of sixteen (16) separate load modules called links. Each link comprises a member of the executable Partitioned Data Set (PDS). There are fifteen (15) functional links which have member names LINKNS01 - LINKNS15, and in addition there is an executive link with member name NASTRAN. The fifteen functional links are overlayed into multiple regions to conserve core storage requirements.

The NASTRAN executable operates as a single job step on the IBM 360-370 class of computer by specifying the executive link (NASTRAN) as the PGM to be executed in the Job Control Language (JCL). The executive link is always core resident. The functional links are loaded, executed and deleted one at a time by the executive link as they are needed. An additional core savings is gained by placing the FØRTRAN support routines, FØRTRAN I/Ø package and the NASTRAN I/Ø package in the executive link. The linkage to these routines, required by the functional links, is provided by the link control routine (LINKNSXX) at execution time.

5.3.2 Input/Output

Within the NASTRAN program all data transfer operations between primary and secondary storage with the exception of card, print, plot and special input/output are performed through the General Input/Output Routine (GINØ).

All non-GINØ I/Ø is performed through normal FØRTRAN I/Ø statements. Printed output is generated by formatted output statements to FØRTRAN logical units 4 and 6 (DDNAME=FT04F001 and DDNAME=FT06F001). This output may be routed as desired although it normally appears as a system output (SYSØUT) class. The usual output for the NASTRAN execution appears on FT06F001 while the Run Log appears on FT04F001. Data card input is read by formatted input statements, a card (record) at a time, from FØRTRAN logical unit 5 (DDNAME=FT05F001). Again, the input source may be designated as desired although it normally appears as the system input (SYSIN) unit. Punched cards are written on FT07F001 which normally appears as a system output (SYSØUT) class.

*The terms SYSTEM 360-370, IBM 360, SYSTEM 360, etc. will be used synonymously in this discussion as NASTRAN execution is identical on both the IBM 360 and the IBM 370 computers.

NASTRAN - OPERATING SYSTEM INTERFACES

The transliteration routine writes its output on FT01F001 in a form as indicated on the PROC given in Section 5.3.5. Other FØRTRAN data sets may be needed by users who use utility modules INPUTT2 or ØUTPUT2 (see Section 5 of the User's Manual).

The SGINØ (Special GINØ) plotter output routines within the S/360 NASTRAN system function independently of other I/Ø. Unique DD cards within the Execution Deck Setup (see Section 5.3.5) describe the required output tape file formats. Two separate plotter files may be generated, one on FØRTRAN unit 13 and the other on unit 14.

GINØ, which is written in assembly language, uses the Basic Sequential Access Method (BSAM) to read and write blocks of a fixed size that may be adjusted to fit hardware and problem requirements by using the NASTRAN control card. GINØ input/output operations result in calls to entry point IØ360 in NASTIØ to do the physical reads or writes.

In addition to the use of the Standard BSAM READ and WRITE macros, NASTRAN uses the NØTE and PØINT macros. The use of the NØTE and PØINT macros, along with fixed block sizes permits the use of the disk secondary storage in a direct access form, and through use of properly kept locators permits NASTRAN to accomplish backspacing across disk boundaries. It should also be noted that use of BSAM I/Ø processing permits I/Ø operations to be accomplished directly in the GINØ buffer areas. As a result, no data transfers take place (as opposed to FØRTRAN I/Ø which does transfer the data) and a considerable savings of both core storage and CPU time is effected.

The way in which temporary storage (scratch files) are allocated in NASTRAN is a by-product of the I/Ø method used. NASTRAN uses three classes of temporary files: primary files, secondary files, and tertiary files.

The following example depicts the way in which this space is managed. Assume NASTRAN data blocks A, B, C, D and E are of the following sizes:

A	10 blocks (records)
B	100 blocks
C	210 blocks
D	625 blocks
E	72 blocks

Further assume that each primary unit has preallocated space of 100 blocks and that each secondary unit has preallocated space of 400 blocks (e.g., 4 blocks/track and 25 tracks allocated to primaries and 100 tracks allocated to secondaries).

NASTRAN ON THE IBM SYSTEM 360-370

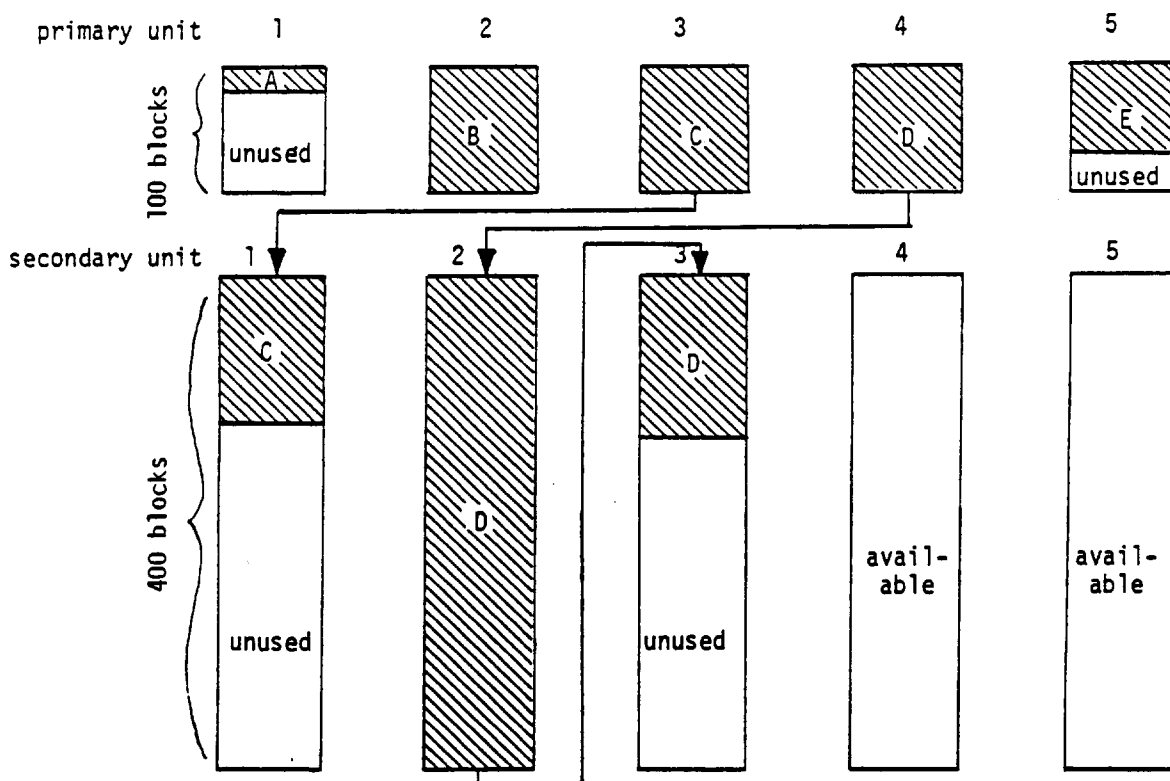


Figure 1. Space Allocation Diagram on the IBM 360

In the previous example, data blocks A, B, and E are contained in their initial primary allocations. Data block C is continued from primary unit 3 to secondary unit 1. Data block D is continued from primary unit 4 to secondary unit 2 to secondary unit 3. Note that since data block B exactly fills primary unit 2, any additional write operation will cause an extent. In this case, secondary unit 4 will be attached to primary unit 2. "Backspacing across Disks", for example from block number 101 in data block C to block number 100, is accomplished by recognizing that block number 100 is in primary unit 3 and issuing a POINT to that block.

In this example no tertiary files are used. Tertiary files are for the most part reserved for extremely large data blocks. When two secondary units have already been assigned to a data block and additional space is required, assignment of a tertiary unit is attempted. If no tertiary unit is available, assignment of another secondary is attempted. The only restriction to the way in which secondary and tertiary units may be chained to the same data block is a tertiary unit

which is on tape. In this case, this unit becomes the last unit in the chain of units assigned. Otherwise, it is possible for secondary units to follow tertiary units in the chain. Whenever the data block is released, all space previously assigned to it is released and made available for assignment to other data blocks. Through efficient use of this primary, secondary, and tertiary file concept of allowing NASTRAN to allocate and attach files as needed, the amount of scratch storage space can be greatly reduced, and the system ABEND B37 can be controlled.

5.3.3 Link Switching

NASTRAN link switching on the IBM 360-370 class of computers is performed dynamically by the executive link driver routine (NASTRAN) within the executive link. The Preface (LINKNS01) is always the first functional link to be loaded and executed.

Within each functional link there are two (2) routines (LINKNSXX, XSEMi, $i=01, 02, \dots, 15$) which assist the executive link in link switching. When the XSEMi subroutine determines that the next module to be executed resides in a link other than the one currently core resident, the required link is requested through a call to SEARCH (an entry point in LINKNSXX). The SEARCH call carries the name of the link requested as an argument. SEARCH places the name of the requested link into the NASTRAN Linkage Control Table (NLCT) and calls EXIT (another entry point in LINKNSXX) to return to the executive link. The executive link determines whether the requested link is a valid link name by checking the NASTRAN Link Name Table (NLNT), and if so, it then deletes the previous link and loads and executes the requested link. Since the NASTRAN links are individual members of the executable PDS, the number of links is essentially open ended.

5.3.4 Overlay Considerations and Implementation of Open Core

Each NASTRAN functional link is created by the Linkage Editor (see Section 5.3.6.3). Each link contains a zero level or root segment and the series of overlay segments necessary to NASTRAN operating logic. The specially named common blocks which define the beginning of various open core areas are placed at their required location by linkage editor INSERT directives. An overlay tree functions by automatically loading all segments in the branch between the calling segment and the called segment. Local FORTRAN variables and common blocks residing within segments are not cleared at load time.

Several NASTRAN links contain a special type origin for the overlay tree. This origin is created by declaring a particular segment boundary to be at a region boundary. This region boundary automatically begins at the end of the longest branch in the previous region. Since many links contain a series of small functional module drivers of different lengths followed by a structure of matrix routines used by these drivers, a region boundary is usually utilized following the drivers. The following sketch illustrates a link structure with regions.

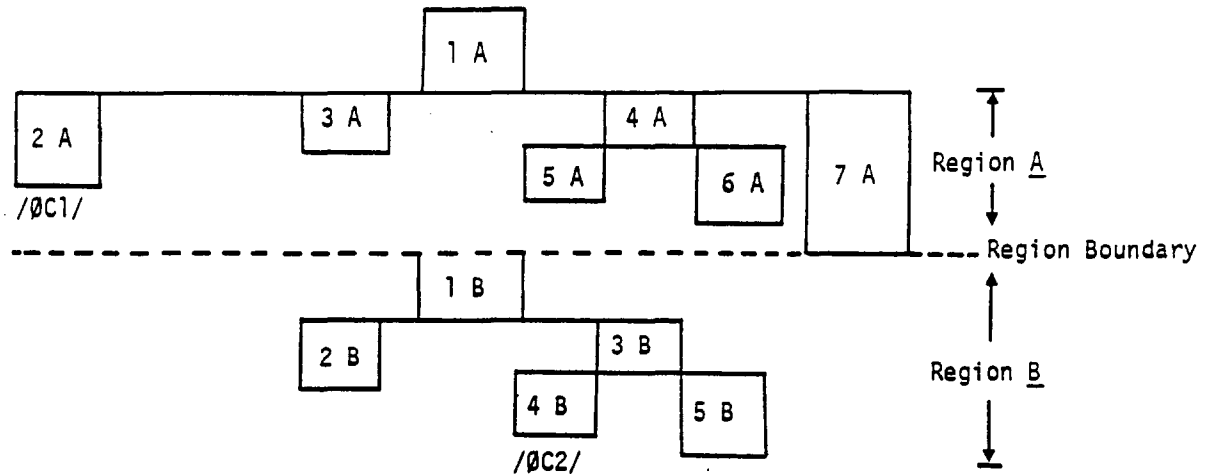


Figure 2. A tree diagram for a NASTRAN link on the IBM 360

Note that since segment 1B was requested to be begun at a region boundary, the Linkage Editor placed its origin following the longest segment (7A) of the previous region.

Special caution must be exercised when operating within open core under a region structure. Operations within each region are independent of other regions. Thus, if a branch of segments within region B, say 1B and 3B, is in core because of previous calls, a call from region A to region B, say 5B, will not reload segments 1B and 3B. Therefore, if an open core area starting at ØC1/ were utilized, some of the programs within region B could be destroyed without the loader's knowledge. Because of this possibility, all NASTRAN open core origins lie within a lower level region. Therefore open core starting areas such as /ØC1/ are not used in S/360 NASTRAN overlay structures.

Because NASTRAN must operate on most models of System 360 with a variety of memory sizes and operating systems*, a flexible method of utilizing all memory available to the job is incorporated.

*Operating systems and systems will be used synonymously in this section.

NASTRAN - OPERATING SYSTEM INTERFACES

In both primary and multi-programmed environments, the loader requests storage from the core memory available to the job through the GETMAIN macro. Storage-management routines make sufficient core available, and the link is placed in memory. Core memory outside of that requested for the link is storage protected by the system. An attempt to store into these protected areas causes an interrupt and job termination. The NASTRAN open core concept requires use of those areas that are available but protected (i.e., the area between the link and the region boundary). To remove this protected status, the NASTRAN initialization program within each functional link issued a conditional GETMAIN for all remaining memory within the job region. The return from this GETMAIN specifies the origin and the size of the block of core memory acquired. A small portion of this block is returned (via the FREEMAIN macro) to the system for use as FORTRAN I/O buffers and for other system functions. The remainder of this memory is made available to NASTRAN by adjusting the upper core address used by the NASTRAN KORSZ function. All NASTRAN modules may thus utilize the maximum core memory provided to the job.

5.3.5 PARM Options

Options may be passed to NASTRAN via the PARM parameter on the EXEC statement. If the PARM parameters are coded incorrectly, NASTRAN will terminate after the Preface. The following information may be specified.

1. Whether open core usage statistics are printed in the NASTRAN log file at the termination of each functional link.
2. The amount of core freed back by NASTRAN for FORTRAN I/O buffers and for use by the Operating System. (Note: This value was previously supplied to NASTRAN via the keyword K0N360 on the NASTRAN card. K0N360 is no longer supported.)

The PARM parameter is coded as follows:

$$\left\{ \begin{array}{c} \text{PARM} \\ \text{PARM.NS} \end{array} \right\} = \text{'CORE'} = \left(\left\{ \begin{array}{c} \text{N0STAT} \\ \text{STAT} \end{array} \right\} , \left\{ \begin{array}{c} \text{nnnnn} \\ \text{nnk} \end{array} \right\} \right) ,$$

The CORE keyword has two subfields. If only the first subfield is coded, the parentheses are not needed. If the first subfield is omitted, a comma must be coded to indicate the omission. The first subfield indicates whether open core usage statistics are printed in the NASTRAN log file. It should be coded STAT or N0STAT. N0STAT is the default value. (Note: For virtual storage users

NØSTAT should reduce the paging rate.)

The second subfield is the integer number of bytes of core that NASTRAN will free back for FØRTRAN I/Ø buffers and for use by the Operating System. It may be coded with or without a trailing K which indicates multiplication by 1000. The default value is 48K.

5.3.6 Execution Deck Setup

Running or executing the NASTRAN system on a System 360 computer once the generation procedure (see Section 5.3.6) is complete requires some basic knowledge of the type of structural problem being solved and the type of output requested. In addition, the hardware configuration and capacities should be known in order to most adequately match the problem being solved to the computer.

The following procedure should provide the basic Job Control Language (JCL) necessary for all NASTRAN runs.

•

```

1. //NASTRAN PROC NAME='XLJLR.NSTNLMOD.L1750.LOAD',
// PUNITS=CYL,P1=1,P2=1,
// SUNITS=CYL,S1=3,S2=1,
// TUNITS=CYL,T1=3,T2=1,
// DENPLT=1
/**-----
/**---
/**--- *****
/**--- *   PROCEDURE   N A S T R A N   *
/**--- *****
/**---
/**-----

2. //NS EXEC PGM=NASTRAN,REGION=410K,PARM='CORE=(STAT,64K)'
3. //FT01F001 DD UNIT=SYSDA,SPACE=(CYL,(5,1)),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=7280,BUFNO=1)
4. //FT04F001 DD SYSOUT=A,DCB=(RECFM=VBA,LRECL=137,BLKSIZE=3429)
5. //FT05F001 DD DDNAME=SYSIN
6. //FT06F001 DD SYSOUT=A,DCB=(RECFM=VBA,LRECL=137,BLKSIZE=3429)
7. //FT07F001 DD SYSOUT=B
8. //FT14F001 DD DDNAME=PLT2 * FORTRAN PLOT REF
9. //INP1 DD DSN=NULLFILE EXTRA
10. //INP2 DD DSN=NULLFILE INPUT
11. //INP3 DD DSN=NULLFILE AND
12. //INP4 DD DSN=NULLFILE OUTPUT
13. //INP5 DD DSN=NULLFILE UNITS
14. //INP6 DD DSN=NULLFILE FOR
15. //INP7 DD DSN=NULLFILE MATHIX
16. //INP8 DD DSN=NULLFILE STORAGE
17. //INP9 DD DSN=NULLFILE AND
18. //INPT DD DSN=NULLFILE RETRIEVAL
/**
/** FOLLOWING FIVE CARDS DEFINE THE EXECUTIVE FILES.
/** EXCEPT FOR THE NPTP AND POOL, THESE FILES ARE DEFINED TO BE NULL
/** AND NO EXTERNAL USE CAN BE MADE OF THESE DATA SETS UNLESS PROC
/** OVERRIDE CARDS ARE SUPPLIED---SPECIFY UNIT, DSN, VOL-SER-NO.
/** FURTHERMORE ANY OVERRIDES MUST APPEAR IN THE SAME ORDER AS
/** THOSE BEING OVERRIDDEN. I.E., ALPHABETICAL.
/**

19. //NPTP DD UNIT=SYSDA,SPACE=(&PUNITS,&(P1,&P2)),DISP=(,PASS)
20. //OPTP DD DSN=NULLFILE,LABEL=(,NL),DISP=OLD
21. //PLT2 DD DSN=NULLFILE
22. //POOL DD UNIT=SYSDA,SPACE=(&PUNITS,&(P1,&P2))
23. //UMF DD DSN=XLJLR.UMF.BULK.L1750.DATA,DISP=SHR
/**
/** **** EXAMPLE TAPE OVERRIDE CARDS FOR EXECUTIVE FILE DD'S ****
/** **** QUESTIONS(?) MUST BE REPLACED WITH APPROPRIATE NAME ****
/**
/** //NS.NPTP DD UNIT=2400,VOL=SER=?,DSN=?,DISP=(NEW,KEEP),
/** // LABEL=(1,BLP)
/** //NS.OPTP DD UNIT=2400,VOL=SER=?,DSN=?,DISP=OLD,
/** // LABEL=(1,BLP)

```

Table 1. Basic Job Control Language (JCL) for NASTRAN Runs (Continued)

```

/**
/**
/** ***** PRIMARY SCRATCH DATA SETS *****
/**
24. //PRI01 DD UNIT=SYSDA,SPACE=(&PUNITS,(&P1,&P2))
25. //PRI02 DD UNIT=SYSDA,SPACE=(&PUNITS,(&P1,&P2))
26. //PRI03 DD UNIT=SYSDA,SPACE=(&PUNITS,(&P1,&P2))
27. //PRI04 DD UNIT=SYSDA,SPACE=(&PUNITS,(&P1,&P2))
28. //PRI05 DD UNIT=SYSDA,SPACE=(&PUNITS,(&P1,&P2))
29. //PRI06 DD UNIT=SYSDA,SPACE=(&PUNITS,(&P1,&P2))
30. //PRI07 DD UNIT=SYSDA,SPACE=(&PUNITS,(&P1,&P2))
31. //PRI08 DD UNIT=SYSDA,SPACE=(&PUNITS,(&P1,&P2))
32. //PRI09 DD UNIT=SYSDA,SPACE=(&PUNITS,(&P1,&P2))
33. //PRI10 DD UNIT=SYSDA,SPACE=(&PUNITS,(&P1,&P2))
34. //PRI11 DD UNIT=SYSDA,SPACE=(&PUNITS,(&P1,&P2))
35. //PRI12 DD UNIT=SYSDA,SPACE=(&PUNITS,(&P1,&P2))
36. //PRI13 DD UNIT=SYSDA,SPACE=(&PUNITS,(&P1,&P2))
37. //PRI14 DD UNIT=SYSDA,SPACE=(&PUNITS,(&P1,&P2))
38. //PRI15 DD UNIT=SYSDA,SPACE=(&PUNITS,(&P1,&P2))
39. //PRI16 DD UNIT=SYSDA,SPACE=(&PUNITS,(&P1,&P2))
40. //PRI17 DD UNIT=SYSDA,SPACE=(&PUNITS,(&P1,&P2))
41. //PRI18 DD UNIT=SYSDA,SPACE=(&PUNITS,(&P1,&P2))
42. //PRI19 DD UNIT=SYSDA,SPACE=(&PUNITS,(&P1,&P2))
43. //PRI20 DD UNIT=SYSDA,SPACE=(&PUNITS,(&P1,&P2))
44. //PRI21 DD UNIT=SYSDA,SPACE=(&PUNITS,(&P1,&P2))
45. //PRI22 DD UNIT=SYSDA,SPACE=(&PUNITS,(&P1,&P2))
46. //PRI23 DD UNIT=SYSDA,SPACE=(&PUNITS,(&P1,&P2))
47. //PRI24 DD UNIT=SYSDA,SPACE=(&PUNITS,(&P1,&P2))
48. //PRI25 DD UNIT=SYSDA,SPACE=(&PUNITS,(&P1,&P2))
49. //PRI26 DD UNIT=SYSDA,SPACE=(&PUNITS,(&P1,&P2))
50. //PRI27 DD UNIT=SYSDA,SPACE=(&PUNITS,(&P1,&P2))
51. //PRI28 DD UNIT=SYSDA,SPACE=(&PUNITS,(&P1,&P2))
52. //PRI29 DD UNIT=SYSDA,SPACE=(&PUNITS,(&P1,&P2))
53. //PRI30 DD UNIT=SYSDA,SPACE=(&PUNITS,(&P1,&P2))
54. //PRI31 DD UNIT=SYSDA,SPACE=(&PUNITS,(&P1,&P2))
55. //PRI32 DD UNIT=SYSDA,SPACE=(&PUNITS,(&P1,&P2))
/**
/** ***** SECONDARY SCRATCH DATA SETS *****
/**
56. //SEC01 DD UNIT=SYSDA,SPACE=(&SUNITS,(&S1,&S2))
57. //SEC02 DD UNIT=SYSDA,SPACE=(&SUNITS,(&S1,&S2))
58. //SEC03 DD UNIT=SYSDA,SPACE=(&SUNITS,(&S1,&S2))
59. //SEC04 DD UNIT=SYSDA,SPACE=(&SUNITS,(&S1,&S2))
60. //SEC05 DD UNIT=SYSDA,SPACE=(&SUNITS,(&S1,&S2))
61. //SEC06 DD UNIT=SYSDA,SPACE=(&SUNITS,(&S1,&S2))
62. //SEC07 DD UNIT=SYSDA,SPACE=(&SUNITS,(&S1,&S2))
63. //SEC08 DD UNIT=SYSDA,SPACE=(&SUNITS,(&S1,&S2))
64. //SEC09 DD UNIT=SYSDA,SPACE=(&SUNITS,(&S1,&S2))
65. //SEC10 DD UNIT=SYSDA,SPACE=(&SUNITS,(&S1,&S2))
66. //SEC11 DD UNIT=SYSDA,SPACE=(&SUNITS,(&S1,&S2))
67. //SEC12 DD UNIT=SYSDA,SPACE=(&SUNITS,(&S1,&S2))
68. //SEC13 DD UNIT=SYSDA,SPACE=(&SUNITS,(&S1,&S2))
69. //SEC14 DD UNIT=SYSDA,SPACE=(&SUNITS,(&S1,&S2))
70. //SEC15 DD UNIT=SYSDA,SPACE=(&SUNITS,(&S1,&S2))
71. //SNAPSHOT DD SYSOUT=A,DCB=(RECFM=VBA,LRECL=125,BLKSIZE=882)
72. //STEPLIB DD DSN=NAME,DISP=SHR
73. //SYSUDUMP DD SYSOUT=A

```

NASTRAN - OPERATING SYSTEM INTERFACES

Table 1. Basic Job Control Language (JCL) for NASTRAN Runs (Continued)

```

/**
/** ***** TERTIARY SCRATCH DATA SETS *****
/**
74. //TER01 DD UNIT=SYSDA,SPACE=(&TUNITS,(&T1,&T2))
75. //TER02 DD UNIT=SYSDA,SPACE=(&TUNITS,(&T1,&T2))
76. //TER03 DD UNIT=SYSDA,SPACE=(&TUNITS,(&T1,&T2))
77. //TER04 DD UNIT=SYSDA,SPACE=(&TUNITS,(&T1,&T2))
78. //XPTD DD UNIT=SYSDA,SPACE=(&PUNITS,(&P1,&P2))
/**-----
/**----- END OF PROCEDURE -----
/**-----
79. // PEND

```

Each card or group of cards within the NASTRAN procedure is discussed by item below. The item numbers match those along the left margin of the preceding deck listing.

Item

Description

1. The PRØC card defines the name of the instream procedure and sets default values for the symbolic parameters.

<u>SYMBOLIC PARAMETER</u>	<u>DEFAULT VALUE</u>	<u>DESCRIPTION</u>
DENPLT	1	Defines the density of the plot tape.
NAME	XLJLR.NSTNLMØD. L1750.LØAD	Defines the data set name of the executable.
PUNITS	CYL	Defines the space allocation units for the primary files.
P1	1	Defines the initial space allocation for the primary files.
P2	1	Defines the increment space allocation for the primary files.
SUNITS	CYL	Defines the space allocation units for the secondary files.
S1	3	Defines the initial space allocation for the secondary files.
S2	1	Defines the increment space allocation for the secondary files.
TUNITS	CYL	Defines the space allocation units for the tertiary files.
T1	3	Defines the initial space allocation for tertiary files.
T2	1	Defines the increment space allocation for tertiary files.

2. The EXEC card defines the name of the program to be executed, NASTRAN, the region size in which NASTRAN is to be executed, and the default PARM.

3. The FT01F001 DD card defines the data set to be used as intermediate storage for NASTRAN input. BCD or EBCDIC card images are read and converted to BCD card images and written on unit 1, then NASTRAN reads its input from unit 1.

4. The FT04F001 DD card defines the data set that will contain the run log. The NASTRAN run log contains internal timing for NASTRAN and a trace of the modules that are executed in an execution. It is usually assigned to a printer (SYSØUT=A); however, it may be assigned to any device or set to dummy and thereby deleted as desired.

NASTRAN - OPERATING SYSTEM INTERFACES

5. The FT05F001 DD card is deferred by the use of the DDNAME=SYSIN parameter and will be discussed in the examples that follow.
6. The FT06F001 DD card defines the printed output from the NASTRAN run. This printed output is written in BCD. As with FT04F001, it can be modified at execution time by the user as desired.
7. The FT07F001 DD card defines the punched output from NASTRAN executions. It can be modified as desired.
8. The FT14F001 DD card is deferred by the use of DDNAME=PLT2 (see Item 23).
- 9 thru 18. These DD cards define data sets to be used as user tapes through the use of INPUTT2 and OUTPUT2 routines. All have the parameter DSN=NULLFILE which restricts the operating system from allocating them unless they are supplied as override JCL.
19. The NPTP DD card is used to describe the new problem tape (checkpoint tape) to be used by NASTRAN. It is set up as a temporary file and should be overridden if it is to be saved for later use.
20. The ØPTP card defines the old problem tape (previous checkpoint tape) for the NASTRAN run. It must exist prior to the NASTRAN run in which it is used, and in this procedure DSN=NULLFILE must be overridden before the ØPTP could be used to retrieve data.
21. The PLT2 DD card defines the output data set containing the SC4020, Calcomp or DD80 plotting data.
22. The PØØL DD cards define the data set to be used as the Data Pool file. It is always present and refers to temporary scratch disk.
23. The UMF DD card defines the data set that contains the user master file data to be input to NASTRAN. It must be specified if it is needed.
- 24 thru 55. These DD cards define the primary units to be used by NASTRAN as temporary working files. (Note, each DD card must reference only one unit).
- 56 thru 70. These DD cards define the secondary units to be used by NASTRAN as temporary working files. (Note, each DD card must reference only one unit).

71. The SNAPSHOT DD card defines the data set that contains the diagnostic dump if NASTRAN takes a user abort. It is assigned to the printer.
72. The STEPLIB card defines the data set name and the location of the executable.
73. The SYSUDUMP DD card defines the data set that contains the diagnostic dump if NASTRAN takes a system abort. It is assigned to a printer.
- 74 thru 77. These DD cards define the tertiary units to be used by NASTRAN as temporary working files. (Note, each DD card must reference only one unit).
78. The XPTD DD card defines the temporary data set that is to contain the checkpoint dictionary during CHECKPOINT runs in a GINØ compatible format (NOT BCD card images).
79. The PEND card signifies the end of an instream procedure. It should be removed before the JCL is placed on the PRØCLIB.

The following restriction for Level 17 should be noted: Plot tapes must be 7-track tapes when plotting for a Stromberg-Carlson plotter; any remaining tapes (problem tapes, UMF, PLT2 tape for the Calcomp plotter, or the General Purpose Plotter, etc.) may be 7 or 9 track as the user desires.

The NASTRAN execution deck setup is presented as an instream procedure such that it may be verified before addition to the installation procedure library (PRØCLIB) to permit easy recall and reuse.

The procedure provided is intended for use on an IBM 360 computer operating under straight ØS. It should be reviewed for each installation prior to actual use. Among the items that may necessitate modification of the procedure are the following:

1. Any modification to the standard IBM ØS operating system could make modification of the PRØC necessary. For example, all SYSØUT=A and SYSØUT=B data sets are provided with DCB information and space allocation. If a particular installation is using HASP (Houston Automatic Spooling System), it becomes necessary to remove the DCB parameters and the space allocation is no longer needed. Similar modifications will be necessary when running under ASP (Auxiliary Support Processor) or related systems.

NASTRAN - OPERATING SYSTEM INTERFACES

2. The procedure as provided has temporary scratch file space sufficient for small to medium size problems. Medium to large problems will require the initial space allocation for primary and secondary files to be doubled (P1=2, S1=6), while very large problems often tax the resources of the computer and must be dealt with on an individual basis.
3. If the number of DD statements in the procedure is too great, it may be reduced (at some cost in performance) as follows with the most expendable data sets listed first:
 - a. INPT,INP1,INP2,---,INP9,NUMF,ØPTP,UMF,PLT2(and FT14F001). The data sets are not used by NASTRAN unless requested by the user through his data.
 - b. TER01,TER02,---. Tertiary files may be eliminated if careful thought is given to the selection of the space allocation parameters P1 and S1.
 - c. The secondary files (SEC01,SEC02,---,SEC15) may be cut back to a single one (SEC01) as long as enough space is pre-allocated for the primary files (PRI01, PRI02,---) to run the problem. It is recommended that at least four or five secondary files be retained, however.

5.3.6.1 Execution Using the Instream Procedure

A sample NASTRAN execution with the instream procedure could be:

```
//NASTRAN JOB -----  
  { Instream  
    Procedure  
    Deck  
  }  
// EXEC NASTRAN  
//NS.SYSIN DD *  
  { NASTRAN EXECUTIVE CONTROL, CASE CONTROL,  
    and BULK DATA CARDS  
  }  
/*
```

5.3.6.2 Execution Using the Demo Driver Decks and UMF Tape

Extra JCL is necessary to execute a NASTRAN demonstration problem. (Demo 1-1-2 is used as an example.)

```
1 // EXEC PRØC=NASTRAN  
2 //NS.UMF DD UNIT=2400,DISP=ØLD,VØL=SER=XLJLRF  
  // DSN=UMF,DCB=DEN=3,LABEL=(1,SL)
```

NASTRAN ON THE IBM SYSTEM 360-370

```

3  //NS.SYSIN DD UNIT=2314,DISP=SHR,VOL=SER=DISKMF,
   //          DSN=XLJLR.UMF.DRIVER.L1750.DATA(D10120),DCB=BUFNO=1,
   //          LABEL=(,,IN)
   //

```

<u>Item</u>	<u>Description</u>
1	Executes the NASTRAN procedure using all of the defaults for the symbolic parameters.
2	Overrides the UMF DD card to reference the UMF tape.
3	Overrides the SYSIN DD card to reference the UMF Demo Driver decks.
NOTE: The Executive Control Deck includes the appropriate UMF card, so no bulk data cards are required.	

5.3.6.3 Execution Using the UMF Disk File

Extra JCL and a NASTRAN card before the Executive Control deck ID card are necessary to use the UMF Disk file.

```

1  //          EXEC PRØC=NASTRAN
2  //NS.UMF DD UNIT=2314,DISP=SHR,VOL=SER=DISKMF,
   //          DSN=XLJLR.UMF.BULK.L1750.DATA
3  //NS.SYSIN DD *
4  NASTRAN FILES=UMF $ READ FROM DISK

{NASTRAN Executive Control and Case Control Decks.}

BEGIN BULK
ENDDATA
//

```

<u>Item</u>	<u>Description</u>
1	Executes the NASTRAN procedure using all of the defaults for the symbolic parameters.
2	Overrides the UMF DD card to reference the UMF disk file.
3	Overrides the SYSIN DD card to reference the card reader.
4	The first input card to NASTRAN must be this card. It specifies that the UMF is on disk. The Executive Control deck includes the appropriate UMF card, so no bulk data cards are required.

5.3.6.4 Execution Using Checkpoint and Restart

Extra JCL is necessary for checkpoint.

```

1  //          EXEC PRØC=NASTRAN
2  //NS.NPTP DD UNIT=2400,DISP=(NEW,KEEP),
   //          VOL=SER=xxxxx,DSN=RESTART,LABEL=(,NL)

```


NASTRAN - OPERATING SYSTEM INTERFACES

3 //NS.SYSIN DD *

{NASTRAN Executive Control, Case Control and Bulk Data decks.}

//

<u>Item</u>	<u>Description</u>
1	Executes the NASTRAN procedure using all of the defaults for the symbolic parameters.
2	Overrides the NPTP DD card to reference the <u>New Problem TaPe</u> (Checkpoint tape).
3	Overrides the SYSIN DD card to reference the card reader.

Extra JCL is necessary for restart.

1 // EXEC PRØC=NASTRAN

2 //NS.ØPTP DD UNIT=2400,DISP=ØLD,VØL=SER=xxxxx,
// DSN=RESTART,LABEL=(,NL)

3 //NS.SYSIN DD *

{NASTRAN Executive Control, Case Control and Bulk Data decks.}

//

<u>Item</u>	<u>Description</u>
1	Executes the NASTRAN Procedure using all of the defaults for the symbolic parameters.
2	Overrides the ØPTP DD card to reference the <u>Old Problem TaPe</u> (Restart tape).
3	Overrides the SYSIN DD card to reference the card reader.

5.3.6.5 Execution Using SC4020 for Plotting

Extra JCL is necessary for a plot run. The SC4020 plotter is assumed in the example.

1 // EXEC PRØC=NASTRAN

2 //NS.PLT2 DD UNIT=2400-7,DISP=(NEW,KEEP),
// VØL=SER=xxxxx,DSN=PLT2,
// DCB=(RECFM=U,BLKSIZE=2400,BUFNØ=1),
// LABEL=(,NL)

3 //NS.SYSIN DD *

{NASTRAN Executive Control, Case Control and Bulk Data decks.}

//

<u>Item</u>	<u>Description</u>
1	Executes the NASTRAN procedure using all of the defaults for the symbolic parameters.
2	Overrides the PLT2 DD card and references the plot tape. For the NASTRAN General Purpose Plotter, the following DCB parameters should be used: DCB=(RECFM=F,BLKSIZE=3000,BUFNØ =1)
3	Overrides the SYSIN DD card and references the card reader.

5.3.6.6 Execution Using Substructure Operating File (SØF)

Extra JCL is necessary for a Multi-stage Substructure Problem. The SØF is assumed to be on FØRTRAN unit 18 in the example (see the NASTRAN User's Manual, Section 2.7).

```

1  //      EXEC PRØC=NASTRAN
2  //NS.FT18F001 DD DSN=SØF.DATA,DISP=ØLD
3  //NS.SYSIN   DD *

{NASTRAN Executive Control, Case Control and Bulk Data decks.}

//

```

<u>Item</u>	<u>Description</u>
1	Executes the NASTRAN procedure using all of the defaults for the symbolic parameters.
2	Overrides the FT18F001 DD card and references the SØF. If this file is to be allocated during this run, the following parameters should be included on the DD card: DISP=NEW,UNIT=xxxx,SPACE=(CYL,yyyy),VØL=SER=zzzz
3	Overrides the SYSIN DD card and references the card reader.

5.3.6.7 Printing a Demonstration Problem Listing

To print the FT06 print file generated when Demonstration Problem 1-2-1 was executed on Level 17.0.1 NASTRAN (XLJLR.NSTNLMØD.L1750.LØAD).

```

1  //      EXEC PGM=IEBGENER
2  //SYSPRINT DD SYSØUT=A
3  //SYSUT1   DD DSN=XLJLR.L1750.PRINT.DATA(D1Ø21Ø),
//           DISP=SHR
4  //SYSUT2   DD SYSØUT=A
5  //SYSIN    DD DUMMY
//

```

<u>Item</u>	<u>Description</u>
1	Executes the IBM utility IEBGENER.
2	Defines the print output file for IEBGENER.
3	Defines the data set name and member name of print file to be printed.
4	Defines the printer as the output file on which the file referenced by SYSUT1 (card 3) is to be copied.
5	Defines the input file to IEBGENER. This is required even though there are no input control cards.

NASTRAN - OPERATING SYSTEM INTERFACES

5.3.7 Physical Items and Generation of the NASTRAN System

5.3.7.1 Description of the NASTRAN Physical Items (Level 17.5.0)

All delivery tapes were created on the GSFC IBM 360/95 (ASP/MVT Release 21.8). All tapes are 9-track, labeled (LABEL=(file,SL)), and were written at 1600 bpi.

All partitioned data sets were unloaded from IBM 2314 disk packs by the IBM utility program IEHMØVE. While on tape, these data sets have the following DCB parameters:

DCB=(RECFM=FB,LRECL=80,BLKSIZE=800,DEN=3)

NAST01 - EXECUTABLE (VOLUME SERIAL NUMBER XLJLRC)

This tape contains the NASTRAN Level 17.5.0 executable and the NASTRAN overlay load map (print output from the Linkage Editor). NAST01 consists of one volume and two files.

File 1 - NASTRAN executable - Unloaded partitioned data set (16 members)

DSNAME=XLJLRC.NSTNLMØD,L1750.LØAD

Restored DCB parameters and SPACE requirements:

DCB=(DSØRG=PØ,RECFM=U,LRECL=0,BLKSIZE=7294)
SPACE=(TRK,(2500,,10))

File 2 - NASTRAN load map - Unloaded partitioned data set (16 members).

DSNAME=XLJLRC.LINKEDIT.L1750.DATA

Restored DCB parameters and SPACE requirements:

DCB=(DSØRG=PØ,RECFM=FB,LRECL=121,BLKSIZE=605)
SPACE=(TRK,(250,,2))

NAST02 - SOURCE (VOLUME SERIAL NUMBER XLJLRD)

This tape contains the Level 17.5.0 machine independent source code (A-M).

File 1 - NASTRAN machine independent SØURCE - Unloaded partitioned data set (852 members).

DSNAME=XLJLRC.MISØURCE.L1750.AM.DATA

Restored DCB parameters and SPACE requirements:

DCB=(DSØRG=PØ,RECFM=FB,LRECL=80,BLKSIZE=7280)
SPACE=(TRK,(3800,,100))

NAST03 - SOURCE (VOLUME SERIAL NUMBER XLJLRE)

This tape contains the Level 17.5.0 machine independent source code (N-Z).

File 1 - NASTRAN machine independent SØURCE - Unloaded partitioned data set (661 members).

DSNAME=XLJLRC.MISØURCE.L1750.NZ.DATA

Restored DCB parameters and SPACE requirements:

DCB=(DSØRG=PØ,RECFM=FB,LRECL=80,BLKSIZE=7280)
SPACE=(TRK,(3700,,100))

NASTRAN ON THE IBM SYSTEM 360-370

NAST04 - MACHINE DEPENDENT SOURCE AND OBJECT (VOLUME SERIAL NUMBER XLJLRF)

File 1 - NASTRAN machine dependent SOURCE (MDSOURCE) - Unloaded partitioned data set
(20 members)

DSNAME=XLJLR.MDSOURCE.L1750.DATA

Restored DCB parameters and SPACE requirements:

DCB=(DSORG=PO,RECFM=FM,LRECL=80,BLKSIZE=7280)
SPACE=(TRK,(60,,5))

THIS PAGE HAS BEEN LEFT BLANK INTENTIONALLY.

NASTRAN ON THE IBM SYSTEM 360-370

File 2 - NASTRAN Assembly SOURCE (ASSOURCE) - Unloaded partitioned data set (22 members).

DSNAME=XLJLR.ASSOURCE.L1750.DATA

Restored DCB parameters and SPACE requirements:

DCB=(DSORG=P0,RECFM=FB,LRECL=80,BLKSIZE=7280)
SPACE=(TRK,(300,,10))

File 3 - MACRØ Library - Unloaded partitioned data set (8 members).

DSNAME=XLJLR.NASTRAN.MACLIB.ASM

Restored DCB parameters and SPACE requirements:

DCB=(DSORG=P0,RECFM=FB,LRECL=80,BLKSIZE=3360)
SPACE=(TRK,(60,,2))

File 4 - NASTRAN overlay control cards - Unloaded partitioned data set (16 members).

DSNAME=XLJLR.ØVERLAY.L1750.DATA

Restored DCB parameters and SPACE requirements:

DCB=(DSORG=P0,RECFM=FB,LRECL=80,BLKSIZE=3200)
SPACE=(TRK,(80,,5))

File 5 - NASTRAN include control cards - Unloaded partitioned data set (16 members).

DSNAME=XLJLR.INCLUDE.L1750.DATA

Restored DCB parameters and SPACE requirements:

DCB=(DSORG=P0,RECFM=FB,LRECL=80,BLKSIZE=3200)
SPACE=(TRK,(80,,5))

File 6 - NASTRAN non-executable load modules - Unloaded partitioned data set (1608 members).

DSNAME=XLJLR.ØBJMØD.L1750.LØAD

Restored DCB parameters and SPACE requirements:

DCB=(DSORG=P0,RECFM=U,LRECL=0,BLKSIZE=7294)
SPACE=(TRK,(2000,,400))

NAST05 - DEMO ITEMS (VOLUME SERIAL NUMBER XLJLRG)

This tape contains the User Master File, the Demonstration Problem Driver Decks and the printout from the UMF verification run. NAST05 consists of one volume and three files.

NOTE: File 1 of NAST05 was created by Level 17.5.0 NASTRAN.

File 1 - User Master File (UMF) - Sequential file.

DSNAME=XLJLR.UMF.BULK.L1750.DATA

Restored DCB parameters and SPACE requirements:

DCB=(DSORG=PS,RECFM=F,LRECL=7288,BLKSIZE=7288)
SPACE=(TRK,240)

NASTRAN - OPERATING SYSTEM INTERFACES

File 2 - Demonstration Problem Driver Decks - Unloaded partitioned data set (82 members).

DSNAME=XLJLR.UMF.DRIVER.L1750.DATA

Restored DCB parameters and SPACE requirements:

DCB=(DSORG=P0,LRECL=80,BLKSIZE=7280,RECFM=FB)
SPACE=(TRK,(70,,10))

File 3 - NASTRAN UMF verification FT06 print file - Unloaded partitioned data set (1 member).

DSNAME=XLJLR.UMF.PRINT.L1750.DATA

Restored DCB parameters and SPACE requirements:

DCB=(DSORG=P0,LRECL=137,BLKSIZE=7265,RECFM=VBA)
SPACE=(TRK,(500,,1))

NAST06 - DEMO PRINT FILES (VOLUME SERIAL NUMBER XLJLRH)

This tape contains the FT06 print files of the 82 Demonstration Problems that were generated using Level 17.5.0 NASTRAN Executable (XLJLR.NSTNLM0D.L1750.L0AD), the Demonstration Problem Driver Decks (XLJLR.UMF.DRIVER.L1750.DATA) and the User Master File (File 1 of VOLUME SERIAL NUMBER XLJLRG). NAST06 consists of one volume and one file.

File 1 - Demonstration Problem Print File - Unloaded partitioned data set (82 members).

DSNAME=XLJLR.DEM0.PRINT.L1750.DATA

Restored DCB parameters and SPACE requirements:

DCB=(DSORG=P0,RECFM=VBA,LRECL=137,BLKSIZE=7265)
SPACE=(TRK,(2500,,10))

NAST07 - DISK DUMP OF RSRABB (VOLUME SERIAL NUMBER XLJLRI)

This tape is a disk dump of an IBM 2314 disk pack (using the IBM Utility Program IEHDASDR). This tape contains the data sets of: Files 1 and 2 of NAST01; Files 1, 2, 3, 4, and 5 of NAST04; File 3 of NAST05; and three extraneous maintenance contractor files. NAST07 consists of one volume and 12 files.

File 1 - DSNAME=DASDRM0

The following data sets are restored from this tape:

DSNAME=XLJLR.NSTNLM0D.L1750.L0AD	(File 1 of NAST01)
DSNAME=XLJLR.LINKEDIT.L1750.DATA	(File 2 of NAST01)
DSNAME=XLJLR.MDS0URCE.L1750.DATA	(File 1 of NAST04)
DSNAME=XLJLR.ASS0URCE.L1750.DATA	(File 2 of NAST04)
DSNAME=XLJLR.NASTRAN.MACLIB.ASM	(File 3 of NAST04)
DSNAME=XLJLR.0VERLAY.L1750.DATA	(File 4 of NAST04)
DSNAME=XLJLR.INCLUDE.L1750.DATA	(File 5 of NAST04)
DSNAME=XLJLR.UMF.PRINT.L1750.DATA	(File 3 of NAST05)

(The following 4 files are extraneous maintenance contractor files)

DSNAME=XLJLR.NSTNLM0D.DUMMY.L0AD
DSNAME=XLJLR.UMF.BULK.L1750.DATA
DSNAME=XLJLR.UMF.DRIVER.L1750.DATA
DSNAME=XLJLR.UMF.BULK.L1700.DATA

NASTRAN ON THE IBM SYSTEM 360-370

NAST08 - DISK DUMP OF DISKM2 (VOLUME SERIAL NUMBER XLJLRJ)

This tape is a disk dump of an IBM 2314 disk pack (using the IBM Utility Program IEHDASDR). This tape contains the data set of File 1 of NAST02. NAST08 consists of one volume and one file.

File 1 - DSNAME=DASDRMO

The following data set is restored from this tape:

DSNAME=XLJLR.MISØURCE.L1750.AM.DATA (File 1 of NAST02)

NAST09 - DISK DUMP OF DISKMF (VOLUME SERIAL NUMBER XLJLRK)

This tape is a disk dump of an IBM 2314 disk pack (using the IBM Utility Program IEHDASDR). This tape contains the data set of File 1 of NAST03. NAST09 consists of one volume and one file.

File 1 - DSNAME=DASDRMO

The following data set is restored from this tape:

DSNAME=XLJLR.MISØURCE.L1750.NZ.DATA (File 1 of NAST03)

NAST10 - DISK DUMP OF DISKM6 (VOLUME SERIAL NUMBER XLJLRL)

This tape is a disk dump of an IBM 2314 disk pack (using the IBM Utility Program IEHDASDR). This tape contains the data set of File 6 of NAST04 and one extraneous maintenance contractor file. NAST10 consists of one volume and one file.

File 1 - DSNAME=DASDRMO

The following data sets are restored from this tape:

DSNAME=XLJLR.ØBJMØD.L1750.LØAD (File 6 of NAST04)

The following file is an extraneous maintenance contractor file:

DSNAME=XLJLR.SØURCE.A1750.DATA

NAST11 - DISK DUMP OF DISKMO (VOLUME SERIAL NUMBER XLJLRM)

This tape is a disk dump of an IBM 2314 disk pack (using the IBM Utility IEHDASDR). This tape contains the data sets of: Files 2 and 3 of NAST05, File 1 of NAST06, and four extraneous maintenance contractor files. NAST11 consists of one volume and one file.

File 1 - DSNAME=DASDRMO

The following data sets are restored from this tape:

DSNAME=XLJLR.UMF.PRINT.L1750.DATA (File 3 of NAST05)

DSNAME=XLJLR.UMF.DRIVER.L1750.DATA (File 2 of NAST05)

DSNAME=XLJLR.DEMØ.PRINT.L1750.DATA (File 1 of NAST06)

The following four files are extraneous maintenance contractor files:

DSNAME=XLJLR.UTILITY.MACLIB.L1750.ASM

DSNAME=XLJLR.UTILITY.SØURCE.L1750.DATA

DSNAME=XLJLR.DRIVER.SAVE.DATA

DSNAME=XLJLR.LMØDS.DATA

NASTRAN - OPERATING SYSTEM INTERFACES

NAST12 - Source Compilations (VOLUME SERIAL NUMBER XLJLR1)

This tape contains the SYSPRINT file generated by version 2.2 of the FØRTRAN H-extended compiler for the machine-independent subroutines ADR through CMTRCE. NAST12 consists of one volume and one file.

File 1 - Source Compilations - Sequential file.

This tape has the following DCB parameters:

DCB=(RECFM=VBA,LRECL=137,BLKSIZE=1922)

NAST13 - Source Compilations (VOLUME SERIAL NUMBER XLJLR2)

This tape contains the SYSPRINT file generated by version 2.2 of the FØRTRAN H-extended compiler for the machine-dependent subroutines CNORM through DQUAD. NAST13 consists of one file.

File 1 - Source Compilations - Sequential file.

This tape has the following DCB parameters:

DCB=(RECFM=VBA,LRECL=137,BLKSIZE=1922)

NAST14 - Source Compilations (VOLUME SERIAL NUMBER XLJLR3)

This tape contains the SYSPRINT file generated by version 2.2 of the FØRTRAN H-extended compiler for the machine-independent subroutines DRAW through FNXTVC. NAST14 consists of one file.

File 1 - Source Compilations - Sequential file.

This tape has the following DCB parameters:

DCB=(RECFM=VBA,LRECL=137,BLKSIZE=1922)

NAST15 - Source Compilations (VOLUME SERIAL NUMBER XLJLR4)

This tape contains the SYSPRINT file generated by version 2.2 of the FØRTRAN H-extended compiler for the machine-independent subroutines FØRFIL through IFX7BD. NAST15 consists of one file.

File 1 - Source Compilations - Sequential file.

This tape has the following DCB parameters:

DCB=(RECFM=VBA,LRECL=137,BLKSIZE=1922)

NAST16 - Source Compilations (VOLUME SERIAL NUMBER XLJLR5)

This tape contains the SYSPRINT file generated by version 2.2 of the FØRTRAN H-extended compiler for the machine-independent subroutines IHEX through MCRØD. NAST16 consists of one volume and one file.

File 1 - Source Compilations - Sequential file.

This tape has the following DCB parameters:

DCB=(RECFM=VBA,LRECL=137,BLKSIZE=1922)

NAST17 - Source Compilations (VOLUME SERIAL NUMBER XLJLR6)

This tape contains the SYSPRINT file generated by version 2.2 of the FØRTRAN H-extended compiler for the machine-independent subroutines MDUM1 through ØUTPT4. NAST17 consists of one volume and one file.

File 1 - Source Compilations - Sequential file.

This tape has the following DCB parameters:

DCB=(RECFM=VBA,LRECL=137,BLKSIZE=1922)

NAST18 - Source Compilations (VOLUME SERIAL NUMBER XLJLR7)

This tape contains the SYSPRINT file generated by version 2.2 of the FØRTRAN H-extended compiler for the machine-independent subroutines PABS through RULER. NAST18 consists of one volume and one file.

File 1 - Source Compilations - Sequential file.

This tape has the following DCB parameters:

DCB=(RECFM=VBA,LRECL=137,BLKSIZE=1922)

NAST19 - Source Compilations (VOLUME SERIAL NUMBER XLJLR8)

This tape contains the SYSPRINT file generated by version 2.2 of the FØRTRAN H-extended compiler for the machine-independent subroutines SADD through SSWTCH. NAST19 consists of one volume and one file.

File 1 - Source Compilations - Sequential file.

This tape has the following DCB parameters:

DCB=(RECFM=VBA,LPECL=137,BLKSIZE=1922)

NAST20 - Source Compilations (VOLUME SERIAL NUMBER XLJLR9)

This tape contains the SYSPRINT file generated by version 2.2 of the FØRTRAN H-extended compiler for the machine-independent subroutines STEP through TYPINT. NAST20 consists of one volume and one file.

File 1 - Source Compilations - Sequential file.

This tape has the following DCB parameters:

DCB=(RECFM=VBA,LRECL=137,BLKSIZE=1922)

NAST21 - Source Compilations (VOLUME SERIAL NUMBER XLJLRA)

This tape contains the SYSPRINT file generated by version 2.2 of the FØRTRAN H-extended compiler for the machine-independent subroutines UMFEDT through ZJ. NAST21 consists of one volume and one file.

File 1 - Source Compilations - Sequential file.

This tape has the following DCB parameters.

DCB=(RECFM=VBA,LRECL=137,BLKSIZE=1922)

NASTRAN - OPERATING SYSTEM INTERFACES

NAST22 - Source Compilations (VOLUME SERIAL NUMBER XLJLRB)

This tape contains the SYSPRINT file generated by version 2.2 of the FORTRAN H-extended compiler for the machine-dependent subroutines CDMPSD through XFLSTS and the SYSPRINT file generated by the GSFC release 21MAR76 of Assembler G compiler for subroutines ACCNT through WALTIM.

File 1 - Source Compilations - Sequential file.

This file has the following DCB parameters:

DCB=(RECFM=VBA,LRECL=137,BLKSIZE=1922)

File 2 - Source Assembly Compilations - Sequential file.

This file has the following DCB parameters:

DCB=(RECFM=FBM,LRECL=121,BLKSIZE=1936)

THIS PAGE HAS BEEN LEFT BLANK INTENTIONALLY.

5.3.7.2 Installation of the NASTRAN Physical Items

The NASTRAN physical items may be installed on an individual data set basis via the IBM utility program IEHMØVE or by restoring the NASTRAN disk packs with the IBM utility program IEHDASDR. Originally, the NASTRAN physical items resided on IBM 2314 disk packs and the SPACE and DCB parameters reflect that environment. IEHMØVE is capable of restoring to all types of disk packs, however, the data sets should be reblocked for optimization of disk space. In the following examples, it is assumed that the data sets will be restored to their original medium.

Loading Individual Data Sets

All individual data sets originally were created with no secondary SPACE allocation. Even though some extra space was allocated for possible expansion, you may want to pre-allocate the data sets yourself via the IBM program IEFBR14 before restoring them with IEHMØVE. The following example illustrates the pre-allocation of the NASTRAN executable data set allowing for a secondary SPACE allocation of 100 tracks.

```
//ALØCATE EXEC PGM=IEFBR14
//ALLØCATE DD UNIT=2314,DISP=(,KEEP),VOL=SER=RSRABB,
//          DSN=XLJLR.NSTNLMØD.L1750.LØAD,
//          DCB=(DSØRG=PØ,RECFM=U,BLKSIZE=7294),
//          SPACE=(TRK,(2500,,10))
```

The following JCL may serve as an example to reload the partitioned data sets unloaded by IEHMØVE. When using this JCL for the machine independent source and the non-executable load modules (both data sets contain more than 600 members) it will be necessary to increase the SPACE allocated to the work data set to 80 contiguous tracks and to specify PARM='PØWER=2' on the EXEC card associated with IEHMØVE.

```
//WØRKDSA EXEC PGM=IEFBR14
//WØRKDS DD UNIT=2314,SPACE=(TRK,(40),,CØNTIG)
//IEHMØVE EXEC PGM=IEHMØVE
//SYSPRINT DD SYSØUT=A
//SYSUT1 DD UNIT=2314,VØL=SER=*.WØRKDSA.WØRKDS,DISP=ØLD
//DD1 DD UNIT=2314,VØL=SER=RSRABB,DISP=ØLD
//TAPE1 DD UNIT=2400,VØL=SER=XLJLRC,DISP=ØLD,LABEL=(1,SL),
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=800,DEN=3),
//          DSN=XLJLR.NSTNLMØD.L1750.LØAD
//SYSIN DD *
COPY PDS=XLJLR.NSTNLMØD.L1750.LØAD,FRØM=2400=NASTØ1,TØ=2314=RSRABB,FRØMDD=TAPE1
//
```

Restoring the NASTRAN Disk Packs

The following JCL may serve as an example to restore the NASTRAN disk packs dumped to tape by IEHDASDR.

```
//T2DISK EXEC PGM=IHEDASDR
//SYSPRINT DD SYSOUT=A
//TAPE DD UNIT=2400,VOL=SER=XLJLRJ,DISP=OLD,DSN=DISKM2,
DCB=DEN=3,LABEL=(1,SL)
//DISK DD UNIT=2314,VOL=SER=DISKM2,DISP=OLD
//SYSIN DD *
RESTORE FROMDD=TAPE,TODD=DISK,CPYVOLID=NØ
//
```

5.3.7.3 Procedure for Updating NASTRAN

Updating NASTRAN may require one or more of the following:

1. Updating the overlay control cards (SUBSYS)
2. Updating NASTRAN's macro library (MACLIB)
3. Updating assembly source code (ASSOURCE)
4. Updating machine-dependent FORTRAN code (MDSOURCE)
5. Updating machine-independent FORTRAN code (MISOURCE)

Due to different installation configurations and requirements, Table 2 and the flowchart in Figure 3 are given as a skeleton of how updates may be made. Each step in the flowchart that requires using a program will have a number in parenthesis that will be an index into Table 2. Table 2 will specify the program required and the DDNames and PARM information.

NASTRAN - OPERATING SYSTEM INTERFACES

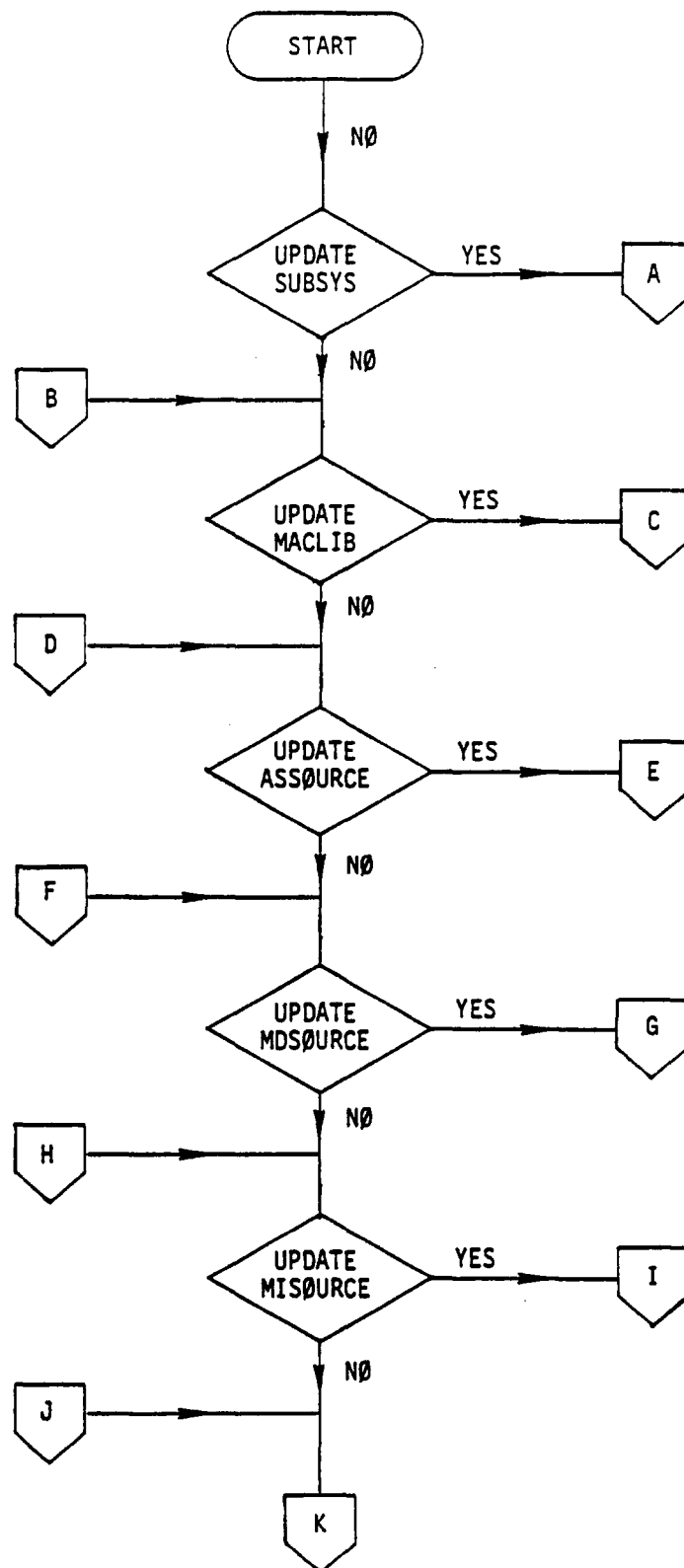


Figure 3. Flowchart of Updating Procedures for NASTRAN

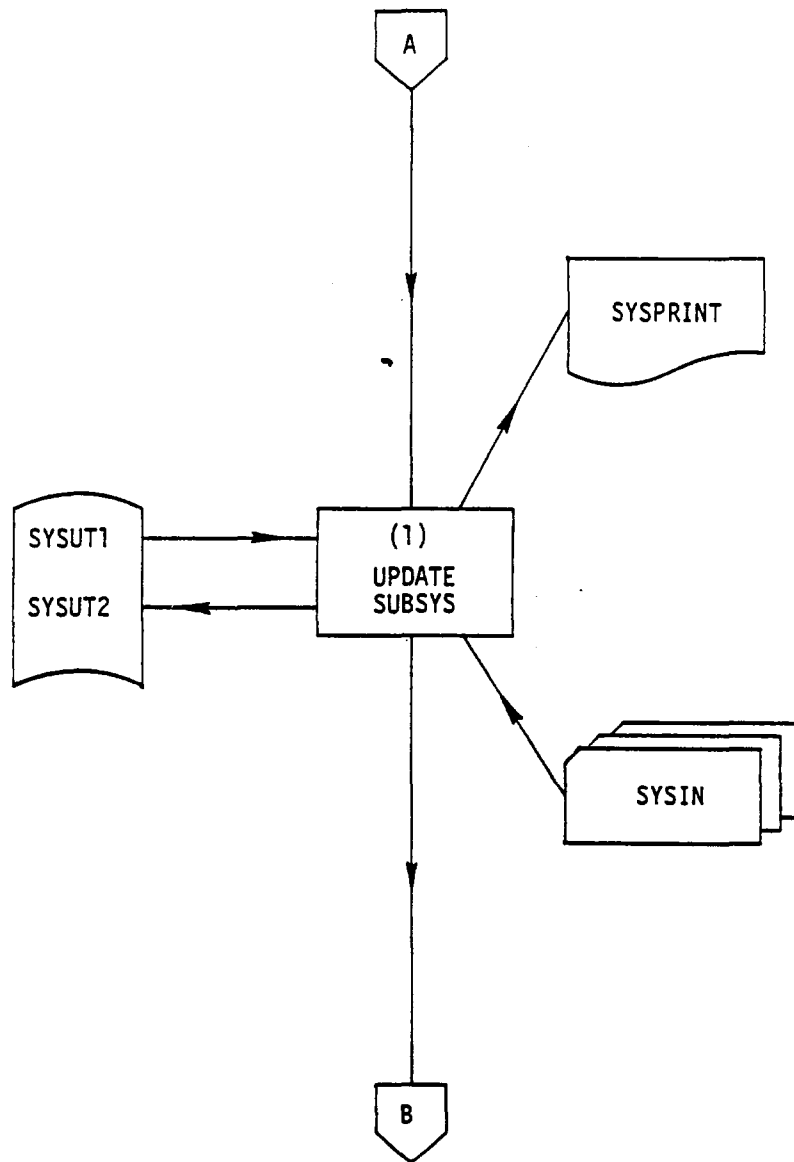


Figure 3(a). Flowchart of Updating Procedures for NASTRAN

NASTRAN - OPERATING SYSTEM INTERFACES

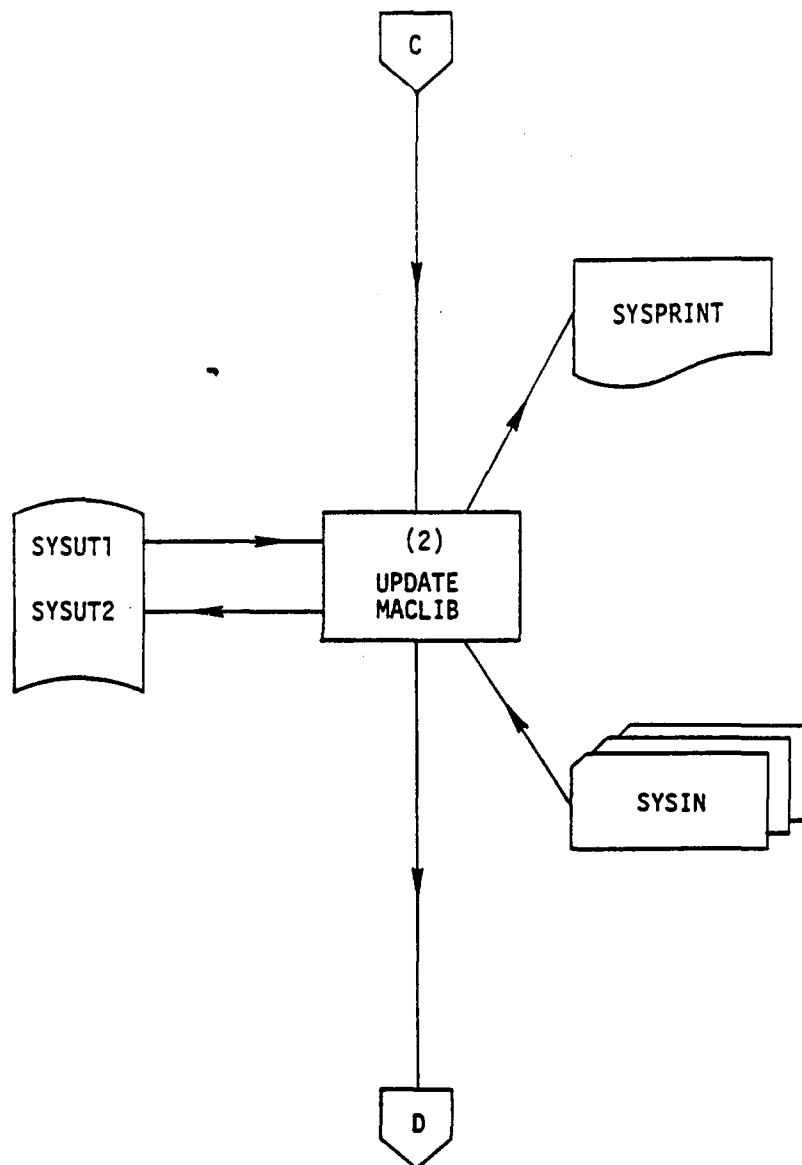


Figure 3(b). Flowchart of Updating Procedures for NASTRAN

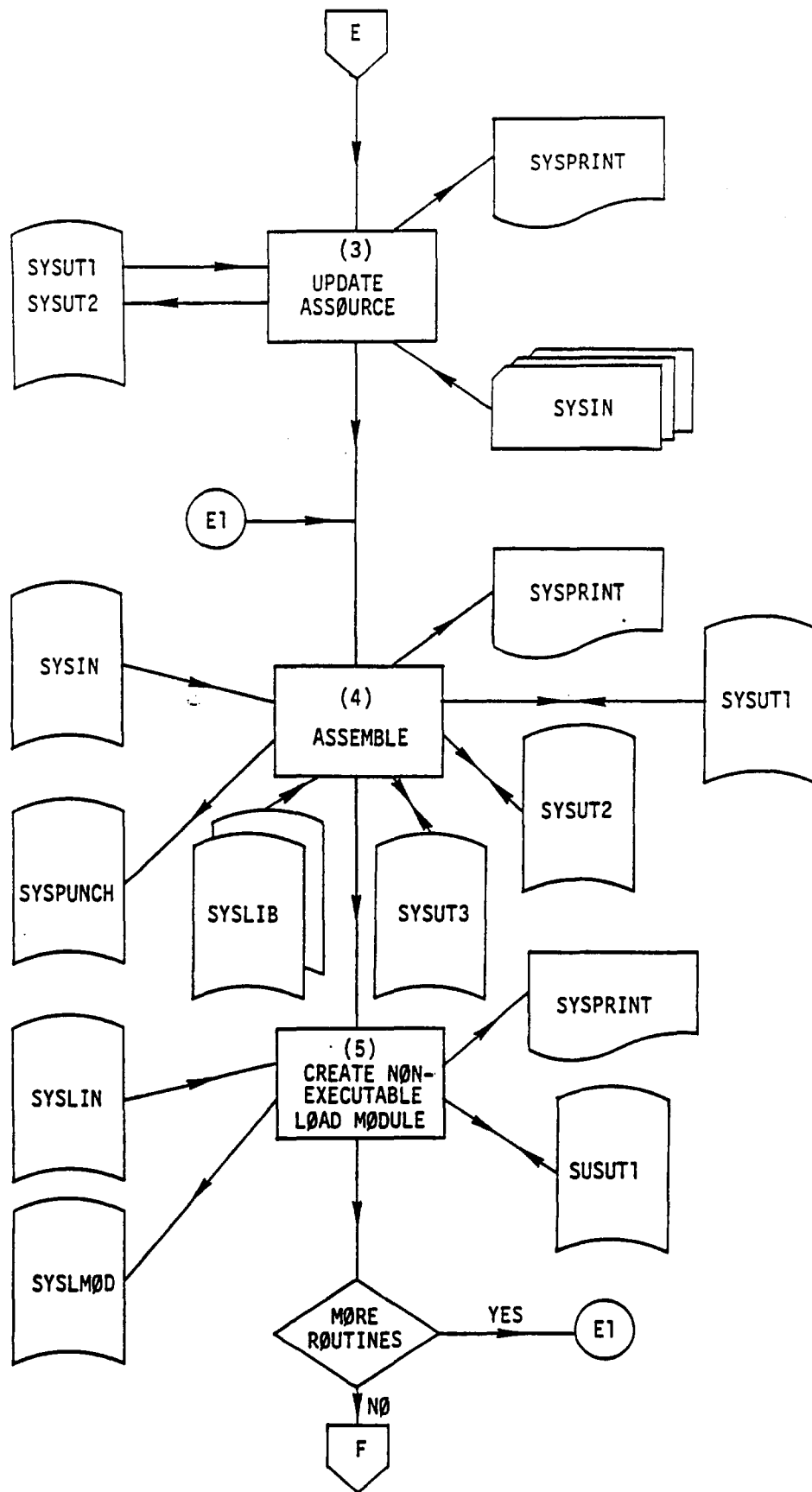


Figure 3(c). Flowchart of Updating Procedures for NASTRAN

NASTRAN - OPERATING SYSTEM INTERFACES

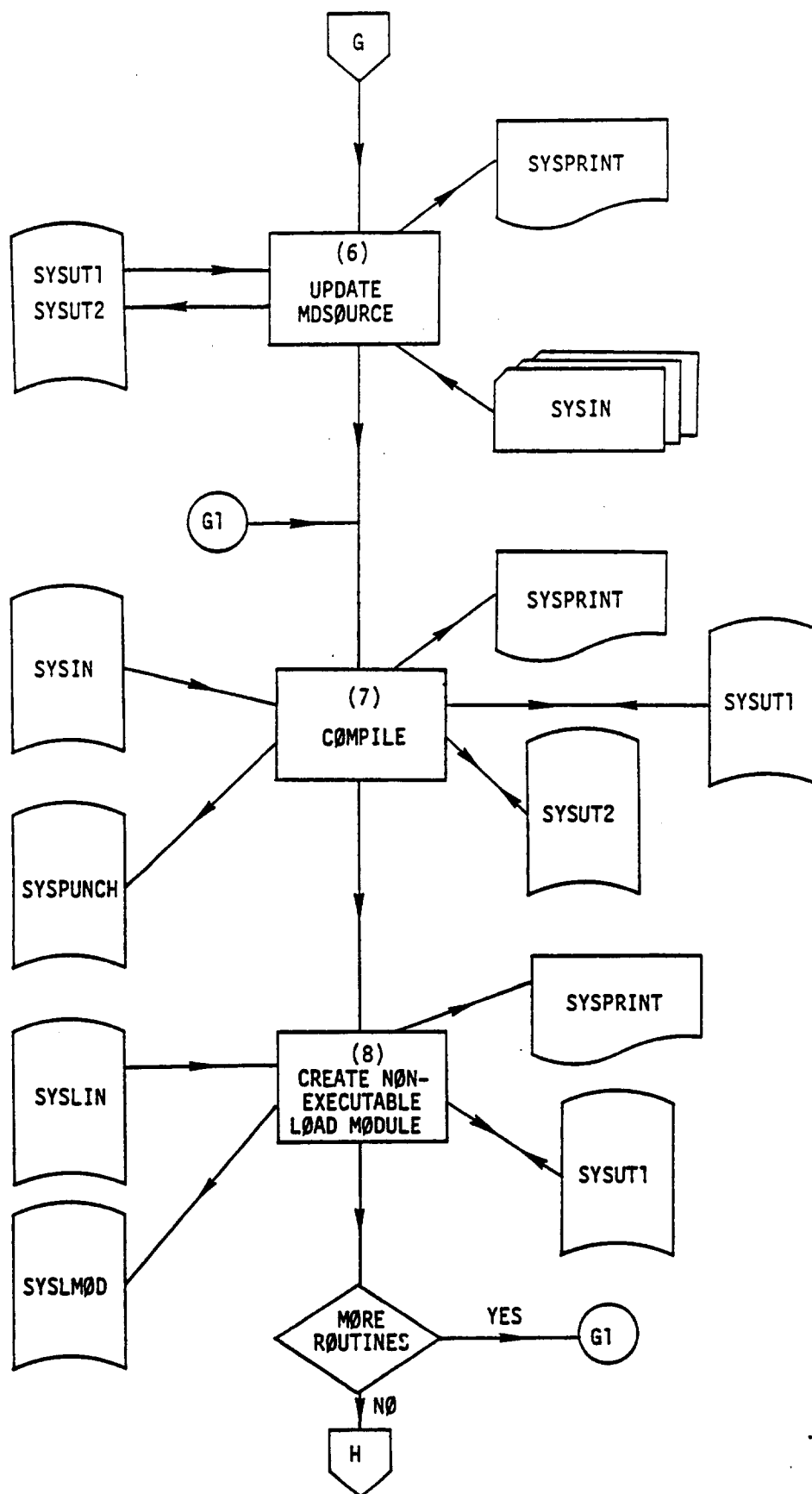


Figure 3(d). Flowchart of Updating Procedures for NASTRAN

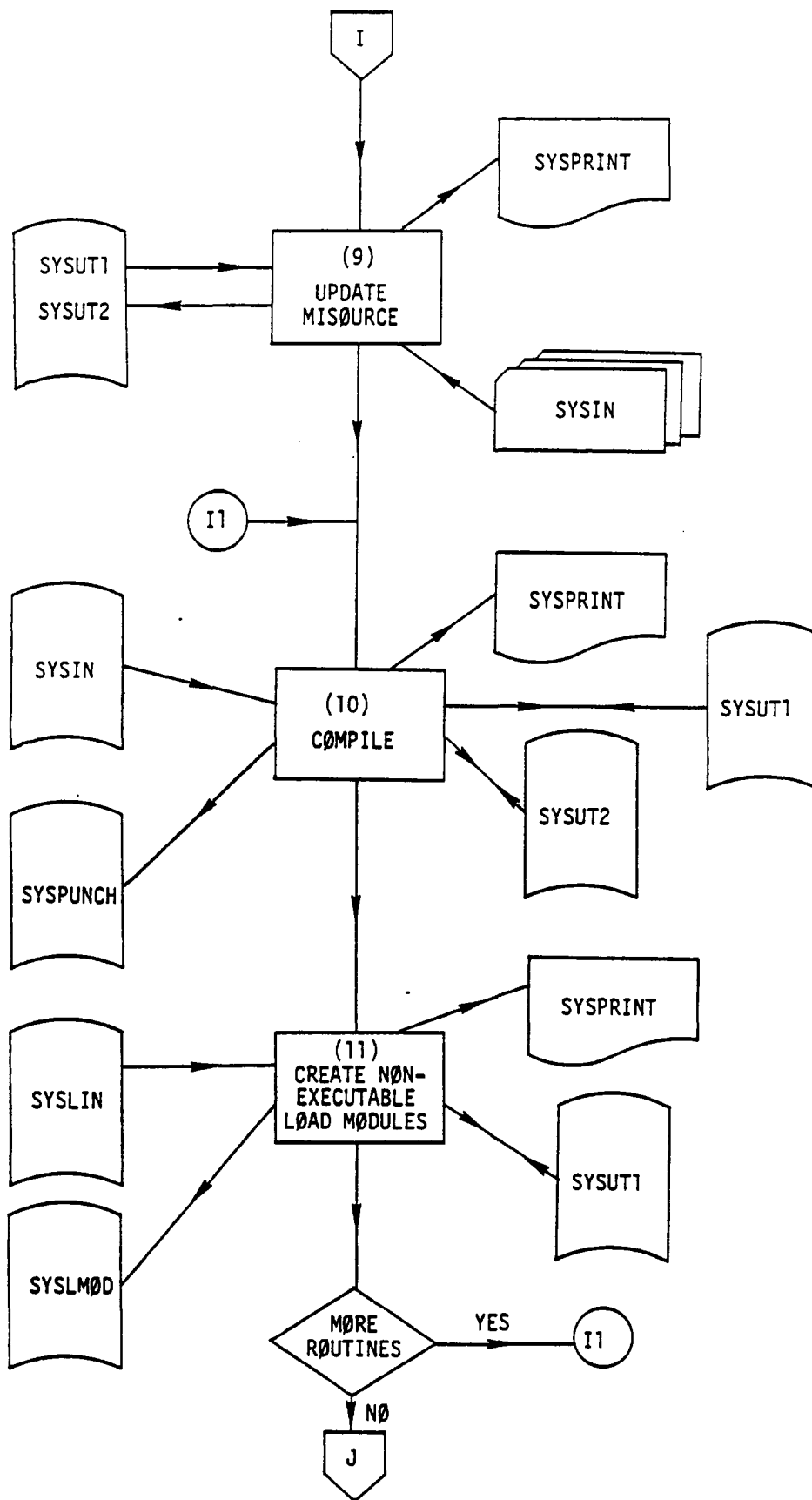


Figure 3(e). Flowchart of Updating Procedures for NASTRAN

NASTRAN - OPERATING SYSTEM INTERFACES

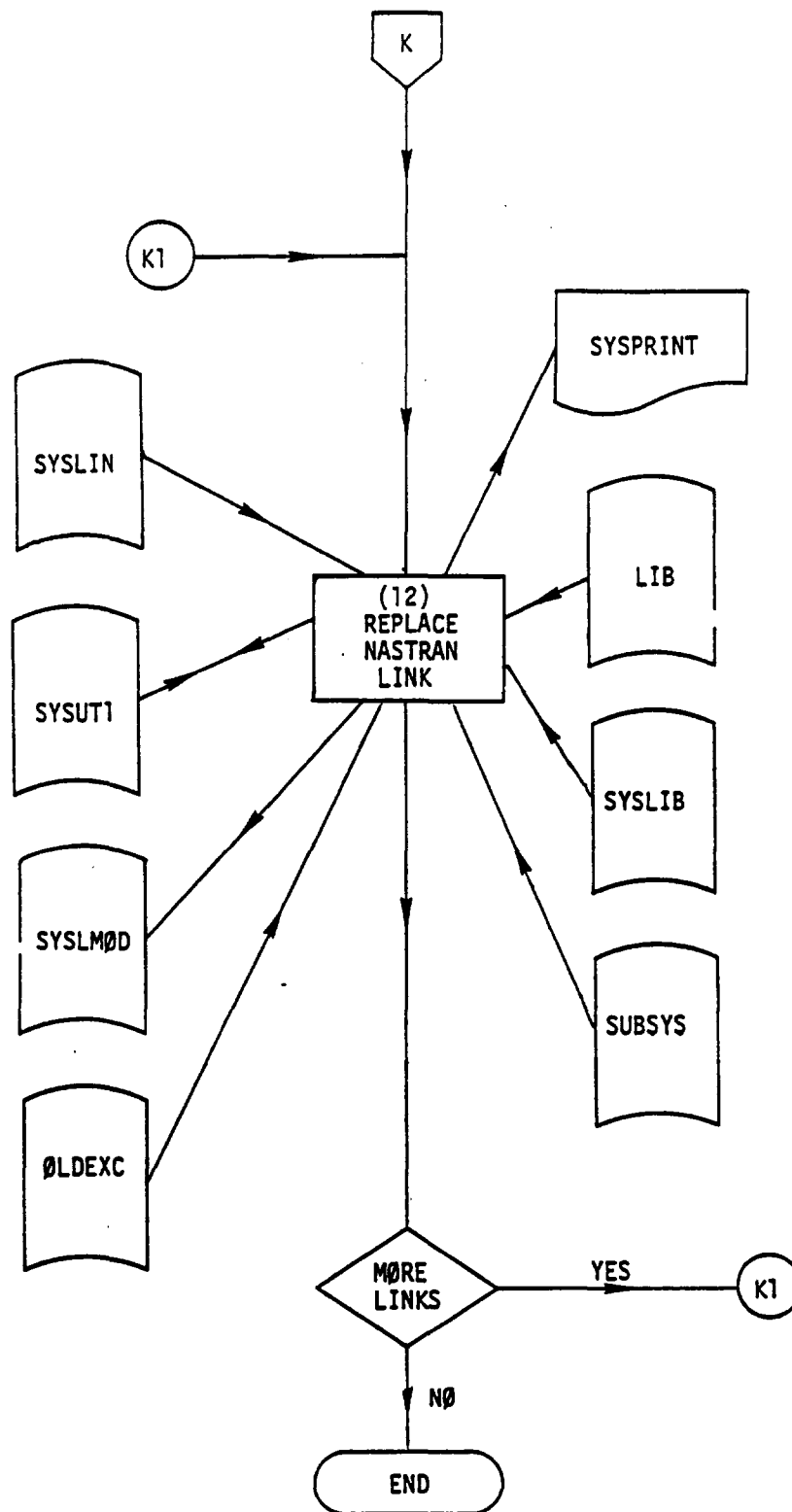


Figure 3(f). Flowchart of Updating Procedures for NASTRAN

NASTRAN ON THE IBM SYSTEM 360-370

<u>Index</u>	<u>Program</u>	<u>DDNAMES and PARM</u>
1	IEBUPDTE	PARM: NØNE DDNAMES: SYSPRINT - Output print file SYSUT1 - DSN=XLJLR.ØVERLAY.L1750.DATA SYSUT2 - DSN=XLJLR.ØVERLAY.L1750.DATA SYSIN - Alter cards
2	IEBUPDTE	PARM: NØNE DDNAMES: SYSPRINT - Output print file SYSUT1 - DSN=XLJLR.NASTRAN.MACLIB.ASM SYSUT2 - DSN=XLJLR.NASTRAN.MACLIB.ASM SYSIN - Alter cards
3	IEBUPDTE	PARM: NØNE DDNAMES: SYSPRINT - Output print file SYSUT1 - DSN=XLJLR.ASSØURCE.L1750.DATA SYSUT2 - DSN=XLJLR.ASSØURCE.L1750.DATA SYSIN - Alter cards
4	IEUASM	PARM: DECK, NØLOAD, XREF DDNAMES: SYSPRINT - Output print file SYSUT1 - Work file SYSUT2 - Work file SYSUT3 - Work file SYSLIB - A concatenation of the System macro library (SYS1.MACLIB) and the NASTRAN macro library (XLJLR.NASTRAN.MACLIB.ASM). NOTE: The NASTRAN maclib and the System mac- lib must have the same DCB attributes. It may be necessary to reblock the NASTRAN maclib. SYSIN - A member of XLJLR.ASSØURCE.L1750.DATA that was updated. SYSPUNCH - This DDNAME should define a data set to con- tain the object module to be passed to the linkage editor.
5	IEWL	PARM: NCAL, LET DDNAMES: SYSPRINT - Output print file SYSUT1 - Work file SYSLIN - This DDNAME should define the data set con- taining the object module passed from the SYSPUNCH file of the step above. SYSLMØD - A member of XLJLR.ØBJMØD.ALTER.LØAD to be replaced.

Table 2. Programs Required for Updating NASTRAN

NASTRAN - OPERATING SYSTEM INTERFACES

<u>Index</u>	<u>Program</u>	<u>DDNAMES and PARM</u>
6	IEBUPDTE	PARM: NONE DDNAMES: SYSPRINT - Output print file SYSUT1 - DSN=XLJLR.MDSOURCE.L1750.DATA SYSUT2 - DSN=XLJLR.MDSOURCE.L1750.DATA SYSIN - Alter cards
7	IEKAA00	PARM: BCD, DECK, NOLoad, ID, XREF, OPT=2 DDNAMES: SYSPRINT - Output print file SYSUT1 - Work file SYSUT2 - Work file SYSIN - A member of XLJLR.MDSOURCE.L1750.DATA that was updated SYSPUNCH - This DDNAME should define a data set to contain the object deck to be passed to the linkage editor.
8	IEWL	PARM: NCAL, LET DDNAMES: SYSPRINT - Output print file SYSUT1 - Work file SYSLIN - This DDNAME should define the data set containing the object deck passed from the SYSPUNCH file of the step above. SYSLMOD - A member of XLJLR.OBJMOD.ALTER.LOAD to be replaced.
9	IEBUPDTE	PARM: NONE DDNAMES: SYSPRINT - Output print file SYSUT1 - DSN=XLJLR.MISOURCE.L1750.DATA SYSUT2 - DSN=XLJLR.MISOURCE.L1750.DATA SYSIN - Alter cards
10	IEKAA00	PARM: BCD, DECK, NOLoad, ID, XREF, OPT=2 DDNAMES: SYSPRINT - Output print file SYSUT1 - Work file SYSUT2 - Work file SYSIN - A member of XLJLR.MISOURCE.L1750. $\left. \begin{matrix} \text{AM} \\ \text{NZ} \end{matrix} \right\}$ -DATA that was updated. SYSPUNCH - This DDNAME should define a data set to contain the object deck to be passed to the linkage editor.

Table 2(a). Programs Required for Updating NASTRAN

NASTRAN ON THE IBM SYSTEM 360-370

<u>Index</u>	<u>Program</u>	<u>DDNAMES and PARM</u>
11	IEWL	<p>PARM: NCAL, LET</p> <p>DDNAMES: SYSPRINT - Output print file</p> <p> SYSUT1 - Work file</p> <p> SYSLIN - This DDNAME should define the data set containing the object deck passed from the SYSPUNCH file of the step above.</p> <p> SYSLMØD - A member of XLJLR.ØBJMØD.ALTER.LØAD to be replaced.</p>
12	IEWL	<p>PARM: ØVLY, LET, MAP, LIST</p> <p>NOTE: It may be necessary to specify the SIZE parameter. The SIZE that is specified will depend upon the level of linkage editor as well as the blocking factors of the primary input data sets and the SYSPRINT file.</p> <p>DDNAMES: SYSPRINT - Output print file</p> <p> SYSUT1 - Work file</p> <p> SYSLIB - DSN=SYS1.FØRTLIB</p> <p> LIB - DSN=XLJLR.ØBJMØD.ALTER.LØAD</p> <p> SYSLIN - Linkage editor control cards. These control cards should include the following:</p> <p> INCLUDE LIB (subroutine(s) being altered)</p> <p> INCLUDE ØLDEXC (LINKNSxx)</p> <p> INCLUDE SUBSYS (LINKNSxx)</p> <p> NAME LINKNSxx(R)</p> <p> SYSLMØD - DSN=XLJLR.NSTNLMØD.ALTER.LØAD</p> <p> ØLDEXC - DSN=XLJLR.NSTNLMØD.L1750.LØAD</p> <p> SUBSYS - DSN=XLJLR.ØVERLAY.L1750.DATA</p>

Table 2(b). Programs Required for Updating NASTRAN

5.3.8 Machine Dependent Routines

The routines discussed in this section consist of those programs unique to the IBM 360 or those routines which are implemented differently on the IBM 360. The language for each deck is indicated by the letter F for the FORTRAN decks or the letter A for the assembly language decks following the deck name. GINØ and GINØ related routines are discussed in Section 5.3.9.

1. EJDUM2 (A)

EJDUM2 initializes and establishes open core (see Section 5.3.4). It is used for each functional link.

2. MAPFNS (A)

In addition to the standard functions described in Section 3, the following were added:

a. TDATE(LØC)

Subroutine TDATE returns the date in a three word integer array LØC where:

LØC(1)=Month
LØC(2)=Day
LØC(3)=Year

b. WHEN(WCLØCK)

Subroutine WHEN executes the ØS TIME macro, which returns time in seconds after midnight. The initial call saves the time which is then used to calculate the elapsed time. WHEN returns the time in integer seconds after the initial call.

c. NØW(TTIME)

The NØW routine initializes and uses the interval timer on S/360 configurations with the timer option. NØW sets and interrogates the interval timer through the STIMER and TTIMER ØS macros. The STIMER macro is executed with the TASK option which specifies that the timer is to operate only when the task is active. This has the effect of measuring the CPU operation time for a particular task. NØW returns the time in integer seconds from the initial call.

d. PDUMPX(SWITCH)

Subroutine PDUMPX through execution of the SNAP macro produces a problem program memory dump or a trace table depending on the value of SWITCH.

SWITCH=1 - Produces a complete problem program dump.

SWITCH=0 - Produces only a trace table

The resulting dump or trace table is output on the SNAPSHOT data set.

e. TYPWRT(MESSAGE)

This subroutine will print up to eighty-character messages on the operator console defined in the variable MESSAGE.

f. DUMPME

Subroutine DUMPME produces a dump of the problem area of core and terminates the run.

3. TYPØUT (A)

The TYPØUT routine is used to print an eighty-character message on the typewriter.

4. WALTIM (A)

Subroutine WALTIM executes the ØS macro TIME and returns the time of day in integer seconds after midnight in the calling argument variable.

5. CØMPBD (F)

CØMPBD is a block data routine to initiate the common block /CDCMPX/.

6. CØNMSG (F)

This routine has several entry points other than the primary entry point CØNMSG. The description of the entry points follow.

a. CØNMSG(BUF,N,K)

The primary entry point for this routine, CØNMSG, is used to write messages on the operator's console and on unit four run log. The message is defined by the input variable BUF, the length of the message is defined by the variable N, and K is used as a switch to determine whether the message is to be written to the operator's console where:

K=0 Do not write message on console

K≠0 Write message on operator's console

b. DUMP

The DUMP routine calls the DUMPME routine described under MAPFNS above to terminate the run.

c. PDUMP

The PDUMP routine calls the PDUMPX routine described above under MAPFNS and produces a dump or traceback depending on whether DIAG 1 is set or not. If DIAG 1 is set a full dump is produced, otherwise only a traceback is produced.

d. CLØCK, ØPRMES, REWNLD

These three routines are dummy routines only. They are included here to avoid having unresolved external references from the linkage editor, and are not used in NASTRAN on the IBM 360.

e. KLØCK(ITIME)

The KLØCK routine uses the NØW routine described under MAPFNS above to return the CPU time in integer seconds in the variable ITIME.

f. SECØND(TIME)

The SECØND routine uses the NØW routine described under MAPFNS above to return the CPU time in the floating point variable TIME.

NASTRAN - OPERATING SYSTEM INTERFACES

7. KØRSZ (F)

The integer function KØRSZ calls the function XCØRSZ in MAPFNS and returns the length of open core. The value of the function is printed on unit four if DIAG 13 is supplied.

8. DCMPCD (F)

DCMPCD is a block data routine to initialize the /DCØMPX/ common block.

9. IØMSG (F)

The IØMSG routine is used by the NASTIØ routine, described in Section 5.3.8, to write the messages needed by that routine.

10. ØPMESG (F)

The ØPMESG routine calls the TYPØUT routine described above to write messages on the operator console.

11. PXIT36

The PXIT36 routine is executed at the end of each NASTRAN run in order to insure the print buffers are flushed properly.

12. SEMTRN (F)

The SEMTRN routine is used to convert NASTRAN data cards that may be in either BCD or EBCDIC to the BCD card images acceptable to NASTRAN and output them onto unit 1 where NASTRAN processing can read them.

13. TAPSWI (F)

The TAPSWI routine provides for multi-volume processing in NASTRAN. When processing a tape and an EØV indicator is sensed, this routine writes a message to the operator who unloads the first tape and pauses to permit the operator to mount the next tape.

14. CN36BD (F)

Defines single and double precision values for several commonly used arithmetic constants.

15. GPERR (A)

GPERR processes fatal errors arising in GINØ, PACKUNPK, NASTIØ, etc. Selected areas of core storage are printed to assist in analysis of the errors.

16. HEXDMP (F)

HEXDMP is called by GPERR to print a message and/or print an array in a hexadecimal format.

17. IØSTAT (A)

IØSTAT is called to collect and print statistics regarding usage of direct access storage space assigned. Actual printing of the statistics is accomplished by calling IØMSG.

18. SØFIØ (F)

SØFIØ directs the input/output operations for substructuring between core and the random access storage device (see Section 3.6). The calling sequence is:

CALL SØFIØ (IRW,IBLKNM,IBUFF)

SØFIØ writes (IRW=1) or reads (IRW=2) block number IBLKNM from/to the buffer IBUFF.

19. DEFILE (A)

DEFILE calculates the number of records available on a substructuring SØF file. For Level 17.0.0 this file must be an FTOx file. The calling sequence is:

CALL DEFILE (UNIT,BLKSIZ,NRECS,ERRØR)

where	UNIT -	FØRTRAN UNIT REFERENCE NUMBER (INTEGER)
	BLKSIZ -	BLØCKSIZE IN WØRDS (INTEGER)
	NRECS -	NUMBER ØF RECØRDS FØR THIS UNIT (INTEGER - RETURNED)
	ERRØR -	ERRØR CØDE (INTEGER - RETURNED)
		ERRØR = 0 NØRMAL RETURN
		ERRØR = 1 MISSING DD CARD
		ERRØR = 2 DEVICE IS NØT DIRECT ACCESS
		ERRØR = 3 DEVICE NØT SUPPØRTED
		ERRØR = 4 IMPRØPER DD CARD
		ERRØR = 5 BLØCKSIZE EXCEEDS MAX FØR DEVICE
		ERRØR = 6 INCØMPATABLE BLØCK SIZE

20. PARM (A)

PARM is a general purpose routine for analyzing PARM parameters on the EXEC statement. Relevant information regarding the type of PARM option to be processed and the status of the request are transmitted via the Parm Control List (PCL). A macro

NASTRAN - OPERATING SYSTEM INTERFACES

(PARMD) which maps the PCL is included in the NASTRAN maclib. A macro (PARM) is also included to create a PCL and call the PARM routine. Definition of the PCL follows:

```

*****
*      USECT TO MAP THE PARM PARAMETER CONTROL LIST.      *
*****
*
&NAME      USECT
*
PARMKR      EQU      B'10000000'      KEYWORD REQUEST
PARMKF      EQU      B'01000000'      KEYWORD FOUND
PARMPCR      EQU      B'00100000'      POSITIONAL CHARACTER REQUEST
PARMPCF      EQU      B'00010000'      POSITIONAL CHARACTER FOUND
PARMPIR      EQU      B'00001000'      POSITIONAL INTEGER REQUEST
PARMPIF      EQU      B'00000100'      POSITIONAL INTEGER FOUND
PARMPIM      EQU      B'00000010'      POS. INT. CONTAINED ALPHA MULT.
PARMUPR      EQU      B'00000001'      UPDATE POINTER REQUEST
*
PARMKEE      EQU      B'10000000'      KEYWORD ENDED WITH =
PARMKEP      EQU      B'01000000'      KEYWORD ENDED WITH (
PARMKEP      EQU      B'00100000'      KEYWORD ENDED WITH )
PARMPBP      EQU      B'00010000'      POSITIONAL BEGAN WITH (
PARMPEP      EQU      B'00001000'      POSITIONAL ENDED WITH )
PARMPCS      EQU      B'00000100'      PARENTHESIS CONTAINED SUBFIELD
PARMMPFL      EQU      B'00000010'      MULTIPLE PARENTHESIS FOUND LEFT
PARMMPFR      EQU      B'00000001'      MULTIPLE PARENTHESIS FOUND RIGHT
*
PARMCNT1     DS      1B      CONTROL BYTE 1
PARMCNT2     DS      1B      CONTROL BYTE 2
PARMPLL      DS      1H      PARM LIST LENGTH
PARMKPL      DS      1H      KEYWORD PARAMETER LENGTH
PARMPPL      DS      1H      POSITIONAL PARAMETER LENGTH
PARMPLP      DS      1F      PARM LIST POINTER
PARMKPP      DS      1F      KEYWORD PARAMETER POINTER
PARMPPP      DS      1F      POSITIONAL PARAMETER POINTER
*
MEXIT
MEND

```

21. TPARM (A)

TPARM has an entry point for each keyword that can be coded in the PARM field of the EXEC statement (at present there is only one possible keyword - CØRE). TPARM initializes the PCL for the other entry points and is usually called only by them.

a. CØRE (NCVTCSF,NCVTFCS)

Entry point CØRE returns the core statistics flag (logical) and the amount of core to free (integer).

22. NCVTINIT (A)

NCVTINIT is called by the executive link driver routine (NASTRAN) to initialize the NASTRAN Communication Vector Table (NCVT). NCVTINIT has no arguments, however register one must still point to the EXEC PARM list at the time NCVTINIT is called. A macro

(NCVT) to map the NASTRAN Communication Vector Table is supplied in the NASTRAN maclib. A map of the NCVT is listed below.

```

*****
*          DSECT TO MAP THE NASTRAN COMMUNICATION VECTOR TABLE          *
*****
*
NCVTD      DSECT
.N002      ANOP
*
NCVTMVS    EQU          X'12'
NCVTMFT    EQU          X'20'
*
NCVT        DS          0D          FORCE DOUBLE WORD ALIGNMENT
NCVTJID     DS          D          JORID
NCVTPRS     DS          F          PARM REGISTER SAVEAREA
NCVTPRC     DS          F          PARM RETURN CODE
NCVTLCA     DS          F          LARGEST CORE ADDRESS
NCVTCSF     DS          F          CORE STATISTICS FLAG
NCVTFCS     DS          F          FREE CORE SIZE
NCVTM2      DS          F          CORE MASK 2
NCVTM1      DS          H          CORE MASK 1
NCVTOSM     DS          B          OPERATING SYSTEM MASK
**
**          END OF NASTRAN COMMUNICATION VECTOR TABLE
**

```

23. ACCNT (A)

ACCNT is called by the executive link driver routine (NASTRAN) just before the termination of the NASTRAN program to provide an EXCP (execute channel program) count for all NASTRAN files. ACCNT requires one argument (INTEGER) which is the FORTRAN unit number of the file on which the EXCP summary is written.

24. LINKNSXX (A)

LINKNSXX serves as the entry point for the functional links. All linkage to the FORTRAN I/O package or the NASTRAN I/O package are resolved at execution time by moving the NASTRAN Linkage Control Table (NLCT) from the executive link driver (NASTRAN) to LINKNSXX. A macro which maps the NLCT is provided in the NASTRAN maclib. The NLCT is also listed below:

```

*****
*          DSECT TO MAP THE NASTRAN LINKAGE CONTROL TABLE              *
*****
*
NLCTD      DSECT
.N002      ANOP
NLCT        DS          0F
DATAH      DC          F'0'
LINKNAME    DC          C'LINKNS01'
COREHELD    DC          2F'0'
AIOTBLE     DC          V(IOTABLE)

```

NASTRAN - OPERATING SYSTEM INTERFACES

ANLCT	UC	A(DATAB)
AIBCOM	UC	V(IHCOM#)
AFDIQCS*	UC	V(FDIQCS#)
AINTSW	UC	V(INTSWTCH)
AFIQCS	UC	V(FIQCS#)
AERRM	UC	V(IHOERRM)
AIO360	UC	V(IO360)
AUATBL	UC	V(IHOUATBL)
AFCVTH	UC	V(IHOFVTH)
AFCVZO	UC	V(FCVZOUTP)
AFCVAO	DC	V(FCVAOUTP)
AFCVLO	UC	V(FCVLOUTP)
AFCVIO	UC	V(FCVIOUTP)
AFCVEO	UC	V(FCVEOUTP)
AFCVCO	DC	V(FCVCOUTP)
AINT6SW	DC	V(INT6SWCH)
ADIQCS	DC	V(DIQCS#)
AADJSW	UC	V(ADJSWTCH)
ATRCH	UC	V(IHOTRCH)
AUOPTN	UC	V(IHOUOPT)
BUFFSIZE	UC	F'0'
AGINO	DC	V(GINO)
AOPEN	UC	V(OPEN)
AWRITE	UC	V(WRITE)
AREAD	DC	V(READ)
ACLOSE	DC	V(CLOSE)
ABCKREC	DC	V(BCKREC)
AFWDREC	DC	V(FWDREC)
ASKPFIL	DC	V(SKPFIL)
AREWIND	UC	V(REWIND)
AEOF	UC	V(EOF)
ABLDPK	UC	V(BLDPK)
ABLDPKI	UC	V(BLDPKI)
AZBLPKI	DC	V(ZBLPKI)
ABLDPKN	DC	V(BLDPKN)
AINTPK	DC	V(INTPK)
AINTPKI	UC	V(INTPKI)
AZNTPKI	UC	V(ZNTPKI)
APACK	UC	V(PACK)
AUNPACK	UC	V(UNPACK)
ASAVPOS	DC	V(SAVPOS)
AFILPOS	DC	V(FILPOS)
APUTSTR	UC	V(PUTSTR)
AENDPUT	UC	V(ENDPUT)
AGETSTR	UC	V(GETSTR)
AENDGET	UC	V(ENDGET)
AGETSTB	UC	V(GETSTB)
AENDGTB	UC	V(ENDGTB)
ANASTIO	UC	V(NASTIO)
ARECTYP	DC	V(RECTYP)
AGPERR	DC	V(GPERR)
AHEXDMP	DC	V(HEXDMP)
AHDBLK	DC	V(RDBLK)
AWRTBLK	DC	V(WRTBLK)
ANCVT	UC	A(NCVT)
AGINOX	DC	V(GINOX)
ACOMMONS	DS	10A
AFNAME	DS	A
AMESAGE	DS	A
**		
**	END OF NASTRAN LINKAGE CONTROL TABLE	
**		

LINKNSXX also contains entry points EXIT and SEARCH which provides the return to the executive link at the termination of the functional link.

SEARCH is called to determine the name of the next functional link to be loaded by the executive link driver routine (NASTRAN). The name of the next functional link is placed in the NLCT.

EXIT is called to return to the executive link driver routine (NASTRAN).

25. NASTRAN (A)

NASTRAN is the executive link driver routine and the entry point for the executive link. NASTRAN obtains the original open core and frees back the user specified amount of core for FØRTRAN I/O buffers and for use by the operating system. NASTRAN utilizes the NLCT to provide linkage to the FØRTRAN and NASTRAN I/O packages for use by LINKNSXX. Each functional link is loaded and called by routine NASTRAN and then deleted by NASTRAN when the functional link is no longer needed. A diagram of core allocation is presented in Figure 4. The core areas for the File Control Blocks (FCB), Data Control Blocks (DCB), and Data Event Control Block (DECB) are reserved by GNFIAT when the functional link LINKNSØ1 is loaded and called by NASTRAN.

26. DACHAR (A)

DACHAR is called by GNFIAT and DEFILE and returns the number of blocks per track and the number of tracks per cylinder for a direct access device associated with a given name. The calling sequence is:

```
CALL DACHAR(DDNAME,BLKKL,BLKDL,NBPT,NTPC,$10,$20)
```

where

DDNAME	An eight byte buffer containing the DDNAME left justified with blank fill if necessary
BLKKL	A full word containing the key length if one is specified.
BLKDL	A full word containing the data length
NBPT	A full word containing the number of blocks per track (returned) for the direct access device associated with the given DDNAME.
\$10	Nonstandard return taken if the given DDNAME is not assigned to a direct access device.
\$20	Nonstandard return if the given DDNAME is not found in the task input/output table (TIØT).

27. XFLSTS (F)

XFLSTS (EXecutive File StaticS housekeeper) will save the accumulative block count for each file maintained in the /XFIAT/. XFLSTS is called by entry point CLØSE in subroutine NASTIØ. The calling sequence follows:

CALL XFLSTS(FIAT,GINØX,FIST,FCB,FCBBYT,FRSTPR,FRSTSC,FRSTTR)

where

FIAT	/XFIAT/ common block in LINKNSxx functional link
GINØX	/GINØX/ common block in LINKNSxx functional link
FIST	/XFIST/ common block in LINKNSxx functional link
FCB	File Control Block maintained by NASTIØ
FCBBYT	same as FCB except referenced as bytes
FRSTPR	internal file number of first primary file
FRSTSC	internal file number of first secondary file
FRSTTR	internal file number of first tertiary file

Link NASTRAN
Link LINKNSxx
Open Core
File Control Blocks (FCB) (one FCB for every file)
Data Control Block (DCB) (one DCB for every file)
Data Event Control Block (DECB) (number is maximum number of open files)
FØRTRAN Buffers and Core for the Operating System

Figure 4. Core Allocation for the IBM NASTRAN System

NASTRAN - OPERATING SYSTEM INTERFACES

5.3.9 GINØ (Generalized Input/Output Processor for NASTRAN)

The GINØ package for the IBM 360 consists of the following three decks with entry points as indicated.

<u>DECK</u>	<u>ENTRY POINT</u>
GINØ	BCKREC PUTSTR CLØSE QØPEN ENDGET RDBLK ENDGTB READ ENDPUT RECTYP EØF SAVPØS FILPØS SKPFIL FWDREC WRITE GETSTB WRTBLK GETSTR
NASTIØ	IØ360
ØPEN	ØPEN

Documentation for GINØ and the calling sequence for its entry points are essentially machine independent and are discussed in Section 3.4 of the Programmer's Manual. The documentation for IØ360 is unique to the IBM 360 and is provided as follows:

Purpose

To perform I/Ø operations for NASTRAN data blocks using the Basic Sequential Access Method (BSAM).

Calling Sequence

CALL IØ360(ØPCØDE,BLØCK), where

ØPCØDE = the operation code for the operation to be performed according to the following list:

1. - Rewind
2. - Write
3. - Read
4. - Backrec
5. - Forward rec
6. - Open
7. - Close
8. - Reread
10. - Unload

BLØCK = address of the GINØ block for the requested operation.

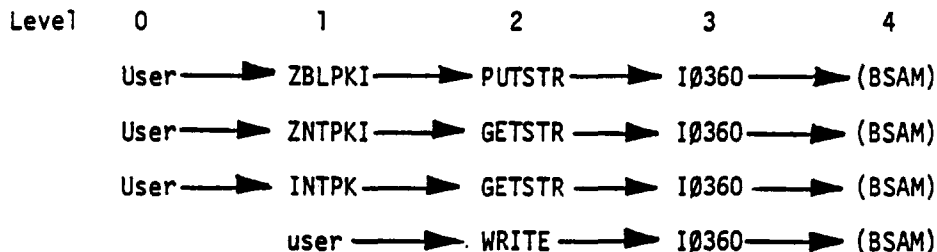
Additional communication is accomplished through the use of the IOTABLE. The IOTABLE is a table defined in NASTIØ (like a common block) which is initialized by GNFIAT and use by NASTIØ. The contents are as follows:

IOTABLE	DS	H	BLKSIZE IN BYTES
	DS	H	1ST UNIT REF NØ. -- PERMANENT UNITS
	DS	H	PRIMARY UNITS
	DS	H	SECONDARY UNITS
	DS	H	TERTIARY UNITS
	DS	H	NUMBER ØF UNITS -- PERMANENT
	DS	H	PRIMARY
	DS	H	SECONDARY
	DS	H	TERTIARY
	DS	H	NUMBER ØF DECB'S
	DS	A	ADDR ØF FILE CØNTRØL BLØCKS (FCB)
	DS	A	ADDR ØF DATA CØNTRØL BLØCKS (DCB)
	DS	A	ADDR ØF DATA EVENT CØNTRØL BLØCKS (DECB)
	DS	H	FCB LENGTH IN BYTES
	DS	H	DCB LENGTH IN BYTES
	DS	H	DECB LENGTH IN BYTES
	DS	H	SPARE
	DS	A	ADDRESS ØF TASK I/Ø TABLE (TIØT)

This table is only referenced by NASTIØ and GNFIAT.

Method

The NASTIØ routine performs transfers of record blocks between main storage and secondary storage and performs positioning of secondary storage devices. All I/Ø operations are performed using the macros of the Basic Sequential Access Method (BSAM) such as ØPEN, CLØSE, READ, WRITE, CHECK, and PØINT. Fields of the appropriate FCB are updated for each operation. Some examples of call chains for NASTIØ are:



Note: Most calls are serviced at levels 1 and 2.

Calls to level 3 occur when a block is to be transferred between main and secondary storage. All calls to level 3 result in at least one call to level 4.

NASTRAN - OPERATING SYSTEM INTERFACES

It should be noted that when a write operation is scheduled which would exceed the space allocated to the unit, the pool of secondary units is examined to find an available unit. If a unit is available, it is attached to the primary, and the write operation is made on the secondary. Correspondingly, if a secondary unit fills, another secondary unit is attached. When two secondary units are filled, NASTRAN attaches a tertiary unit, if available, and only when all units are used is an OS extent requested. It should also be noted that each time a file is opened to write with rewind, all secondary and tertiary units previously assigned to the file are released and made available for future assignment to other files (data blocks).

Relevant information regarding I/O operations and status is maintained in the File Control Blocks (FCB). There is one FCB for each permanent, primary, secondary, and tertiary unit as defined in the JCL for the JOB. A description of the FCB is found in Table 3.

Parameter	Length (Bytes)	Description
FCBFRST*	2	Points to first block of core file
FCBCURNT*	2	Points to current block of core file
FCBLAST	2	Last block number of file
FCBFLAGS	1	I/O flag X'80' File has been opened X'40' File positioned at end of data X'20' File is kept in core X'10' File is on tape X'01' File opened to read
FCBDECB	1	Points to DECB assigned to file
FCBBLKNØ	2	Block number at which file is currently positioned
FCBPREV	1	Previous unit assigned to the file Also used as a status byte X'80' Last I/O operation was a write X'40' Last I/O operation was a read X'20' Last operation sequence was "read; backspace" or "write; backspace" X'CO' Last I/O operation was a read or write
FCBNEXT	1	Next unit assigned to the file
FCBLØW	2	First block number of file for this unit
FCBHIGH	2	Last block number of file for this unit
FCBLKPRI	2	Number of blocks in this unit's primary allocation
FCBLKSEC	2	Number of blocks in this unit's secondary allocation
FCBCLAST*	2	Last block number in core for a core file
FCBNBPT	2	Number of blocks per track
FCBBUFF*	4	Address of I/O buffer assigned
FCBMAX	2	Usage statistics
FCBTIØT	2	Offset in TIØT to entry for this file

*Used only for files kept in core

Table 3. Description of the FCB

NASTRAN - OPERATING SYSTEM INTERFACES

5.3.10 Special Error Codes from NASTRAN on the System 360

One of the following two types of error codes will accompany any abnormal end (ABEND) output from NASTRAN operating under OS on the IBM System 360.

1. SYSTEM = XXX

Where XXX is an OS code described in the IBM System Reference Library (SRL), Form 628-6631, titled IBM System/360 Operating System Messages and Codes.

2. USER = YYY

Where YYY is a NASTRAN code described as follows:

<u>USER CODE</u>	<u>SUBROUTINE</u>	<u>CAUSE</u>
001	NASTRAN	A link name has been requested that is beyond the range of those available.
002	GNFIAT	Insufficient core for NASTIØ. Core requirement for NASTIØ equals 26 times the number of non-dummy DD statements.
004	GNFIAT	Core is discontinuous.
005	EJDUM2	Insufficient core - GETMAIN failure.
012	MAPFNS	DUMP was called.
101	GNFIAT	Unable to determine DDNAME to be PRIxx, SECxx, TERxx, or a permanent file (files in /XFIST/).
102	GNFIAT	Unable to find DDNAME in the permanent file list (files in /XFIST/).
200	GNFIAT	Unrecognized device class in DD statement.
254	EJDUM2	FREEMAIN failure.
255	EJDUM2	FREEMAIN failure.
500	NASTRAN	Insufficient core - GETMAIN failure.
501	NASTRAN	FREEMAIN failure.
502	NASTRAN	FREEMAIN failure.
680	MPYQ	Bad string definition word (occurred in entry MPY2NT).
777	MAPFNS	SNAP macro failed.
777	GPERR	SNAP macro failed.
3333	GPERR	Fatal message already printed - job to be terminated.

MATERIAL PREVIOUSLY ON PAGES 5.3-49 THROUGH 5.3-77a HAS BEEN DELETED.

MODULE INDEX	DMAP NAME OF MODULE	MODULE ENTRY POINT NAME	LINKS	MODULE RESIDES IN	ON	360												
3	CHKPNT	XCHK	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
5	REPT	XCEI	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
6	JUMP	XCEI	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
7	COND	XCEI	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
8	SAVE	XSAVE		2	3	4	5	6	7	8	9	10	11	12	13	14	15	
9	PURGE	XPURGE	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
10	EQUIV	XEQUIV	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
11	END	XCEI	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
12	EXIT	XCEI	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
14	ADD	DADD				4			7									
15	ADD5	DADD5							7									
16	AMG	AMG									9							
17	AMP	AMP									9							
18	APD	APD									9							
19	HMG	HMG										10						
20	CASE	CASE										10						
21	CYCT1	CYCT1							7									
22	CYCT2	CYCT2							7									
23	CEAD	CEAD											11					
24	CURV	CURV													13			
26	DDR	DDR								3								
27	DDR1	DDR1													12			
28	DDR2	DDR2													12			
29	DDRMM	DDRMM													12			
30	DECOMP	DUCOMP							7									
31	DIAGONAL	DIAGON															15	
32	DPD	DPD						6										
33	DSCHK	DSCHK							7									
34	DSMG1	DSMG1													13			
35	DSMG2	DSMG2				4												
37	DUMMOD1	DUMMOD1							7									
38	DUMMOD2	DUMMOD2							7									
39	DUMMOD3	DUMMOD3							7									
40	DUMMOD4	DUMMOD4							7									
42	EMA1	EMA1								8								
43	EMG	EMG								6								
44	FA1	FA1												11				
45	FA2	FA2												11				
46	FHS	DFHS							7									
47	FPLG	FPLG												10				
48	FRRD	FRRD												10				
50	GI	GI																
51	GKAD	GKAD												10				
52	GKAM	GKAM												10				
53	GPI	GPI				2												

Figure 20. Link Specification Table.

5.3-78 (12/29/78)

2-44

MODULE INDEX	MAP NAME OF MODULE	MODULE ENTRY POINT NAME	LINKS	MODULE RESIDES IN	ON	360														
54	GP2	GP2	2																	
55	GP3	GP3	2																	
56	GP4	GP4		4																
57	GPCYC	GPCYC				7														
58	GPFDR	GPFDR																	13	
59	GPSP	GPSP		4																
60	GPWG	GPWG		4																
62	INPUT	INPUT	2																	
63	INPU11	INPU11	2																	
64	INPU12	INPU12	2																	
65	INPU13	INPU13	2																	
66	INPU14	INPU14	2																	
67	MATGEN	MATGEN				7														
68	MATGPR	MATGPR								8										
69	MATPRN	MATPRN								8										
70	MATPRT	PRTINT								8										
71	MCE1	MCE1		4																
72	MCE2	MCE2		4																
73	MERGE	MERGE1				7														
75	MODA	MODA				7														
76	MODACC	MODACC																	12	
77	MODH	MODH				7														
78	MODC	MODC				7														
79	MPYAD	DMPYAD				7														
80	MTRXIN	MTRXIN										10								
81	OFF	OFF																	14	
82	OPTPR1	OPTPR1	2																	
83	OPTPR2	OPTPR2								8										
85	OUTPUT	OUTPT																	14	
86	OUTPUT1	OUTPT1																	14	
87	OUTPUT2	OUTPT2																	14	
88	OUTPUT3	OUTPT3																	14	
89	OUTPUT4	OUTPT4																	14	
90	PARAM	QPARAM	2	3	4	5	6	7	8	9	10	11	12	13	14	15				
91	PARAML	PARAML	2	3	4	5	6	7	8	9	10	11	12	13	14	15				
92	PARAMR	QPARAMR	2	3	4	5	6	7	8	9	10	11	12	13	14	15				
93	PARTN	PARTN1						7												
95	MRED1	MRED1																	15	
96	MRED2	MRED2																	15	
97	CARED2	CARED2																	15	
98	PLA1	PLA1		3																
99	PLA2	PLA2																	13	
100	PLA3	PLA3																	13	
101	PLA4	PLA4																	13	
103	PLOT	DPL0T	2																	

Figure 20. Link Specification Table (Continued)

5.3-79 (12/29/78)

NASTRAN - OPERATING SYSTEM INTERFACES

MODULE INDEX	MAP NAME OF MODULE	MODULE ENTRY - LINKS MODULE RESIDES IN ON	POINT NAME
104	PLTSET	2	
105	PLITRA		
106	PRTMSG	2	
107	PRTPRM		
108	RANDOM		
109	RBMG1		14
110	RBMG2		
111	RBMG3		
112	RBMG4		
114	READ		
115	RMG		
116	SCALAR		
117	SCE1		
118	SDH1		
119	SDH2		
120	SDH3		
121	SDHHT		
122	SEEMAT		
124	SETVAL		
125	SMA1		
126	SMA2		
127	SMA3		
128	SMP1		
129	SMP2		
130	SMPYAD		
131	SOLVE		
133	SSG1		
134	SSG2		
135	SSG3		
136	SSG4		
137	SSGHT		
138	TAI		
139	TABPCH		
141	TABFMT		
142	TABPT		
144	TIMETEST		
145	TRU		
146	TRHT		
147	TRLG		
148	TRISP		
149	UMERGE		
150	UPARTN		
151	VDR		
152	VEC		
154	XYPLOT		

Figure 20. Link Specification Table (Continued)

MODULE INDEX	MAP NAME OF MODULE	MODULE ENTRY POINT NAME	LINKS MODULE RESIDES IN ON 360
155	XYPRNPLT	XYPRPT	14
156	XYTRAN	XYTRAN	14
158	COMH1	COMH1	15
159	COMB2	COMB2	15
160	EXIO	EXIO	15
161	RCOVR	RCOVR	15
163	RCOVR3	RCOVR3	15
164	REDUCE	REDUCE	15
165	SGEN	SGEN	15
166	SOFI	SOFI	15
167	SOF0	SOF0	15
168	SOFUT	SOFUT	15
169	SUBPH1	SUBPH1	15
170	PLTMRG	PLTMRG	15
172	COPY	COPY	7
173	SWITCH	SWITCH	7
174	MPY3	MPY3	7
175	SDCMPS	SDCMPS	7
176	LOUAPP	LOUAPP	15
177	GPSPC	GPSPC	4
178	EQMCK	EQMCK	12
179	ADR	ADR	10
180	FRRD2	FRRD2	10
181	GUST	GUST	10
182	IFT	IFT	10
183	LAMX	LAMX	9
184	MTRXTEST	MTRXTS	14
185	EMA	EMA	14

ADRI			10	
ADRPR			10	
ADRX			10	
ADR			10	
AIS	5			13
AI	5			13
ALLMAT				
AMATRX				13
AMGRFS			9	
AMGMN			9	
AMGP2			9	
AMGRND			9	
AMGSBA			9	
AMGX			9	
AMG			9	
AMPA1X			9	
AMPA			9	
AMPB1X			9	
AMPB1			9	
AMPB2X			9	
AMPB2			9	
AMPB			9	
AMPCOM			9	
AMPC1X			9	
AMPC1			9	
AMPC2			9	
AMPC			9	
AMPD1X			9	
AMPD			9	
AMPEX			9	
AMPE			9	
AMPEX			9	
AMPE			9	
AMP			9	
APDCS			9	
APDF			9	
APDNE			9	
APDR			9	
APD12C			9	
APD1			9	
APD1C			9	
APD1D			9	
APD12			9	
APD2			9	
APD3			9	
APD4			9	
APD5			9	

11

Figure 21. Alphabetical Deck Name - Link Table

APD								9											
ARRM						6													
ASCM01	1																		
ASCM02	1																		
ASCM03	1																		
ASCM04	1																		
ASCM05	1																		
ASCM06	1																		
ASCM07	1																		
ASCM08	1																		
ASCM09	1																		
ASCM10	1																		
ASCM11	1																		
ASCM12	1																		
ASCM13	1																		
ASDRD	1																		
ASDMP	1																		
ASDZZZ	1																		
ASPRD	1																		
ATFIG												11							
AUTOCK	1																		
AUTOCM	1																		
AUTOHD	1																		
AUTOSM	1																		
AUTOSV	1																		
AXIS10		2																	
AXIS3		2																	
AXIS		2																	
BARD								8											
BARS								8											
BAR						5													
PASGLB						5													
BCB				3		5			8										
RDAT01																		15	
RDAT02																		15	
RDAT03																		15	
RDAT04																		15	
RDAT05																		15	
RDAT06																		15	
RFSMAT										9									
RINT						5													
RISHEL		2																	
RISLOC	1	2	3		5		7	8	9		10		12		13				
RITPAT																			
RITPOS	1	2	3	4	5	6	7	8	9		10	11	12	13		14		15	
RLANK	1	2	3	4	5	6	7	8	9		10	11	12	13		14		15	
RMGTNS																			
RMGZZZ											10								

Figure 21. Alphabetical Deck Name - Link Table (Continued)

RMG										10					
BORDER		2													
RTSTRP	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RUG	1								9	10					15
RVISC			3					8							
CALCV				4	5	6	7		9	10	11	12			15
CASCOR										10					
CASE										10					
CDCMPX							7		9	10	11	12			15
CDCOMP							7		9	10	11				15
CDETMX											11				
CDETM2											11				
CDETH											11				
CDIFBS											11				
CDIVID											11				
CDTFBS											11				
CFADA1											11				
CFAD1A											11				
CFAD1X											11				
CFAD											11				
CENTRE		2													
CFACTR									9	10	11				
CFACTX									9	10	11				
CFBSOR									9	10	11				
CFBSRX									9	10	11				
CFCNTL											11				
CFEER1											11				
CFEER2											11				
CFEER3											11				
CFEER4											11				
CFER3D											11				
CFER3S											11				
CFE1AD											11				
CFE1MY											11				
CFE2AD											11				
CFE2MY											11				
CFNOR1											11				
CFNOR2											11				
CF1FBS											11				
CF1ORT											11				
CF2FBS											11				
CF2ORT											11				
CHAR94		2													
CHKOPN															15
CHRDW		2													
CINFBS											11				
CINFBX											11				
CINVPR											11				

Figure 21. Alphabetical Deck Name - Link Table (Continued)

CINVPX											11			
CINVP1											11			
CINVP2											11			
CINVP3											11			
CINVX											11			
CINV1X											11			
CINV2X											11			
CINV3X											11			
CLSTAB	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CLSTRS	2	3		5			8				12	13		
CLVEC										11				
CMAUTO														15
CMBFND														15
CMR7ZZ														15
CMR001														15
CMR002														15
CMR003														15
CMR004														15
CMCASE														15
CMCKCD														15
CMCKDF														15
CMCONB														15
CMCONT														15
CMDISC														15
CMHGEN														15
CMIWPT														15
CMHCNN														15
CMRD2A														15
CMRD2B														15
CMRD2C														15
CMRD2D														15
CMRD2E														15
CMRD2F														15
CMRD2G														15
CMRD2Z														15
CMRD2														15
CMRELS														15
CMSEIL														15
CMSDFD														15
CMTIMU										11				
CMTDC														15
CMTPCF														15
CNORM1										11				
CNORM										11				
CNSTRC		2												
COMPIN				5										
CMRAN	1													
CMR1														15

Figure 21. Alphabetical Deck Name - Link Table (Continued)

COMB2X															15
COMB2															15
COMECT		2													
COM12							7		9	10	11				15
CONDAD	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CONDAS	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CDNE					5										
CONMSG	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CONM10								8							
CONM15								8							
CONM20								8							
CONM25								8							
CONTOP		2													
COPYZZ							7								
COPY							7								
CORSZ										10					
CPYFIL	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CPYSTR	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CPDRD2				4											
CPDRD				4											
CREATE		2													
CPIGGP				4											
CRSUB															15
CSORTX											11				
CSUB											11				
CSUMM											11				
CTRNSP							7		9	10	11				15
CURCAS												12			
CURVC1													13		
CURVC2													13		
CURVC3													13		
CURVIT													13		
CURVPS													13		
CURVTR													13		
CUPVXX													13		
CURV1													13		
CURV2													13		
CURV3													13		
CURV													13		
CXLONP							7		9	10	11				15
CYTRNY											11				
CYCT1Z							7								
CYCT1							7								
CYCT2A							7								
CYCT2B							7		9	10	11	12			
CYCT2X							7								
CYCT2							7								
DACHAR	1														

Figure 21. Alphabetical Deck Name - Link Table (Continued)

Command	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DADD5							7								
DADD				4			7								
DADDA				4			7								
DADNTB			3		5			8							
DARCOS								8							
DAXB			3		5			8							
DABP														13	
BCD													13		
DCOMPX				4	5	6	7		9	10	11	12			15
DCONE													13		
DCMPS							7								
DCOMP							7								
DDRA1												12			
DDPB1												12			
DCRMC1												12			
DDRMMA												12			
DDRMMP												12			
DDRMMS												12			
DDRMHX												12			
DDRMM1												12			
DDRMM2												12			
DDRMM												12			
DDR1A												12			
DDR1B												12			
DDR1X												12			
DDR1												12			
DDR2												12			
DDR								8							
DDUM1													13		
DDUM2													13		
DDUM3													13		
DDUM4													13		
DDUM5													13		
DDUM6													13		
DDUM7													13		
DDUM8													13		
DDUM9													13		
DFCONE								8					13		
DFCOMP				4	5	6	7			10	11				15
DFCPSX							7								
DFCP1X							7								
DFCP2X							7								
DFCP3X							7								
DEFILE	1														
DELETE															
DFLKLS								8							15
DELSET		2	3		5			8					13		
DFLTKL			3					8							

Figure 21. Alphabetical Deck Name - Link Table (Continued)

5.3-87 (12/29/78)

DESCRP	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DETCX								8							
DETC			3												
DETD						6									
DETD						6									
DETFBS						6									
DETM						6									
DETM1						6									
DETM3						6									
DETM4						6									
DETM5						6									
DETM						6									
DEFS1X							7								
DEFS2X							7								
DEFS							7								
DIAGNN															15
DIAGXX															15
DIHEX													13		
DISPLA		2													
DKINT			3					8							
DKI			3					8							
DKLS								8							
DKL			3					8							
DK100			3					8							
DK211			3					8							
DK89			3					8							
DLAMBY									9						
DLANG									9						
DLAXX									9						
DLBDY									9						
DLBPT2									9						
DLBXX									9						
DLCON									9						
DLN									9						
DLDDP				4	5	6	7			10	11				
DLPT2									9						
DLP2X									9						
DMATRX			3					8							
DMFGR		2													
DMINT			3												
DMI			3												
DMPFIL									9	10					15
DMPY							7								
DMPYAD							7								
DMPYX							7								
DM100			3												
DM211			3												
DM89			3												

Figure 21. Alphabetical Deck Name - Link Table (Continued)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DDWN						6									
DDAAA						6									
DDDCOM						6									
DDDCOR						6									
DD01						6									
DD02						6									
DD03						6									
DD04						6									
DD05						6									
DD0						6									
DPLOT		2													
DPLTST		2													
DPPS									9						
DPPSR									9						
DPZY									9						
DDMEM													13		
DDUAD													13		
DRAW		2													
DRDD													13		
DRWCHR		2													
DRWDAT		2													
DSCHKX							7								
DSCHK							7								
DSHFAR													13		
DSMG1													13		
DSMG2X				4											
DSMG2				4											
DSTRDY															15
DS1AXX													13		
DS1A													13		
DS1AAA													13		
DS1ADP													13		
DS1AET													13		
DS1B													13		
DS1ETD													13		
DS1ETT													13		
DS1X													13		
DS1													13		
DTBF													13		
DTPANP							7								
DTRANX				4	5	6	7		9						
DTPBSC													13		
DTRIA													13		
DTRMEM													13		
DTSHLD													13		
DTSHLS													13		
DUMFRG							7								
DUMDD1							7								

Figure 21. Alphabetical Deck Name - Link Table (Continued)

DUM002							7								
DUM003							7								
DUM004							7								
DUMPER	1														
DUMPRM														15	
DUM1XX							7								
DUM1					5										
DUM2XX							7								
DUM2					5										
DUM3XX							7								
DUM3					5										
DUM4XX							7								
DUM4					5										
DUM5					5										
DUM6					5										
DUM7					5										
DUM8					5										
DUM9					5										
DUPART							7								
DVECTR		2													
DYPZ									9						
DZPY									9						
DZYMAT									9						
DZY									9						
EADD						6									
FCTLOC		2						8							
EDIT														15	
EDTL					5										
EGNVCT											11				
FJDUM2	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
EJECT		2						8	9			12	13		
EKTRBD								8							
EKTRBS								8							
EKTRMD								8							
EKTRMS								8							
ELELRL		2													
ELIMX				4											
ELIM				4											
EMADTQ								8							
EMASTQ								8							
EMAXXX								8							
FMA1D								8							
FMA1S								8							
FMA1XX								8							
FMA1								8							
FMA								8							
FMGCNG								8							
FMGCOR								8							

Figure 21. Alphabetical Deck Name - Link Table (Continued)

5.3-91 (12/29/78)

FMGDIC
FMGFST
FMGFIL
FMGFIN
FMGOLD
FMGOUT
FMGPPM
FMGPPN
FMGSNO
FMGTA8
FMGTRX
FMGXXX
FMGX01
FMGX02
FMGX03
FMGX04
FMGX05
FMGX06
FMGX07
FMGX08
FMGX09
FMGX10
FMGX11
FMGX12
FMGX13
FMGX14
FMGX17
FMGX18
FMGX19
FMGX20
FMGX21
FMGX22
FMGX23
FMGX24
FMGX25
FMGX26
FMGX27
FMGX28
FMGX29
FMGY30
FMGX31
FMGY32
FMGX33
FMGX34
FMGX35
FMGX36
FMGX37
FMGY38

FMGX39								8								
FMGX40								8								
FMGX41								8								
FMGX42								8								
FMGX43								8								
FMGX44								8								
FMGX45								8								
FMGX46								8								
FMGX47								8								
FMGX48								8								
FMG1RX								8								
FMG1B								8								
FMG								8								
EMPCOR						6										
EMSG									9							
EMTRBD								8								
EMTPBS								8								
FNCPDE																15
ENDSSS	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
ENDSYS	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
FOMCKM												12				
FOMCKS												12				
FOMCK												12				
FOMKS												12				
EONUT1																15
FOSCOD																15
EOSNF																15
EOSNUT																15
FPRMKN																15
FSFA	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
FSORT	1															
ESTOUT					5											
FXFORT																15
EXIN1X																15
EXIN1																15
EXIN2F																15
FXIN2P																15
FXIN2X																15
EXIN2																15
EXIN																15
FXI2																15
FXLVL																15
FXO2																15
EXTFRN					5											
FACTOR				4	5	6					11					
FACTRU											11					
FACTRX				4	5	6					11					
FAIKXX											11					

Figure 21. Alphabetical Deck Name - Link Table (Continued)

FA1K								11			
FA1KE								11			
FA1PKA								11			
FA1PKC								11			
FA1PKE								11			
FA1PKI								11			
FA1PKV								11			
FA1PKY								11			
FA1XX								11			
FA1								11			
FA2X								11			
FA2								11			
FBSINT								11			
FBSI	4	5		7		10	11	12			
FBSX	4	5		7			11	12		15	
FBS1	4	5		7		10	11	12		15	
FBS21							11				
FBS2										15	
FBS3	4	5		7		10	11	12		15	
FBS4	4	5		7			11	12		15	
FBS	4	5		7		10	11	12		15	
FCNTL					6						
FCTRX							11				
FCURL		5									
FDIT										15	
FDNAME										15	
FDSUR										15	
FDVECT					6						
FEFPAA							11				
FEERCX					6						
FEERXC							11				
FEFRXX					6						
FEERX					6						
FEERZC							11				
FEERZ1							11				
FEERZ2							11				
FEERZ3							11				
FEERZ4							11				
FEER1X					6						
FEER1					6						
FEER2X					6						
FEER2					6						
FEER3X					6						
FEER3					6						
FEER4X					6						
FEER4					6						
FF100		5									
FILCOR					6						

Figure 21. Alphabetical Deck Name - Link Table (Continued)

FILSWI				4	5	6	7		9	10	11	12			15
FIND		2													
FINDC				4	5	6	7		9	10	11				15
FINDER															15
FLLO									9						
FMDI															15
FNAME	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FNDGRD															15
FNDLVL															15
FNDNXL															15
FNDPAR		2	3	4	5	6	7	8	9	10	11	12	13	14	15
FNDPLT	1	2													
FNDPNY					5										
FNDSTF		2													
FNXTVC						6									
FNXTV						6									
FNXT														14	15
FORFIL		2													
FORMAT								8							
FORMGG				4											
FORMG2				4											
FORM12															
FORM1											11				
FORM22											11				
FORM2											11				
FPONT					5										
FPT					5										
FORWV						6									
FORW						6									
FRBK2						6									
FRBK						6									
FRD2AX										10					
FRD2A										10					
FRD2BX										10					
FRD2B										10					
FRD2CX										10					
FRD2C										10					
FRD2DX										10					
FRD2D										10					
FRD2FX										10					
FRD2E										10					
FRD2I										10					
FREAD	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FPLG										10					
FRMAX						6									
FRMLTA						6									
FRMLTD						6									
FRMLTX						6									

Figure 21. Alphabetical Deck Name - Link Table (Continued)

FRMLT					6										
FPRDA1										10					
FRRDR1										10					
FPRDC1										10					
FRRDC2										10					
FRRDC3										10					
FRRDD1										10					
FRRDD2										10					
FRRDF1										10					
FRRDST										10					
FRRD1A										10					
FRRD1B										10					
FRRD1C										10					
FRRD1D										10					
FRRD1F										10					
FRRD2X										10					
FRRD2										10					
FRRD										10					
FRR1A1										10					
FRSW2						6									
FRSW						6									
FWMW									9						
F7Y2									9						
F6211					5								13		
F89					5								13		
GENDSB									9						
GFND									9						
GFNELX				4											
GENFLY				4											
GENVFC				4	5	6	7		9	10	11			15	
GETALK														15	
GETDEF		2													
GFBSX				4	5		7		9	10	11			15	
GFBS				4	5		7		9	10	11			15	
GICOM									9						
GIGGKS									9						
GIGGX									9						
GIGTKA									9						
GIGTKG									9						
GINOX	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
GIPSST									9						
GIPSY									9						
GIVN						6									
GI									9						
GKADA1										10					
GKAD1A										10					
GKAD1C										10					
GKAD										10					

Figure 21. Alphabetical Deck Name - Link Table (Continued)

GKAM1A										10					
GKAM1B										10					
GKAM1X										10					
GKAM										10					
GMMATC								9		10					
GMMATD		2	3	4	5			8		10			13		
GMMATS		2	3	4	5			8	9	10	11	12	13		15
GMMERG															15
GMPRTN															15
GNFIAT	1														
GNFIST		2	3	4	5	6	7	8	9	10	11	12	13	14	15
GOPEN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
GN									9						
GPA1		2													
GPA2		2													
GPCYCX							7								
GPCYC							7								
GPFDRX													13		
GPFDR													13		
GPSP				4											
GPSPA				4											
GPSPB				4											
GPSPC				4											
GPSPD				4											
GPTA1		2	3		5			8				12	13		
GPTLBL		2													
GPTSYM		2													
GPWGA1				4								12			
GPWGB1				4											
GPWG1A				4											
GPWG1B				4								12			
GPWG1C				4											
GPWG				4											
GP1		2													
GP2		2													
GP3A		2													
GP3B		2													
GP3COM		2													
GP3CNR		2													
GP3C		2													
GP3D		2													
GP3		2													
GP4CNR				4											
GP4FIL				4											
GP4PRM				4											
GP4PRT				4											
GP4				4											
GPAV					5										

Figure 21. Alphabetical Deck Name - Link Table (Continued)

GRAVL1
GRAVL2
GRAVL3
GRIDIP
GTMAT1
GUSTY
GUST1
GUST2X
GUST2
GUST3
GUST
HBDY
HEAD
HESS1X
HESS1
HESS2X
HESS2
HMAT
HMATDD
HMTOUT
HPRNG
HSBG
IAPD
IDF1
IDF2
IDPLOT
IFPDC0
IFPD0A
IFPMLT
IFPXX
IFPX0
IFPX1
IFPX2
IFPX3
IFPX4
IFPX5
IFPX6
IFPX7
IFP1A
IFP1B
IFP1C
IFP1D
IFP1E
IFP1F
IFP1G
IFP1XY
IFP1X
IFP1

IFP377	1																		
IFP3	1																		
IFP3RD	1																		
IFP3B	1																		
IFP3LV	1																		
IFP47Z	1																		
IFP4	1																		
IFP4A	1																		
IFP4R	1																		
IFP4C	1																		
IFP4E	1																		
IFP4F	1																		
IFP4G	1																		
IFP527	1																		
IFP5	1																		
IFP5A	1																		
IFP	1																		
IFS1P	1																		
IFS2P	1																		
IFS3P	1																		
IFS4P	1																		
IFS5P	1																		
IFTE2										10									
IFTE4										10									
IFTG										10									
IFTX										10									
IFT										10									
IHEXSD			3		5			8								13			
IHEXSS																13			
IHEX					5														
IHOCSABS													11						
IHOCSAS									9				11						
IHOCSST													11						
IHOFCXPI													11						
IHOFOXPD			3					8					11					15	
IHOFOXPI	1	2	3		5	6		8								13			
IHOFIXPI	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15				
IHOFRXPI		2	3		5	6		8	9		11		13	14					
IHOFRXPR		2	3	4	5	6	7	8	9	10	11	12	13	14	15				
IHOLATN2							7						11						
IHOLEXP			3					8					11					15	
IHOLLOG			3		5	6		8					11		13			15	
IHOLSCN			3				7	8											
IHOLSORT		2	3	4	5	6	7	8	9	10	11		13				15		
IHOSASCN								8					13						
IHOSATN2		2	3	4	5	6	7	8	9	10	11	12	13	14	15				
IHOSEXP		2	3	4	5	6	7	8	9	10	11	12	13	14	15				
IHOSLOG	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15				

Figure 21. Alphabetical Deck Name - Link Table (Continued)

THOSSCNH											11				
IHOSSCN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
IHOSSQRT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
IHOSTNCT		2	3	4	5	6	7	8	9	10	11	12	13	14	15
INCORE										10	11				
INCRO									9						
INFBSX						6					11				
INITL2											11				
INITL											11				
INITX											11				
INPTT1		2													
INPTT2		2													
INPTT3		2													
INPTT4		2													
INPUTA		2													
INPUTX		2													
INPUT		2													
INP1XX		2													
INP2XX		2													
INTFBS											11				
INTLST		2													
INTPRT								8							
INTVFC		2													
INT2AL													13		
INVERD		2	3	4	5			8					13		
INVERP				4	5			8	9	10	11		13		15
INVERT						6									
INVFB											11				
INVPWR						6									
INVPWX						6									
INVPXX						6									
INVPX						6									
INVP1X						6									
INVP1						6									
INVP2V						6									
INVP2X						6									
INVP2						6									
INVP3X						6									
INVP4X						6									
INVRTX						6									
INVTRX						6									
INVTR						6									
ISFT	1														
ITCDDF															15
ITFMDT															15
ITMPRT															15
ITTYPE															15
IUNION		2													

Figure 21. Alphabetical Deck Name - Link Table (Continued)

KRAAR			3													
KCONFEX			3					8								
KCONFY			3					8								
KCONFZ			3					8								
KCONF			3					8								
KDS									9							
KDUM1			3					8								
KDUM2			3					8								
KDUM3			3					8								
KDUM4			3					8								
KDUM5			3					8								
KDUM6			3					8								
KDUM7			3					8								
KDUM8			3					8								
KDUM9			3					8								
KELAS			3					8								
KFLUD2			3					8								
KFLUD3			3					8								
KFLUD4			3					8								
KHRDY			3					8								
KIHFX			3													
KORSZ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
KPANEL			3					8								
KPLTST			3					8								
KODMEM			3					8								
KODMM1			3													
KODM2D			3													
KODPLT			3					8								
KRND			3													
KSLNT			3					8								
KSNLID			3					8								
KTETRA			3					8								
KTORDR			3					8								
KTPZ			3													
KTRAPR			3					8								
KTRBSC			3					8								
KTRIA			3													
KTRIQD			3					8								
KTRIRG			3					8								
KTRMEM			3					8								
KTRM6D								8								
KTRM6S								8								
KTRPLD								8								
KTRPLS								8								
KTRPLT			3					8								
KTSHLD								8								
KTSHLS								8								
KTUBE			3													

Figure 21. Alphabetical Deck Name - Link Table (Continued)

LAMXXX									9							
LAMX									9							
LD01	1															
LD02	1															
LD03	1															
LD04	1															
LD05	1															
LD06	1															
LD07	1															
LD08	1															
LD09	1															
LD10	1															
LD11	1															
LD12	1															
LD13	1															
LD14	1															
LD15	1															
LD21	1															
LD22	1															
LD23	1															
LD45	1															
LD46	1															
LINE1		2														
LINE10		2														
LINE3		2														
LINE4		2														
LINE		2														
LINKNSXX	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
LOADS					5											
LOADX					5											
LOADAPP															15	
LOADAPX															15	
ISPLIN									9	10	11					
MABC					5								13			
MAR			3		5			8					13			
MAGPHA												12	13			
MAKMCB	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
MAPFNS	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
MAPSET		2														
MASSD			3					8								
MASSTO			3					8								
MATR					5								13			
MATDUM								8								
MATGEN							7									
MATGNX							7									
MATGPR								8								
MATIN		2	3		5			8				12	13			
MATOUT		2	3		5			8				12	13			

Figure 21. Alphabetical Deck Name - Link Table (Continued)

MATPRG				10	
MATPRN			8		
MATPPT			8		
MATVC2				11	
MATVEC				11	
MATWRT					15
MA1XX			8		
MBAMGX				9	
MBAMG				9	
MBAR	3				
MRRSLJ				9	
MBCAP				9	
MBCTR1				9	
MBCTR2				9	
MRDPDH				9	
MRGAE				9	
MRGATE				9	
MRGAW				9	
MRGEOD				9	
MRMNDE				9	
MRQXA				9	
MRQXC				9	
MRPLOT				9	
MRPRIT				9	
MRREG				9	
MCBAR	3				
MCEA1		4			
MCEB1		4			
MCEC1		4			
MCED1		4			
MCE1		4			
MCE1A		4			
MCE1B		4			
MCE1C		4			
MCE1D		4			
MCE2		4			
MCONE	3		8		
MCNNMX	3		8		
MCRPD	3				
MDUM1	3				
MDUM2	3				
MDUM3	3				
MDUM4	3				
MDUM5	3				
MDUM6	3				
MDUM7	3				
MDUM8	3				
MDUM9	3				

Figure 21. Alphabetical Deck Name - Link Table (Continued)

MERGED								9		11					
MERGE1						7									
MERGE					5	6	7	9	10	11	12			15	
MESSAGE	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MFLUD2			3					8							
MFLUD3			3					8							
MFLUD4			3					8							
MFREE			3					8							
MHBDY			3					8							
MIHEX			3												
MINMAX		2													
MINTRP										10	11				
MINV			3		5			8					13		
MODA							7								
MODACC												12			
MODACX												12			
MODAC1												12			
MODAC2												12			
MODAC3												12			
MODB							7								
MODC							7								
MODDMP	1														
MODEL	1														
MPLPRT	1														
MPRTX								8							
MPYADX				4	5	6	7			10	11	12			15
MPYADZ				4	5	6	7		9	10	11	12			15
MPYAD				4	5	6	7		9	10	11	12			15
MPYA1D							7								
MPYA2D							7								
MPYA3D							7								
MPYL					5										
MPYQ				4	5	6	7		9	10	11	12			15
MPY3A							7								15
MPY3B							7								15
MPY3C							7								15
MPY3DR							7								15
MPY3NU							7								15
MPY3OC							7								15
MPY3P							7								15
MPY3TL							7								15
MPY3ZZ							7								
MPY3							7								
MPY3CP							7								15
MPY3IC							7								15
MPZDA			3												
MODPLT			3					8							
MPENDZ															15

Figure 21. Alphabetical Deck Name - Link Table (Continued)

MRED1A															15
MRED1B															15
MRED1C															15
MRED1D															15
MRED1E															15
MRED1															15
MRED2A															15
MRED2B															15
MRED2C															15
MRED2D															15
MRED2E															15
MRED2F															15
MRED2G															15
MRED2H															15
MRED2I															15
MRED2J															15
MRED2L															15
MRED2M															15
MRED2N															15
MRED2O															15
MRED2P															15
MRED27															15
MPED2															15
MRGF								8							
MRGEDX									9						
MRING			3					8			11				
MROD			3												
MSB															
MSGCOM	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MSGWRT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MSGX	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MSLOT			3					8							
MSOLID			3					8							
MSTRIA			3												
MTIMSU						6									
MTMSU1						6									
MTORDR			3					8							
MTRAPR			3					8							
MTRBSC			3					8							
MTRIQQ			3					8							
MTRIRG			3					8							
MTPPLT			3					8							
MTRXIN										10					
MTRXIX															15
MTRXI															15
MTRXDX															15
MTRXD															15
MTRXXX										10					

Figure 21. Alphabetical Deck Name - Link Table (Continued)

MTURE			3												
MXCIDS							7								
MXCID							7					12			
NAMES	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
NASCAR	1														
NORM11						6									
NORM1						6									
NTIMEX	1														
NTIME	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
NCTRCD											11				
NDUM1														14	
NDUM2														14	
NDUM3														14	
NDUM4														14	
NDUM5														14	
NDUM6														14	
NDUM7														14	
NDUM8														14	
NDUM9														14	
NDUM														14	
NFPR9														14	
NFPRD1														14	
NFPRD5														14	
NFPR1														14	
NFPR2														14	
NFPR3S														14	
NFPR3														14	
NFPR4														14	
NFPR5														14	
NFPR6														14	
NFPR7S														14	
NFPR7														14	
NFPR8														14	
NFPCF1														14	
NFPCF2														14	
NFPCNM														14	
NFPCS1														14	
NFPCS2														14	
NFPMIS														14	
NFPPNT														14	
NFPPUN														14	
NFPRF1														14	
NFPRF2														14	
NFPRS1														14	
NFPRS2														14	
NFPSN1														14	
NFPSS1														14	
NFPXXX														14	

Figure 21. Alphabetical Deck Name - Link Table (Continued)

QFP1																14
QFP1A																14
QFP																14
QFRF2S																14
QFRS2S																14
QFSN1																14
QFSS1																14
QNETW0				4	5	6	7			10	11					
QPINV						6										
QPMESG		2														14
QPTPR1		2														
QPTPR2								8								
QPTPW1		2														
QPTPW2								8								
QPTPX1		2														
QPTPX		2														
QPTP1A		2														
QPTP1R		2														
QPTP1C		2														
QPTP1D		2														
QPT2A								8								
QPT2B								8								
QPT2C								8								
QPT2D								8								
ORDER		2														
ORTCK						6										
ORTHO											11					
OSCENT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
OSCXRF	1															
OUTPT1														14		
OUTPT2														14		
OUTPT3														14		
OUTPT4														14		
OUTPT														14		
OUTPUT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
OUT1XX														14		
OUT2XX														14		
OUT3XX														14		
PABS											11					
PACKX	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
PAGE	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
PARAML		2	3	4	5	6	7	8	9	10	11	12	13	14	15	
PARAM		2														
PARMEG				4	5	6	7		9	10	11	12			15	
PARMLX		2	3	4	5	6	7	8	9	10	11	12	13	14	15	
PARTN1							7									
PARTN2							7									
PARTN3							7									

Figure 21. Alphabetical Deck Name - Link Table (Continued)

PARTN				4	5		7		9	10	11	12			15
PASSER	1														
PATX				4	5	6	7		9	10	11	12			15
PERMUT					5										
PERPEC		2													
PEXIT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PHDMIA														14	
PHDMIX														14	
PKRAR														14	
PKQAD1													13		
PKQAD2													13		
PKQDMS													13		
PKQDM1													13		
PKQDM													13		
PKQDPL													13		
PKRDD													13		
PKT01													13		
PKT02													13		
PKTRBS													13		
PKTRI1													13		
PKTRI2													13		
PKTRMS													13		
PKTPM1													13		
PKTRM													13		
PKTRPL													13		
PKTROD													13		
PKTRQ2													13		
PLAGP													13		
PLAMAT													13		
PLA1													13		
PLA2X													13		
PLA2													13		
PLA3ES													13		
PLA3UV													13		
PLA31X													13		
PLA31													13		
PLA32C													13		
PLA32E													13		
PLA32S													13		
PLA32X													13		
PLA32													13		
PLA3													13		
PLA4B													13		
PLA4ES													13		
PLA4UV													13		
PLA41X													13		
PLA41													13		
PLA42C													13		

Figure 21. Alphabetical Deck Name - Link Table (Continued)

PLA42D																		13
PLA42E																		13
PLA42S																		13
PLA42X																		13
PLA42																		13
PLA4																		13
PLOAD3					5													
PLOAD					5													
PLOT		2																
PLTDAT		2																
PLTMGX																		15
PLTMRG																		15
PLTOPR		2																
PLTRN1								8										
PLTSCP		2																
PLTSET		2																
PLTTRA								8										
PREFIX																		15
PRELOC	1	2	3	4	5	6	7	8	9	10	11	12	13	14				15
PREMAT		2	3		5			8				12	13					
PRESAX					5													
PRETAB			3		5			8		10	11		13	14				15
PRETRD		2	3	4				8					13					
PRETRS			3	4	5			8	9			12	13					15
PRINT		2																
PROCES		2																
PRTINT								8										
PRTMRG							7											
PRTMSG		2																
PRTPRM								8										
PSRAP													13					
PSQAD1													13					
PSQAD2													13					
PSQDM1													13					
PSODM													13					
PSOPL1													13					
PSRND													13					
PSTA									9									
PSTAMG									9									
PSTONC									9									
PSTONX									9									
PSTPL1													13					
PSTO1													13					
PSTO2													13					
PSTR01													13					
PSTRI1													13					
PSTRI2													13					
PSTRM1													13					

Figure 21. Alphabetical Deck Name - Link Table (Continued)

PSTRM													13		
PSTRO2													13		
PTMGZZ							7								
PULL	1														
PUSH	1														
PVECA														15	
PVECA									9						
PVECKX									9						
PVECK									9						
PVECOM									9						
PVECPX									9						
PVECP									9						
PVECSX									9						
PVECS									9						
PVECO5									9						
PVEC10									9						
PVEC20									9						
PVEC									9						
PXIT36	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
QDMEM					5										
QDMM10								8							
QDMM1S								8							
QDMM1					5										
QDMM20								8							
QDMM2S								8							
QDMM2					5										
QDPLT					5										
QHBDY					5										
QLOADL					5										
QPARAM		2	3	4	5	6	7	8	9	10	11	12	13	14	15
QPARAMR		2	3	4	5	6	7	8	9	10	11	12	13	14	15
QRITER						6									
QVECT					5										
QVOL					5										
Q2BCD			3												
Q2BCS					5										
Q2TRMD			3										13		
Q2TRMS					5								13		
RANDMX														14	
RANDNM														14	
RAND1														14	
RAND2														14	
RAND3														14	
RAND5														14	
RAND6														14	
RAND7														14	
RAND8														14	
PBMG1				4										14	

Figure 21. Alphabetical Deck Name - Link Table (Continued)

NASTRAN - OPERATING SYSTEM INTERFACES

RBMG2		4		
PPMG3		4		
RBMG4		4		
RCARD	1			
RCNVAX				15
RCOVA				15
RCNVB				15
RCOVCM				15
RCOVCR				15
RCOVCS				15
RCNVC				15
RCOVDS				15
RCOVEM				15
RCOVEX				15
RCOVE				15
RCOVIM				15
RCOVLS				15
RCOVMS				15
RCOVOX				15
RCOVO				15
RCNVQV				15
RCNVR3				15
RCNVR				15
RCOVSL				15
RCNVSS				15
RCNVUI				15
RCOVUO				15
RCOVVA				15
RCOV3X				15
RDMODX	2			
READ1A		6		
READ1		6		
READ2A		6		
READ2		6		
READ3		6		
READ4		6		
READ6X		6		
READ6		6		
READ7		6		
REDU	1			
RFDUCE				15
RED777				15
REGFAN		6		
RFIC		6		
RFIGA		6		
RFIGKR		6		
RENAME				15
RETRLK				15

Figure 21. Alphabetical Deck Name - Link Table (Continued)

RETURN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RE2AL								8			11				
RFORCE					5										
RMGZZZ					5										
RMG					5										
RODD								8							
RDDS								8							
RDD					5										
ROMADK			3					8							
ROMBER					5								13		
ROTAT		2													
ROTAX						6									
ROWDYZ									9						
RSORT											11				
RSTXXX		2													
RULER				4	5	6	7		9	10	11	12			15
SADDX				4		6	7		9	10					15
SADD				4		6	7		9	10	11				15
SADOTB			3		5			8	9				13		
SAXB			3		5			8	9				13		
SAXIF1													13		
SAXIF2													13		
SBAR1													13		
SBAR2													13		
SBSPL2													13		
SCALAR															15
SCALED								8							
SCALFX				4											15
SCALXX				4											
SCE1															
SCONE1													13		
SCONE2													13		
SCONE3													13		
SCRLM													13		
SDCINS							7								
SDCIN				4	5	6	7			10	11				15
SDCMMX							7								
SDCMM							7								
SDCMPS							7								
SDCMO							7								
SDCOMP				4	5	6	7			10	11				15
SDCOMX				4	5	6	7			10	11				15
SDCOM1				4	5	6	7				11				15
SDCOM2										10					15
SDCOM3				4	5	6	7			10	11				15
SDCOM4				4	5	6	7			10	11				15
SDCOM				4	5	6	7			10	11				15
SDCOUT				4	5	6	7			10	11				15

Figure 21. Alphabetical Deck Name - Link Table (Continued)

SDHTFF					13	
SDHTF1					13	
SDHTF2					13	
SDRA1					12	
SDRA2					13	
SDR81	5	6	7		12	
SDRCHK					13	
SDRC2					13	
SDRETD					13	
SDRETT					13	
SDRHTZ					13	
SDRHT					13	
SDR1					12	
SDR1A					12	
SDR1R	5	6	7		12	
SDR2AA					13	
SDR2A					13	
SDR2R					13	
SDR2C					13	
SDR2DE					13	
SDR2D					13	
SDR2E					13	
SDR2XX					13	
SDR2X1					13	
SDR2X2					13	
SDR2X4					13	
SDR2X5					13	
SDR2X6					13	
SDR2X7					13	
SDR2X8					13	
SDR2X9					13	
SDR2					13	
SDR3ZZ						14
SDR3						14
SDR3A						14
SDUM11					13	
SDUM12					13	
SDUM21					13	
SDUM22					13	
SDUM31					13	
SDUM32					13	
SDUM41					13	
SDUM42					13	
SDUM51					13	
SDUM52					13	
SDUM61					13	
SDUM62					13	
SDUM71					13	

Figure 21. Alphabetical Deck Name - Link Table (Continued)

SDUM72																	13
SDUM81																	13
SDUM82																	13
SDUM91																	13
SDUM92																	13
SD2RHD																	13
SEEMAT		2															13
SEEMTX		2															
SELAS1																	13
SELAS2																	13
SFLCAM		2															
SEMINT	1																
SEMTRN	1																
SEM	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		
SETEQ																	15
SETFND												12	13				15
SETINP		2															15
SETLVL																	15
SETTIM	1																
SETUP	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		
SETVAL		2	3	4	5	6	7	8	9	10	11	12	13	14	15		
SFACT				4	5	6	7	8	9	10	11	12	13	14	15		
SFATCH											11				15		
SGENZZ															15		
SGEN															15		
SGENA															15		
SGENB															15		
SGENM															15		
SGINN1		2													15		
SGINO		2															
SHAPE		2															
SHEARD								8									
SHEARS								8									
SICOX						6											
SIHFX1																	
SIHFX2													13				
SJUMP													13				
SKPFRM		2															15
SKPREC	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		
SMAR3				4													
SMAC3				4													
SMA1X			3														
SMA1			3														
SMA1A			3														
SMA1BK			3					8									
SMA1B			3														
SMA1CL			3					8									
SMA1DP			3					8									

Figure 21. Alphabetical Deck Name - Link Table (Continued)

SDUM72																13
SDUM81																13
SDUM82																13
SDUM91																13
SDUM92																13
SD2RHD																13
SEEMAT		2														
SEEMTX		2														
SELAS1																13
SELAS2																13
SELCAM		2														
SEMINT	1															
SEMTRN	1															
SEM'	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
SETEQ																15
SETFND												12	13			15
SETINP		2														
SETLVL																15
SETTIM	1															
SETUP	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
SETVAL		2	3	4	5	6	7	8	9	10	11	12	13	14	15	
SFACT				4	5	6	7				11					15
SFETCH																15
SGENZZ																15
SGEN																15
SGENA																15
SGENB																15
SGENM																15
SGINQ1		2														
SGINO		2														
SHAPE		2														
SHEARD								8								
SHEARS								8								
SICOX						6										
SIHEX1													13			
SIHEX2													13			
SJUMP																15
SKPFRM		2														
SKPREC	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
SMAB3				4												
SMAC3				4												
SMA1X			3													
SMA1			3													
SMA1A			3													
SMA1BK			3					8								
SMA1B			3													
SMA1CL			3					8								
SMA1DP			2					8								

Figure 21. Alphabetical Deck Name - Link Table (Continued)

SMA1ET			3					8									
SMA1HT			3					8									
SMA1IO			3					8									
SMA2X			3														
SMA2			3														
SMA2A			3														
SMA2BK			3					8									
SMA2B			3														
SMA2CL			3					8									
SMA2DP			3					8									
SMA2ET			3					8									
SMA2HT			3					8									
SMA2IO			3					8									
SMA3				4													
SMA3A				4													
SMA3B				4													
SMA3C				4													
SMLFIG						6											
SMMATS												13					
SMPYAD							7										
SMP1				4													
SMP2				4													
SMSG																15	
SNPDF								9									
SNFCLS																15	
SNFCOM	1	2	3	4	5	6	7	8	9	10	11	12	13	14		15	
SNFINT																15	
SNFIOF																15	
SNFIOI	1															15	
SNFIO																15	
SNFI																15	
SNFOPN																15	
SNFO																15	
SNFSIZ																15	
SNFTOC																15	
SNFTRL																15	
SNFUTX																15	
SNFUT																15	
SNF																15	
SNLID					5												
SNLVER				4													
SOLVE1													13				
SOLVE							7										
SNLVPX				4													
SNLV1X							7										
SNLV2X							7										
SNLV3X							7										
SNLV4X							7										

Figure 21. Alphabetical Deck Name - Link Table (Continued)

SOLV5X							7								
SNRT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SNRTCM															15
SNUT													13		
SPANL1													13		
SPANL2													13		
SPLT10													13		
SODME1															15
SODM11													13		
SODM12													13		
SODM21													13		
SODM22													13		
SODPL1													13		
SNRTM															
SRDD1						6							13		
SRDD2													13		
SSGA1X					5										
SSGA2					5				9	10	11	12			
SSGA3				4	5							12			
SSGB1X					5										
SSGB2				4	5	6	7		9	10	11	12			15
SSGC2				4					9	10	11				15
SSGETD					5										
SSGETT					5										
SSGHTP					5										
SSGHTZ					5										
SSGHT1					5										
SSGHT2					5										
SSGHT					5										
SSGKHI					5										
SSGSLT					5										
SSGTRI					5										
SSGWRK					5										
SSG1					5										
SSG1A					5										
SSG2A					5				9	10	11	12			
SSG2X					5										
SSG2					5										
SSG2B				4	5	6	7		9	10	11	12			15
SSG2C				4					9	10	11				15
SSG3					5										
SSG3A				4	5							12			
SSG4					5										
SSLDT1													13		
SSLDT2													13		
SSOLD1													13		
SSOLD2													13		
SSPLIN									9		11		13		

Figure 21. Alphabetical Deck Name - Link Table (Continued)

SSWTCH	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
STAPID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
STEP2											11				
STEP											11				
STIME	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
STORD1													13		
STORD2													13		
STPAIC									9						
STPAX1													13		
STPAX2													13		
STPAX3													13		
STPB6									9						
STPBSO									9						
STPBS1									9						
STPDA									9						
STPK									9						
STPLOT		2													
STPPHI									9						
STPPT2									9						
STOME2													13		
STRAP1													13		
STRAP2													13		
STRAX1													13		
STRAX2													13		
STRAX3													13		
STRBS1													13		
STRIPC									9						
STRIPX									9						
STRIR1													13		
STRIR2													13		
STRME1													13		
STRM61													13		
STRM62													13		
STRPL1													13		
STRPTS													13		
STRP11													13		
STRP12													13		
STROD1													13		
STROD2													13		
STRSLV													13		
STRSL1													13		
STRSL2													13		
STURE1													13		
SUBR									9						
SUBI									9						
SUBPZ															
SUBP									9						15
SUBPB									9						

Figure 21. Alphabetical Deck Name - Link Table (Continued)

SUBPH1																		15
SUB1						6												
SUB						6												
SUMM						6												
SUMPHI									9									
SUPLT		2																
SUREAD																		15
SUWRT																		15
SWITCH							7											
SWSRT	1																	
SYMRLS		2																
SYMROL		2																
SYSTEM	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15			
SYS																		15
TAA1		2																
TAA2		2																
TABFMT								8										
TABFTX								8										
TABFTZ								8										
TABPCH									9									
TABPHX									9									
TABPRE									9									
TABPRT								8						14	15			
TABPRX								8						14	15			
TABPT								8										
TAC1		2																
TAC1AX		2																
TAPBIT	1	2												14	15			
TAPSWI		2												14				
TA1A		2																
TA1AB		2																
TA1ACH		2																
TA1B		2																
TA1C		2																
TA1CA		2																
TA1ETD		2																
TA1FTT		2																
TA1H		2																
TA1		2																
TETRA					5													
TIMAL3									9									
TIMEEQ				4	5	6	7		9	10	11							15
TIMTST									9									
TIMTS1									9									
TIMTS2									9									
TIMTS3									9									
TIMTS4									9									
TIMTS5									9									

Figure 21. Alphabetical Deck Name - Link Table (Continued)

TIMTS6								9						
TIMTS7								9						
TIMTS8								9						
TIM1XX								9						
TIM2XX								9						
TIM3XX								9						
TIM4XX								9						
TIM5XX								9						
TIM6XX								9						
TIM7XX								9						
TIM8XX								9						
TIPE		2												
TKER								9						
TKTZTK			3									13		
TLODM6					5									
TLNDSL					5									
TLODT1					5									
TLODT2					5									
TLODT3					5									
TMTOGO	1	2	3	4	5	6	7	8	9	10	11	12	13	14
TPREPX									9					15
TPSWIT		2												14
TPZTEM					5									
TRAILE									9					
TRANEM													13	
TRANP1				4	5	6			9					
TRANSP				4	5	6	7			10	11			
TRANX					5									
TRAPAD								8						
TRAPAX								8						
TRBSCD								8						
TRBSCS								8						
TRBSC					5									
TRDA1											11			
TRDC1											11			
TRDD1											11			
TRDE1											11			
TRDXX											11			
TRD1A2											11			
TRD1A											11			
TRD1C2											11			
TRD1C											11			
TRD1D2											11			
TRD1D											11			
TRD1E											11			
TRD1X											11			
TRD											11			
TPHTX											11			

Figure 21. Alphabetical Deck Name - Link Table (Continued)

TRHT1A										11									
TRHT1B										11									
TRHT1C										11									
TRHT										11									
TRIAAD									8										
TRIAAX									8										
TRIDI						6													
TRIMEM					5														
TRIMEX					5														
TRIQD					5														
TRLGA					5														
TRLG1C					5														
TRLG					5														
TRLGR					5														
TRLGC					5														
TRLGD					5														
TRMEMD									8										
TRMEMS									8										
TRNSPX				4	5	6	7			9								15	
TRNSP				4	5	6	7			9								15	
TRPLT					5														
TRTTEM					5														
TSPL1D									8										
TSPL1S									8										
TSPL2D									8										
TSPL2S									8										
TSPL3D									8										
TSPL3S									8										
TTLPGE	1																		
TTORDR					5														
TTRAPR					5														
TTRIRG					5														
TUBED									8										
TUBES									8										
TVOR										9									
TWISTD									8										
TWISTS									8										
TWO	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15				
TYPE10		2																	
TYPE3		2																	
TYPE	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15				
TYPFLT		2																	
TYPINT		2																	
TYPOUT		2																	
UMFEDT	1															14			
UMFXXX	1																		
UMFZZZ	1																		
UNPAKX	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15				

Figure 21. Alphabetical Deck Name - Link Table (Continued)

UPARTX				4			7			10					
UPART				4			7			10					
USRMSG	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VALVEC						6									
VDRA												12			
VDRB												12			
VDRCOM												12			
VDRCOR												12			
VDR												12			
VECPRT								8							
VFCXXX							7								
VEC							7								
VTSCD								8							
WALTIM	1														15
WILVEC						6									
WPLT10		2													
WPLT3		2													
WPLT4		2													
WRTMSG		2													
WRTPRT		2													15
WRTTRL	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XCEITB	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XCEI	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XCHK	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XCLEAN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XCSA	1														
XCSABF	1														
XCSTM					5										
XDPH	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XDPL	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XFADJ1	1														
XFIAI	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XFIST	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XFLDEF	1														
XFLORD	1														
XFLSZD	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XGPIBS	1														
XGPIC	1														
XGPIDG	1														
XGPID	1														
XGPI1	1														
XGPI2X	1														
XGPI2	1														
XGPI3	1														
XGPI4	1														
XGPI5	1														
XGPI6	1														
XGPI7	1														

Figure 21. Alphabetical Deck Name - Link Table (Continued)

XGP18	1														
XGPI	1														
XIHEX							8								
XIPFL	1														
XLINK	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XLKSPC	1														
XLNKHD	1														
XMDMSK	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
YNSTRN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XOLDPT	1														
XNSGEN	1														
XPARAM	1														
XPFIIST	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XPOLCK	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XPUNP	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XPURGE	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XRCARD	1														
XRECPS	1														
XRGDFM	1														
XSAVE		2	3	4	5	6	7	8	9	10	11	12	13	14	15
XSBSET	1														
XSCNDM	1														
XSEM01	1														
XSEM02		2													
XSEM03			3												
XSEM04				4											
XSEM05					5										
XSEM06						6									
XSEM07							7								
XSEM08								8							
XSEM09									9						
XSEM10										10					
XSEM11											11				
XSEM12												12			
XSEM13													13		
XSEM14														14	
XSEM15															15
XSFA1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XSFA		2	3	4	5	6	7	8	9	10	11	12	13	14	15
XSORT	1														
XNSGN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XSPTRD	1														
XTRNSY						6									
XTRNY1						6									
XVPS	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XXFIAT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XXMPRT								8							
XXOPT1		2													

Figure 21. Alphabetical Deck Name - Link Table (Continued)

XXOPT2							8								
XXPARM	2														
XXPLOT	2														
XXPMMSG	2														
XXPRTI							8								
XXPSET	2														
XXVLVC					6										
XYCHAR													14		
XYDUMP													14		
XYFIND													14		
XYGRAF													14		
XYLONG													14		
XYOUT													14		
XYPLIN	2														
XYPLOT	2														
XYPLXX	2														
XYPPPP													14		
XYPRPL													14		
XYPRPT													14		
XYTICS													14		
XYTRAN													14		
XYTRZZ													14		
XYWORK													14		
ZAPD								9							
ZBLPKX	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ZFRNC									9	10	11				
ZJ									9						
ZNTPKX	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Figure 21. Alphabetical Deck Name - Link Table (Continued)

5.4 NASTRAN ON THE UNIVAC 1108/1110 (EXEC 8)

5.4.1 Introduction

The purpose of this section is to describe the unique operating features of NASTRAN on the UNIVAC 1108/1110 computers under the Exec 8 operating system. All references to the 1108 apply to the 1110 unless otherwise stated.

NASTRAN operates as a single job on UNIVAC 1108/1110 computers under Exec 8. NASTRAN is created as 15 separate absolute programs, with each program containing the structure and content of a NASTRAN link. These links are executed serially (usually) by the use of 'XQT' cards. It will operate in core of from 64K to 256K decimal words.

5.4.2 Input/Output

The machine dependent I/O routines in NASTRAN can be classified into three categories: 1) obtaining the logical files, 2) performing the actual I/O, and 3) generating the plot tapes.

GNFIAT is the subroutine responsible for identifying the logical files available for use by NASTRAN. For the UNIVAC 1108, it is written in FORTRAN. Its functional description can be found in Section 3. Under Exec 8, GNFIAT operates in conjunction with the system routine NTAB\$. NTAB\$ was reassembled for NASTRAN to allow access to logical units 1-40. Correspondingly, a table is generated in GNFIAT to indicate the status of each of these units. The status code of 1 implies the unit is unconditionally available for insertion into the FIAT Executive Table. A code of 2 means the unit can be used if it is not set up as a tape. A status code of 3 means the unit is unavailable for the FIAT, but is available for XFIAT. A status code of 4 means the unit is not to be assigned by NASTRAN. Any units with status codes of 1, 2, or 3 will be dynamically assigned by NASTRAN unless they have been previously assigned by user-supplied assign cards. GNFIAT searches this table and inserts the available logical units into the FIAT.

NASTRAN - OPERATING SYSTEM INTERFACES

The following table indicates the status of the 40 units:

<u>FØRTRAN Unit No.</u>	<u>Status Code</u>	<u>External Name</u>
1	4	None
2	3	2
3	1	3
4	1	4
5	4	None (Input File)
6	4	None (Print File)
7	3	PØØL
8	3	ØPTP
9	3	NPTP
10	2	UMF
11	2	NUMF
12	3	PLT1
13	3	PLT2
14	2	INPT
15	2	INP1
16	2	INP2
17	2	INP3
18	2	INP4
19	2	INP5
20	2	INP6
21	2	INP7
22	2	INP8
23	2	INP9
24	3	24
25	1	25
26	1	26
27	1	27
28	1	28
29	1	29

NASTRAN ON THE UNIVAC 1108/1110 (EXEC 8)

<u>FØRTRAN Unit No.</u>	<u>Status Code</u>	<u>External Code</u>
30	1	30
31	1	31
32	1	32
33	1	33
34	1	34
35	1	35
36	1	36
37	1	37
38	1	38
39	1	39

FØRTRAN unit numbers 7 through 23 are attached to their external names by the use of @USE cards which are added internally. Note that this requires assignments be made by the external name, rather than the FØRTRAN unit number, as was done in the past. Dynamic assigns for all units not assigned by the user (up to MAXFIL units (System (29))) will be of the following form @ASG,T XX,F17//PØS/30. This assign card allows a maximum of 1,920 tracks, or approximately 3,500,000 words, for each file. The F17 requests mass storage as available from first FH1782's, second FH880's, and third FASTRAND model 11. PØS requests that 64 contiguous tracks be assigned at once. 30 will terminate the run if more than 30 x 64 tracks of data are written on any one file. To change the drum allocation of dynamic assigns from PØS (positions) to TRK (tracks) insert a NASTRAN card before the ID card setting SYSTEM (34)=2. This results in files being assigned in the following form @ASG,T XX,F17//TRK/1920. TRK requests that 64 sectors (28 words/sector) be assigned at one time. This is beneficial for smaller jobs being run on Exec 8 release 32 and earlier. For Exec 8 release 33 and later allocation should be by tracks due to a redesign of mass storage tables within EXEC 8. ØPTP, NUMF and UMF are "FREEed" at the end of the PREFACE if they are assigned to tape.

In addition to the expanded NTAB\$, NTRAN\$ was assembled with the number of packets increased to 37; the stack table increased to 15; and the packet size set to 9. This was necessary due to the number of files used in NASTRAN at any one time.

The actual I/O on the UNIVAC 1108 is controlled through a blocked I/O package, consisting of GINØ and its associated routines. The physical I/O directives are restricted to IØ1108. Physical records of size BUFSIZ-3 (System (1)) words are written on all units. BUFSIZ is currently set to 1794. Under Exec 8, NTRAN is used exclusively by IØ1108 for I/O. The reader should consult the subroutine descriptions for GINØ in Section 3.4.12 and for IØ1108 in section 5.4.7 for more information.

The final I/O dependent routine is SGINØ, which generates the unformatted tapes for the plotters. The restriction in SGINØ is that the output records must be free of any NASTRAN or system control words. NTRAN satisfies these requirements and is used to perform the I/O.

5.4.3 Link Switching

A link on the UNIVAC 1108 consists of a main program, MAINi, that calls subroutine XSEMi, (i = 01, 02...15) which in turn calls the subroutines corresponding to the modules residing in that link. Thus, a link is, in itself, a complete executable program.

The problem of link switching reduces to the problem of selectively controlling the execution of the various programs (links). This is accomplished by NASTRAN's dynamically issuing elements to EXEC 8 that contain @XQT cards for executing a specific NASTRAN link. There are 15 symbolic elements that have the names LINK2, LINK2,...LINK15, and LINK99, respectively, and within each symbolic element there is one @XQT card for the corresponding absolute element for LINK2, LINK3, ...LINK15, and LINK99, respectively.

5.4.4 Overlay Considerations and Implementation of Open Core

The complex nature of the NASTRAN overlay picture and the peculiarities of the Exec 8 loader pose some special problems on the 1108. The loader under Exec 8 is a "block" or "segment" loader. A segment is loaded only when a subroutine within that segment is called. Also, when a segment is loaded, local data and common blocks are set to zero.

The implications of these features of the loader can be seen in the following overlay example given in Table 1 and Figure 1. If common block /XX/ is initialized by the Block Data subprogram E, and subroutine A calls D directly, then D cannot reference the data in /XX/, since that segment has not been loaded. Also, if subroutine A stores data in /YY/, and subsequently calls subroutine C, /YY/ will be reset to zero as it is loaded.

NASTRAN ON THE UNIVAC 1108/1110 (EXEC 8)

As seen in Table 1 and Figure 1, some common blocks will have to be repositioned higher (in lower order core) in the overlay on the UNIVAC 1108 than on the IBM 360 or CDC 6600 to protect NASTRAN from the segment loader.

The implementation of open core on the UNIVAC 1108 consists of defining a common block (/DFCØR/) in subroutine DEFCØR and using the executive required (ER) to MCØRE\$ to reserve core from the first cell of the common block to 177770₈. This will define a 65K NASTRAN system. Larger NASTRAN versions can be defined by changing the length of open core with the use of the HICØRE parameter on the NASTRAN card, e.g., NASTRAN HICØRE=256000 will cause NASTRAN to execute in 256000₁₀ words.

FØRTRAN programmers coding subroutines on the UNIVAC 1108 should be aware of two addressing difficulties if the subroutine references addresses larger than 65K (177777₈). First, it is not possible to send an address greater than 65K to a FØRTRAN subprogram through its calling sequence. Second, the limit to the address portion of an instruction is 65K. However, an array which begins at an address less than 65K and extends beyond 65K is usable because FØRTRAN places the starting address of the array in the address portion of an instruction and the subscript in an index register.

For NASTRAN to run with more than 65K of core, care must be taken that arguments involving parameters in open core which may have addresses greater than 65K are passed as arrays and not as a non-dimensioned variable. To pass a non-dimensioned variable, that may have an address greater than 65K, involves passing the starting address of open core (e.g., Z(1)) as a dimensioned variable in a subroutine argument list and an index to the open core array where the variable is stored as a second argument.

NASTRAN - OPERATING SYSTEM INTERFACES

Table 1. Example of Input to the MAP Processor for a NASTRAN Overlay on the UNIVAC 1108.

Sample

```

@MAP,I
      SEG  AA*
      IN   A
      SEG  BB*
      IN   B,XX,E
      SEG  CC*
      IN   C,YY
      SEG  DD*,CC
      IN   D
      SEG  CORE*,BB
      IN   DEFCOR,DFCOR
    
```

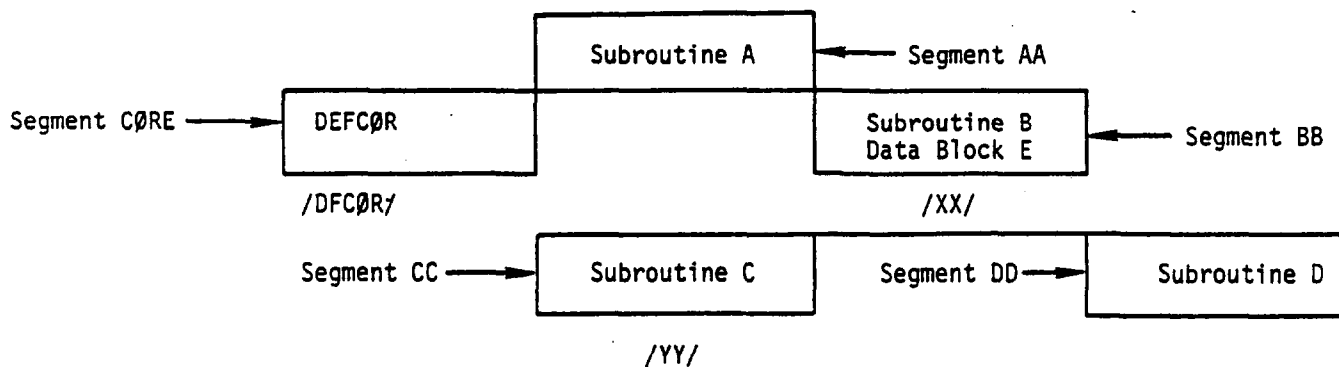


Figure 1. Example of NASTRAN Overlay on the UNIVAC 1108

NASTRAN - OPERATING SYSTEM INTERFACES

Table 2. Execution Deck Set-Up for NASTRAN on the UNIVAC 1108

@RUN			001
@ASG,T	2,F//PDS/30		002
@ASG,T	3,F//PDS/30		003
@ASG,T	4,F//PDS/30		004
@ASG,T	PDDL,F//PDS/30	. PDDL	005
@ASG,T	PTP,F//PDS/30	. PTP	006
@ASG,T	NPTP,F//PDS/30	. NPTP	007
@ASG,T	UMF,F//PDS/30	. UMF	008
@ASG,T	NUMF,F//PDS/30	. NUMF	009
@ASG,T	PLT1,F//PDS/30	. PLT1	010
@ASG,T	PLT2,F//PDS/30	. PLT2	011
@ASG,T	INPT,F//PDS/30		012
.	.	.	.
.	.	.	.
.	.	.	.
@ASG,T	39,F//PDS/30		037
@HDG,N			038
@XQT	*NASTRAN.LINK1		039

NASTRAN DATA DECKS (Executive Control Deck, Case
Control Deck, Bulk Data Deck)

NASTRAN - OPERATING SYSTEM INTERFACES

THIS PAGE HAS BEEN LEFT BLANK INTENTIONALLY.

5.4.5 Execution Deck Set-Up

The following comments explain particular cards or groups of cards:

1. Card 001 consists of the standard RUN card required by the installation.
2. Cards 002-037 assign the logical files to FASTRAND or some other auxiliary storage device. These assign cards allow a maximum of 1,920 tracks or approximately 3,500,000 words for each file. This assignment can vary per run or per device used.
3. Cards 005-011 have a dual usage and can be used as user tapes. If a checkpoint is required, then card 007 must be replaced with

@ASG,T NPTP,U,TAPE#
4. Depending upon the devices available, the assign cards can reference FASTRAND, drum, disk, or tape.
5. Card 038 turns off the standard system heading.
6. Card 039 initiates execution of LINK1.
7. The NASTRAN data decks are inserted between cards 039 and 040.
8. For user convenience, 002-037 are automatically generated by NASTRAN, except for any assignment supplied by the user (usually just tapes).

The execution deck set-up can then consist of:

```

@RUN      ...etc...
@ASG,T    ...etc...      (option tape assignment)
@HDG,N
@XQT      *NASTRAN.LINK1
{NASTRAN DATA CARDS}
@ADD,P    *NASTRAN.NASTRAN
@FIN
    
```

NASTRAN - OPERATING SYSTEM INTERFACES

5.4.6 Description of NASTRAN Physical Items and Generation of the NASTRAN Executable System

5.4.6.1 Description of NASTRAN Physical Items (Level 17.5.0)

The following seven tapes constitute NASTRAN Level 17.5.0 COSMIC delivery tapes.

All delivery tapes are nine-track and written at a density of 1600 fpi. Each file was written using the "G" option of the UNIVAC @COPY utility.

NAST01 - EXECUTABLE

This tape contains the executable version of NASTRAN plus 15 print files (1 file per link) of the load map output of the Collector.

File 1 - NASTRAN executable

This file contains 32 elements: 16 absolute and symbolic elements each for LINK1, LINK2,...,LINK15, and LINK99. 2000 tracks are required for this file.

Files 2-17 - Collector's load map

These files contain the print files generated by the UNIVAC collector when collecting LINK1, LINK2,...,LINK15, and LINK99. (File 2 has the LINK1 print file, File 3 has the LINK2 print file, etc.) 128 tracks are required for each of these files.

NAST02 - SOURCE

These tapes contain the source code for all UNIVAC 1108/1110 NASTRAN decks.

File 1 - Machine Independent Source Elements (A-M)

Contains all machine independent source elements that begin with A through M. 1586 tracks are required for this file.

File 2 - Machine Independent Source Elements (N-Z)

Contains all machine independent source elements that begin with N through Z. 1637 tracks are required for this file.

File 3 - Machine Dependent Source Elements

Contains all machine dependent FORTRAN and Assembly source elements. 40 tracks are required for this file.

NAST03 - RELOCATABLE

This tape contains all the relocatable elements for NASTRAN plus the SUBSYS source elements for all links. The SUBSYS element names are LINK1, LINK2,...,LINK14, LINK15, and LINK99.

File 1 - NASTRAN relocatable elements and SUBSYS elements. 868 tracks are required for this file.

NAST04 - DEMO ITEMS

This tape contains the NASTRAN User Master File, the Demonstration Problem Driver Decks and a print file of the UMF creation run.

NASTRAN ON THE UNIVAC 1108/1110 (EXEC 8)

File 1 - User Master File. 210 tracks are required for this file.

File 2 - Demonstration Problem Driver Decks (82 elements). 12 tracks are required for this file.

File 3 - Print of UMF creation job. 167 tracks are required for this file.

NAST05 - DEMO PRINT FILES

This tape contains the print files for the 82 NASTRAN Demonstration Problem executions. Each file requires 128 tracks.

Files 1-82 - NASTRAN Demonstration Problem print files.

NAST06 - COMPILATION LISTINGS (A-M)

This tape contains the print files generated during the compilation of the subroutines from A-M. Each file requires 128 tracks.

Files 1-18 - Compilation print files (A-M).

NAST07 - COMPILATION LISTINGS (N-Z)

This tape contains the print files generated during the compilation of the subroutines from N-Z followed by the machine dependent routines. Each file requires 128 tracks.

Files 1-13 - Compilation print files (N-Z).

File 14 - Compilation listings machine dependent FORTRAN subroutines.

File 15 - Assembly listings machine dependent ASSEMBLY subroutines.

5.4.6.2 Procedure to Load, Catalog and Execute a Single NASTRAN Problem

1	@RUN	
2	@QUAL	NASTRAN
3	@ASG,UP	*NASTRAN,F/1/TRK/2000
4	@FREE	*NASTRAN.
5	@ASG,T	TAP,U,EXEC
6	@REWIND	TAP.
7	@COPY,G	TAP,U9V,NAST01
8	@FREE	TAP.
9	@HDG,N	
10	@XQT	*NASTRAN.LINK1
11	ID	
	:	
12	CEND	NASTRAN executive, case control,
	:	and bulk data cards
	:	
13	BEGIN BULK	
14	ENDDATA	
15	@ADD,P	*NASTRAN.NASTRAN
16	@FIN	

NASTRAN - OPERATING SYSTEM INTERFACES

<u>Item</u>	<u>Description</u>
1	Standard UNIVAC RUN card required by the installation.
2	Initializes the EXEC 8 qualifier assigned to files to be "NASTRAN" instead of contents of the project field read from the RUN card.
3	Allocates a maximum of 2000 tracks for the NASTRAN Executable file *NASTRAN. This file will be catalogued unconditionally ("U" option) for public use ("P" option).
4	The FREE command causes the file *NASTRAN to be catalogued. Note the ASG does not cause the file to be catalogued but only schedules it for cataloging at job completion or when FREEd.
5	Assigns the COSMIC Executable tape to be mounted on any 9 track tape drive ("U" - device code) to be mounted temporarily ("T" option) at 1600 fpi ("V" device code).
6	Rewinds the Executable tape.
7	Copies the first file of the Executable tape to the file *NASTRAN under the roll-in ("G" option) format. Note, these COSMIC tapes were created using the "G" option and therefore, must be read with the "G" option.
8	Releases the Executable tape and thereby frees the tape drive back to the EXEC 8 system.
9	Turns off the EXEC 8 heading since NASTRAN provides its own page heading.
10-14	Executes the absolute element LINK1 of the *NASTRAN file. LINK1 contains the Preface modules for NASTRAN. All NASTRAN executive, case control, and bulk data cards follow this card. If this NASTRAN problem uses the INPUT module, the cards to be input to the INPUT module should be added after the ENDDATA card. These F0RTRAN data cards must be preceded by the @XQT *NASTRAN.LINK2 card.
15	Dynamically adds the execution driver for each link needed. Each link is executed by the pair @XQT *NASTRAN.DRIVER and @XQT *NASTRAN.LINKx.
16	Standard Exec 8 job FIN card.

NOTE: If plots are to be generated by NASTRAN, the plot file (PLT2) must be assigned to tape and the assign card must be placed before the first @XQT *NASTRAN.LINK1 card. An example for assigning a 7 track 556 fpi ("M" option) PLT2 plot tape is as follows:

```
@ASG,TM    PLT2,U,xxxxxW
```

An example of assigning a 9 track tape for the Calcomp plotter is as follows (note the quarter word mode requirement):

```
@ASG,TH    PLT2,U9V/////Q,xxxxxW
```

Similarly if checkpointing is done by NASTRAN there should be an assign card for the file NPTP before the first @XQT *NASTRAN.LINK1. The file may be either a tape or drum file. An example of allocating a NPTP file to drum is as follows:

```
@ASG,UP    NPTP,F/1/PDS/4
```

This will catalog the NPTP for public use and allocate a maximum of 4 positions (256 tracks) and a guaranteed minimum of 64 tracks (1 position) to the file.

5.4.6.3 Procedure To Copy and Print the Collector's Load Map for LINK1.

```
1    @RUN
2    @ASG,UP          PRINT.,F
3    @ASG,T          TAP,U9V,NAST01
4    @REWIND          TAP.
5    @MOVE            TAP.,1
6    @COPY,G          TAP.,PRINT.
7    @FREE            PRINT.
8    @FREE            TAP.
9    @SYM,U           PRINT,,PR
10   @FIN
```

<u>Item</u>	<u>Description</u>
1	Standard UNIVAC run card required by the installation.
2	Assign a file PRINT to be catalogued as public with a file allocation of 128 tracks (default).
3	Assign the COSMIC Executable tape on any 9 track tape drive and 1600 fpi.
4	Rewind the COSMIC Executable tape.
5	Moves the COSMIC Executable tape to the beginning of the second file.
6	Copies the second file of the COSMIC executable tape to the file PRINT. under the roll-in ("G" option) format.
7	Releases the file PRINT to the EXEC 8 system to be catalogued.
8	Releases the COSMIC Executable tape and thereby frees the tape drive back to the EXEC 8 system.
9	Executes the SYM processor to print the PRINT file on the PR print device. NOTE: PR is an installation dependent parameter.
10	Standard Exec 8 job FIN card.

NASTRAN - OPERATING SYSTEM INTERFACES

NOTE: This same procedure can be used to print the third file of the COSMIC Demo Items tape (print file of the UMF verification job) by changing card 5 to position the tape to the beginning of the third file:

@MOVE TAP.,2

5.4.6.4 Procedure To Read the COSMIC Relocatable Tape and Catalog the File on FASTRAND.

1	@RUN	NASTRAN
2	@ASG,UP	OBJ,F/1/TRK/868
3	@ASG,T	TAPE,U9V,NAST03
4	@REWIND	RELDC.
5	@COPY,G	RELDC.,OBJ.
6	@FIN	

<u>Item</u>	<u>Description</u>
1	Standard UNIVAC RUN card required by the installation.
2	Assign the file OBJ to FASTRAND allowing a maximum of 868 tracks. This file will be unconditionally ("U" option) catalogued public ("P" option) at the end of the job.
3	Assign the COSMIC Relocatable tape to any 9 track tape drive at a density of 1600 fpi.
4	Rewind the COSMIC Relocatable tape.
5	Copies the first file of the COSMIC Relocatable tape to the file OBJ under the roll-in ("G" option) format.
6	Standard Exec 8 job FIN card.

NOTE: Similarly the COSMIC Source Tape may be read. Also the second and third files of the COSMIC Demo Items tape may be read by positioning the tape with the @MOVE command to the beginning of the appropriate file before issuing the @COPY command.

5.4.6.5 Procedure to Alter a Subroutine, Recollect a Link, and Execute a Problem

1	@RUN	
2	@QUAL	XXXX
3	@ASG,A	MISØUR.
4	@ASG,A	ØBJ.
5	@ASG,A	*NASTRAN.
6	@FØR,US	MISØUR.NASCAR,ØBJ.NASCAR
7	-1	
8	C	
9	@PREP	ØBJ.
10	@DELETE,A	*NASTRAN.LINK1
11	@PACK	*NASTRAN.
12	@MAP,SL	ØBJ.LINK1,*NASTRAN.LINK1
13	@HDG,N	
14	@XQT	*NASTRAN.LINK1 NASTRAN Executive, Case Control, and Bulk Data Cards
15	@XQT	*NASTRAN.NASTRAN
16	@FIN	

<u>Item</u>	<u>Description</u>
1	Standard UNIVAC RUN card required by the installation.
2	Defines the qualifier to be applied to the *NASTRAN file. The qualifier "xxxx" is dependent upon how this file was cataloged. Whatever qualifier was used when this file was cataloged should be used in the "xxxx" field. If no qualifier was supplied when the file was cataloged, the default qualifier is the project field found on the @RUN card of the job that cataloged the file.
3	Assigns file MISØUR which is cataloged and contains the element NASCAR. Should the file not be cataloged the run will abort ("A" option).
4	Assigns file ØBJ which is cataloged and contains NASTRAN's relocatable elements and SUBSYS (overlay control cards) elements.
5	Assigns the file *NASTRAN which is cataloged and contains the NASTRAN absolute elements.
6	Executes the FØRTRAN compiler to update ("U" option) and compile with a source ("S" option) listing the element NASCAR. The relocatable element generated will be written in element NASCAR of the file ØBJ. This will not save the source generated. To replace the source of NASCAR with the updated source, the following card should be used instead: @FØR,US MISØUR.NASCAR,ØBJ.NASCAR,MISØUR.NASCAR
7	Control card to the FØRTRAN compiler directing it to insert the following card (item 8) after card number 1 in the element NASCAR.
8	A comment card to be inserted by the fortran compiler after card number 1 in the element NASCAR.

NASTRAN - OPERATING SYSTEM INTERFACES

<u>Item</u>	<u>Description</u>
9	Generates an entry point table for the ØBJ file. This card is required any time a relocatable element is inserted into the ØBJ file. Without a PREP, the collector may be unable to find all relocatable elements in the ØBJ file.
10	Deletes the current absolute ("A" option) element LINK1 from the file *NASTRAN.
11	Packs the *NASTRAN file to make space available for the new LINK1 absolute element to be collected.
12	Executes the collector to generate the absolute element LINK1 in the file *NASTRAN using the SUBSYS cards in element LINK1 of file ØBJ.
13	Turns off the standard UNIVAC Exec 8 heading. NASTRAN will supply its own heading.
14	Executes the NASTRAN preface found in LINK1.
15	Dynamically executes other links required.
16	Standard Exec 8 job FIN card.

5.4.6.6 Procedure to Run a NASTRAN Demonstration Problem

1	@RUN	
2	@QUAL	xxxx
3	@ASG,A	*NASTRAN.
4	@ASG,A	DRIVER.
5	@ASG,A	UMF.
6	@ADD	DRIVER.D11010
7	@FIN	

<u>Item</u>	<u>Description</u>
1	Standard UNIVAC RUN card.
2	Defines the qualifier for the *NASTRAN file.
3	Assigns the *NASTRAN file.
4	Assigns the DRIVER file that contains the Demonstration Problem Driver Decks.
5	Assigns the User Master File.
6	Adds the Demonstration Problem Driver Deck for Demo D11010.
7	Standard Exec 8 job FIN card.

NASTRAN ON THE UNIVAC 1108/1110 (EXEC 8)

5.4.6.7 Procedure to Run NASTRAN Using Multi-Stage Substructuring

```

1      @RUN
2      @QUAL                XXXX
3      @ASG,A              INPT,F
4      @ASG,A              *NASTRAN.
5      @XQT                 *NASTRAN.LINK1
6      NASTRAN              FILE=INPT
7      ID
      .
      .
      .
8      SØF(1)=INPT
9      @XQT                 *NASTRAN.NASTRAN
10     @FIN
    
```

<u>Item</u>	<u>Description</u>
1	Standard UNIVAC RUN card.
2	Defines the qualifier for the *NASTRAN file.
3	Assigns the Substructure Operating File (SØF).
4	Assigns the NASTRAN executable.
5	Executes NASTRAN
6	NASTRAN card to specify the INPT as being unavailable for scratch use.
7	NASTRAN data card assigning the SØF to the INPT file.
10	Standard EXEC 8 job FIN card

5.4.7 Machine Dependent Routines

The routines discussed in this section consist of those programs unique to the UNIVAC 1108 or those which are implemented differently from other machines. The language for each deck is indicated by an F (FORTRAN) or S (SLEUTH) following the deck name. GINØ is discussed in Section 5.4.16, matrix packing is discussed in Section 5.4.17, and single precision decks are discussed in Section 5.4.18.

1. MAINi (F)

MAINi is the main program for LINKi. Its functions are to call SETC, DEFCØR and XSEMI.

2. DEFCØR (F)

DEFCØR is responsible for calling ZCØRSZ to reserve core for NASTRAN.

3. MAPFNS (S)

In addition to the standard functions described in Section 3, the following functions were added:

a. FACIL(UNIT, ID) - Sets ID = 0 if UNIT is assigned to FASTRAND

Sets ID = 2 if UNIT is not assigned.

Sets ID = 7 if UNIT is assigned to a 7 track tape drive

Sets ID = 9 if UNIT is assigned to a 9 track tape drive

b. XDATE(DATE) - Return the date in a four-word array DATE.

DATE(1) = MONTH

DATE(2) = DAY

DATE(3) = YEAR

DATE(4) = time in elapsed wall clock seconds from midnight

c. ZCØRSZ(X(1), LENGTH) - Reserves core between X(1) and 65K (or the value supplies in the

HICØRE parameter of the NASTRAN card) and returns the length in LENGTH.

d. SEC(T) - Return CPU time used in T in integer milliseconds.

e. LOGFIL(I(1), WØRDS) - Writes the number of words in WØRDS starting with I(1)

on the run log file.

f. ELAPSE(T) - Return in T the wall clock time used.

g. TSWAP(ID) - Unloads a unit (ID) and mounts another tape on that unit.

- h. ADDCRD(IMAGE,LENGTH) - Add a control card image of length LENGTH to the control stream.
 - i. COM(IMAGE,LENGTH) - Type an image of length LENGTH on the operator console.
 - j. ZCORSZ - Perform the function of CORSZ.
 - k. PCTABL(BUF) - Returns the first 25 words of the PCT table in BUF.
 - l. ITRENT - Location to return to in the event of a guard mode interrupt. ITRENT then calls YTRACE and PPDUMP.
 - m. CONTIN - Places ITRENT in the guard mode interrupt return word. CONTIN is called by BGNSYS on the 1108 only.
4. UTIL (F)
- a. SECND(T) - Returns the CPU time in floating point seconds by calling SEC.
 - b. TDATE(II) - Returns the date in the first three words of II and zero in the fourth word of II by calling XDATE. It also initializes STIME (SYSTEM(32)) to the elapsed wall clock seconds from midnight.
 - c. KLOCK(I) - Returns the current CPU time used in integer seconds measured from NASTRAN start.
 - d. CONMSG(IX,IY,IZ) - Writes a message of length IY words from the first IY words of IX. This message is displayed on the operator's console if IZ \neq 0 (controlled by DIAG 5 and 6). The messages are normally stored in the log file. The number of messages previously written is accumulated in SYSTEM (7). If SYSTEM (7) > 100, messages are included into the normal print file. The messages printed out by CONMSG are preceded by timing data as follows: XX CPU^S YY COR^S ZZ ELP^S where XX is determined by a call to SEC, YY is a measure of active core residency time and ZZ is the elapsed wall clock time.
 - e. XTRACE - Is a print routine for a trace back routine.
 - f. WALTIM(I) - Returns the elapsed wall clock seconds from midnight.
5. XEOT (F)
- XEOT causes reels to be swapped for multireel old and new problem tapes.

NASTRAN - OPERATING SYSTEM INTERFACES

9. PEXIT (F)

PEXIT performs normal (machine dependent) termination procedures and also sets the condition code (15) for final termination.

10. SEARCH (F)

SEARCH frees the ØPTP, UMF, and NUMF at the conclusion of LINK1 if these files reside on a tape. In addition, SEARCH sets the condition code for the next link to be executed.

11. SGINØ (F)

SGINØ performs physical I/Ø on the plot tapes (PLT1 and PLT2).

- a. SØPEN (\$N, PLTTP, BUF, BUFSIZ) - Establishes a buffer for PLTTP of length BUFSIZ at location BUF. It also sets the parity of the file, depending on whether PLTTP = PLT1 or PLT2.
- b. SWRITE (PLTTP,A,N,EØR) - Writes N characters from N words starting at A(1) into the plot buffer.
- c. SCLØSE (PLTTP) - Flushes the plot buffer, places a physical end-of-file on the plot tape, and backspaces over it.
- d. SEØF (PLTTP) - Places a physical end-of-file on PLTTP.

12. KØRSZ (F)

KØRSZ provides the length of the open core length obtained from the SLEUTH subroutine ZCØRSZ.

13. MPYQ (S)

MPYQ is a SLEUTH deck written to increase the speed of MPYAD's inner loops. The documentation for MPYQ is machine independent, and is in Section 3 as an auxiliary routine to MPYAD.

14. PPDUMP (F)

Supplies entry points DUMP, PDUMP, and PPDUMP.

PPDUMP captures control after a guard mode interrupt. DUMP and PDUMP will print a trace back via YTRACE and if DIAG1 is on obtain a formatted dump via adding the element DMP. That part of open core that resides in the load module region will be printed by PPDUMP. The element DMP will execute a ØPMD and execute *NASTRAN LINK99 which will print the NASTRAN fatal error if any occurred. The fatal message is past by PDUMP and DUMP to LINK99 by writing the common block /MSGX/ on fortran unit 2.

15. YTRACE (S)

YTRACE provides a walk back trace to XSEMI from the calling routine to aid in subroutine debugging.

16. CØN8BD (F)

Defines several constants such as etc.

17. SEGALT (S)

SEGALT provides the last d-bank address of the highest segment that has been loaded and the last D-bank address of the executable's region. This gives PPDUMP the address of open core that resides in the executable's region.

18. SØFIØ (F)

Directs the input/output operations for substructuring between core and the random access storage device. (see section 3.6)

a. SØFIØ (IRW,IBLKNM,IBUFF) - Writes (IRW=1) or reads (IRW=2) block number IBLKNM from/to the buffer IBUFF.

19. SØTRAN (F)

Performs actual I/Ø for SØFIØ using NTRAN.

a. SØTRAN (\$N,ØPCØDE,BUF) - Performs I/Ø functions based on the value of ØPCØDE. \$N is the error return and BUF is I/Ø buffer supplied.

<u>Opcode</u>	<u>Function Performed</u>
2	Write an I/Ø block
3	Read an I/Ø block
4	Position file backwards
5	Position file forward

20. SETC (F)

SETC calls entry point ADDCRD in subroutine MAPFMS to dynamically add the symbolic element that contains the @XQT card for the next link to be executed.

a. SETC (I) - I is the integer value of the next link to be executed.

21. EXFØRT (F)

EXFØRT performs formatted I/Ø for module EXIØ (see Section 4.130.1)

a. EXFØRT (RW,U,F,BUF,NWDS,PREC,DBUF), where

RW = I/O operation to be performed

1. read a physical record
2. locate record destination in the buffer
3. position the file
4. write a logical end of file

U = unit number of file

F = index to FORMAT in label common block /EXIØ2F/

BUF = buffer containing single precision values

NWDS = number of words to read or write

PREC = precision flag

1. single precision
2. double precision

DBUF = buffer containing double precision values

NASTRAN ON THE UNIVAC 1108/1110 (EXEC 8)

THIS PAGE HAS BEEN LEFT BLANK INTENTIONALLY.

NASTRAN - OPERATING SYSTEM INTERFACES

5.4.8 GINØ (S-Generalized Input/Output Processor for NASTRAN)

The GINØ package for the UNIVAC 1108 consists of the following decks with entry points as indicated.

<u>Deck</u>	<u>Entry Point</u>	
GINØ	BCKREC	GETSRT
	CLØSE	PUTSTR
	ENDGET	QØPEN
	ENDGTB	READ
	ENDPUT	RECTYP
	EØF	SAVPØS
	FILPØS	SKPFIL
	FWDREC	WRITE
	GETSTB	RDBLK
		WRTBLK
IØ1108	IØ1108	
ØPEN	ØPEN	

Documentation for GINØ and the calling sequence for its entry points and ØPEN are essentially machine independent and are discussed in Section 3 of the Programmer's Manual. The documentation for IØ1108 is unique to the UNIVAC 1108 and is provided as follows:

Purpose

To perform I/Ø operations for NASTRAN data blocks using NTRAN.

Calling Sequence

CALL IØ1108 (ØPCØDE,BLØCK), where

ØPCØDE = the operation code for the operation to be performed according to the following list:

1. - Rewind
2. - Write
3. - Read
4. - Backspace
8. - Reread
9. - Swap tape reels
10. - Unload

BLØCK = address of the GINØ block for the requested operation.

Description

All I/Ø operations are performed using NTRAN. All statuses are reported by NTRAN to IØ1108 and referred by IØ1108 to GINØ by setting a flag in the /GINØX/ common block. The statuses returned are as follows:

<u>Status</u>	<u>Description</u>
-2	EØF encountered
-3	Hardware, parity or character count errors

Any I/Ø error correction to be performed is done so via direction of GINØ (e.g. tape swapping).

When an operation is requested of IØ1108 such as rereading or backspacing where the type of file device (i.e., tape or mass storage) is required, a call is made to FACIL for file type determination.

If any errors not associated with I/Ø such as an invalid opcode are detected by IØ1108, a call to GPERR with the associated error message number is made.

5.4.9 The LINK99 Element

The DMP element found in the *NASTRAN file consists of the following cards:

@PMD,LP

@XQT *NASTRAN.LINK99

The LINK99 element is automatically added by subroutine PPDUMP when a guard mode or a NASTRAN fatal error occurs and DIAG 1 is on. The @PMD card will cause a formatted dump and the @XQT executes LINK99 which will print out any NASTRAN warning or fatal errors that were accumulated and not printed. It is necessary to have a separate absolute element for this purpose to allow the PMD to give a true picture of core when the problem occurred. If the message was printed via a call to MSGWRT by PPDUMP the core would be modified because the segment containing MSGWRT would have been loaded.

5.4.10 UNIVAC Overlay Diagrams

The NASTRAN overlay structure for the UNIVAC 1108 may be obtained by printing the contents of the tape (see Section 5.4.6.1).

NOTES:

The following comments are included to aid the user in reading the overlay diagram.

1. The directives (SUBSYS) do not contain "IN" cards for every subroutine to be added in a specified segment. This is to take advantage of the Collector's capability of placing subroutines to satisfy external references within the required segments. The Overlay Diagrams are generated from the SUBSYS cards and are therefore incomplete. The Collector's load map should be referenced.
2. Note that on the 1108, contrary to any other NASTRAN computer, there are really two similarly shaped overlay pictures, one for the I bank (instructions), and another for the D bank (data).

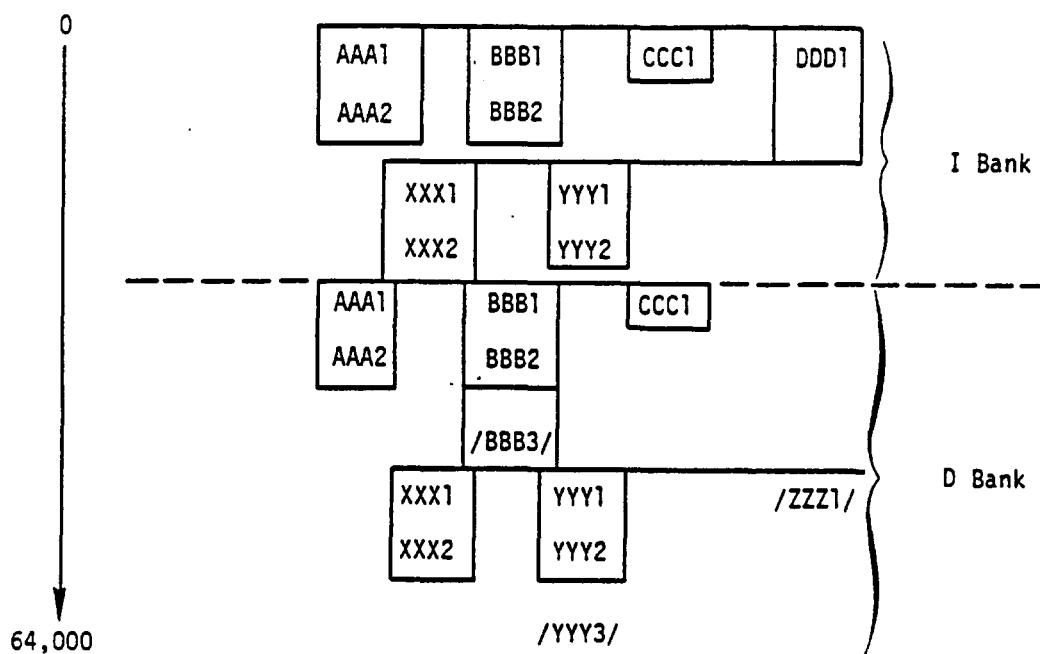


Figure 2. Overlay Sample

Several important points result from this figure. First, all modules are penalized by long instruction banks in other modules. Second, a reasonable overlay tree derived from total routine lengths may not be reasonable for the 1108. Thus, the 1108 overlay tends to be more complex than the overlay for the other NASTRAN computers. Critical statistics are the length of the I-bank, the length of the D-bank, the total length of each link, and the smallest open core address possible (End of D-bank of the root segment).

3. Note that the same module may appear in more than one link. The following table (Figure 18) shows which link each module occurs in on the 1108.

MATERIAL PREVIOUSLY ON PAGES 5.4-25 THROUGH 5.4-56 HAS BEEN DELETED.

NASTRAN - OPERATING SYSTEM INTERFACES

THIS PAGE HAS BEEN LEFT BLANK INTENTIONALLY.

MODULE INDEX	LINK NAME OF MODULE	MODULE ENTRY POINT NAME	LINKS POINT NAME	MODULE RESIDES IN CH	1108												
5	CHAPMT	XCHM	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
6	CEET	XCEI	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
6	JUMP	XCEI	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
7	CGND	XCEI	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
8	SAVE	XSAVE	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
9	PURCE	XPURCE	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
10	EGCIV	XEGCIV	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
11	END	XCEI	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
12	EXIT	XCEI	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
14	ADD	BADD				4											
15	ABDS	BABDS							7								
16	AMC	APC									9						
17	AMF	APF									9						
18	APC	APL									9						
19	PRG	PRG										10					
20	CASE	CASE										10					
21	CYCT1	CYCT1							7								
22	CYCT2	CYCT2							7								
23	CEAD	CEAD											11				
24	CURV	CURV													13		
26	DER	DER								8							
27	DER1	DER1												12			
28	DER2	DER2												12			
29	DERMM	DERMM												12			
30	DECOMP	DECOMP							7								
31	DIAGONAL	DIAGONAL															15
32	DRD	DRD						6									
33	DSCHK	DSCHK							7								
34	DSRG1	DSRG1													13		
35	DSRG2	DSRG2				4											
37	DUMMO1	DUMMO1							7								
38	DUMMO2	DUMMO2							7								
39	DUMMO3	DUMMO3							7								
40	DUMMO4	DUMMO4							7								
42	ENAL	ENAL								8							
43	ENG	ENG								8							
44	FA1	FA1												11			
45	FA2	FA2												11			
46	FOL	FOL							7								
47	FRLG	FRLG															
48	FRRD	FRRD												10			
50	GI	GI									9			10			
51	GRAD	GRAD												10			
52	GRAM	GRAM												10			
53	GPI	GPI															

Figure 18. Link Specification Table.

5.4-57 (12/29/78)

MODULE INDEX	ORIG NAME OF MODULE	MODULE ENTRY POINT NAME	LINKS MODULE RESIDES IN ON 1108														
54	CF2	CF2	2														
55	CF3	CF3	2														
56	CF4	CF4		4													
57	CPCYC	CPCYC					7										
58	GPEDR	GPEDR														13	
59	GPSP	GPSP		4													
60	GPWG	GPWG		4													
62	INPUT	INPUT	2														
63	INPUT11	INPUT11	2														
64	INPUT12	INPUT12	2														
65	INPUT13	INPUT13	2														
66	INPUT14	INPUT14	2														
67	MATGEN	MATGEN					7										
68	MATGPR	MATGPR						8									
69	MATPRN	MATPRN						8									
70	MATPRI	PRIINT						8									
71	MCE1	MCE1		4													
72	MCE2	MCE2		4													
73	MERGE	MERGE1					7										
74	MOLA	MOLA					7										
76	MCDACC	MCDACC														12	
77	MOUR	MOUR					7										
78	MOLC	MOLC					7										
79	MPYAD	MPYAD					7										
80	MTRXIN	MTRXIN								10							
81	OFF	OFF															14
82	OPTPR1	OPTPR1	2														
83	OPTPR2	OPTPR2						8									
85	OUTPUT	OUTPUT															14
86	OUTPUT1	OUTPUT1															14
87	OUTPUT2	OUTPUT2															14
88	OUTPUT3	OUTPUT3															14
89	OUTPUT4	OUTPUT4															14
90	PARAM	CPARAM	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
91	PARAML	PARAML	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
92	PARAMF	PARAMF	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
93	PARTN	PARTN1						7									
95	MFED1	MFED1															15
96	MFED2	MFED2															15
97	CMFED1	CMFED1															15
98	PLA1	PLA1		7													
99	PLA2	PLA2															
100	PLA3	PLA3														13	
101	PLA4	PLA4														13	
103	FLUT	DFLOT	2													13	

Figure 18. Link Specification Table (Continued)

5.4-58 (12/29/73)

MODULE - MAP NAME - MODULE ENTRY - LINKS MODULE RESIDES IN ON 1108
INDEX OF MODULE POINT NAME

Index	Variable	Value
104	FLTSET	2
105	FLITRAK	8
106	PRIMSG	2
107	PRTFARM	8
108	RANCOM	14
109	RBMG1	4
110	RBMG2	4
111	RBMG3	4
112	RBMG4	4
114	READ	6
115	RMG	5
116	SCALAR	15
117	SCE1	4
118	SDR1	12
119	SDR2	13
120	SDR3	14
121	SDRHT	13
122	SEEMAT	2
124	SETVAL	2
125	SMA1	3
126	SMA2	3
127	SMA3	4
128	SMP1	4
129	SMP2	4
130	SMPYAD	7
131	SOLVE	7
133	SSG1	5
134	SSG2	5
135	SSG3	5
136	SSG4	5
137	SSGHT	5
138	TA1	2
139	TAEPCN	9
141	TAEPR1	8
142	TAEPT	8
144	TINTST	9
145	TRD	11
146	TRHT	11
147	TRLC	5
148	TRNSP	7
149	UNFOLD	7
150	UPARTN	7
151	VDR	12
152	VEC	7
154	XYFLOT	2

Figure 18. Link Specification Table (Continued)

5.4-59 (12/23/73)

NASTRAN - OPERATING SYSTEM INTERFACES

MODULE INDEX	MAP NAME OF MODULE	MODULE ENTRY POINT NAME	LINKS MODULE RESIDES IN ON	1108
155	XYPRNPLT	XYPRNLT		14
156	XYTPAN	XYTPAN		14
158	COMP1	COMP1		15
159	COMP2	COMP2		15
160	EXIO	EXIO		15
161	RCOVR	RCOVR		15
163	RCCOVR3	RCCOVR3		15
164	REDUCE	REDUCE		15
165	SGLN	SGLN		15
166	SOFI	SOFI		15
167	SOF0	SOF0		15
168	SOFUT	SOFUT		15
169	SUBPH1	SUBPH1		15
170	PLTMRG	PLTMRG		15
172	COPY	COPY	7	
173	SWITCH	SWITCH	7	
174	HPY3	HPY3	7	
175	SDCMPS	DDCMPS	7	
176	LODAPP	LODAPP		15
177	GPSPC	GPSPC	4	
178	LGCK	LGCK		12
179	ADR	ADR		10
180	FRND2	FRND2		10
181	GUST	GUST		10
182	IFT	IFT		10
183	LAMX	LAMX	9	
184	MTXTEST	MTXTS		14
185	LPA	LPA	8	

Figure 18. Link Specification Table (Continued)

ADRI				10	
ADPPPT				10	
ADRY				10	
ADR				10	
AIS	5				13
AI	5				13
ALLMAT			11		
AMATRX					13
AMGBFS			9		
AMGMN			9		
AMGP2			9		
AMGRND			9		
AMGSRA			9		
AMGX			9		
AMG			9		
AMPA1X			9		
AMPA			9		
AMPA1X			9		
AMPR1			9		
AMPR2X			9		
AMPR2			9		
AMPR			9		
AMPCOM			9		
AMPC1X			9		
AMPC1			9		
AMPC2			9		
AMPC			9		
AMPD1X			9		
AMPD			9		
AMPEX			9		
AMPE			9		
AMPEX			9		
AMPE			9		
AMP			9		
APDCS			9		
APDF			9		
APDNF			9		
APDR			9		
APD12C			9		
APD1			9		
APD1C			9		
APD1D			9		
APD12			9		
APD2			9		
APD3			9		
APD4			9		
APD5			9		

Figure 19. Alphabetical Deck Name - Link Table

APD								9											
ARRM						6													
ASCM01	1																		
ASCM02	1																		
ASCM03	1																		
ASCM04	1																		
ASCM05	1																		
ASCM06	1																		
ASCM07	1																		
ASCM08	1																		
ASCM09	1																		
ASCM10	1																		
ASCM11	1																		
ASCM12	1																		
ASCM13	1																		
ASDRD	1																		
ASDMP	1																		
ASDZZZ	1																		
ASPRD	1																		
ATEIG												11							
AUTDCK	1																		
AUTDCH	1																		
AUTDHD	1																		
AUTDSM	1																		
AUTDSV	1																		
AXIS10		2																	
AXIS3		2																	
AXIS		2																	
RARD									8										
RARS									8										
RAR						5													
RASGLB						5													
RCB				3		5			8							13			
PDAT01																	15		
PDAT02																	15		
PDAT03																	15		
PDAT04																	15		
PDAT05																	15		
PDAT06																	15		
REFSMAT								9											
PINT						5										13			
RISHL		2										12							
RISLNC	1	2	3		5		7	8	9	10		12	13						
RITPAT																			
RITPOS	1	2	3	4	5	6	7	8	9	10	11	12	13	14			15		
PLANK	1	2	3	4	5	6	7	8	9	10	11	12	13	14			15		
BMGTNS																			
BMGZZZ																			

Figure 19. Alphabetical Deck Name - Link Table (Continued)

RMG										10					
RORDER		2													
RTSTPP	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PUG	1								9	10					15
BVISC			3					8							
CALCV				4	5	6	7		9	10	11	12			15
CASCOR										10					
CASE										10					
CDCMPX							7		9	10	11	12			15
CDCOMP							7		9	10	11				15
CDFTMX											11				
CDFTM2											11				
CDFTM											11				
CDIFAS											11				
CDIVID											11				
CDTFBS											11				
CFADA1											11				
CFAD1A											11				
CFAD1X											11				
CFAD											11				
CENTPE		2													
CFACTR									9	10	11				
CFACTX									9	10	11				
CFBSOR									9	10	11				
CFBSRX									9	10	11				
CFCNTL											11				
CFEER1											11				
CFEER2											11				
CFEER3											11				
CFEER4											11				
CFER3D											11				
CFER3S											11				
CFE1AN											11				
CFE1MY											11				
CFE2AN											11				
CFE2MY											11				
CFNOR1											11				
CFNOR2											11				
CF1FBS											11				
CF1ORT											11				
CF2FBS											11				
CF2ORT											11				
CHAR94		2													
CHKOPN															15
CHRDPR		2													
CINFBS											11				
CINFBX											11				
CINVPR											11				

Figure 19. Alphabetical Deck Name - Link Table (Continued)

CINVPX											11				
CINVP1											11				
CINVP2											11				
CINVP3											11				
CINVX											11				
CINV1X											11				
CINV2X											11				
CINV3X											11				
CLSTAB	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
CLSTRS	2	3		5			8				12	13			
CLVEC										11					
CMAUTD														15	
CMRFND														15	
CMRZZZ														15	
CMR001														15	
CMR002														15	
CMR003														15	
CMR004														15	
CMCASE														15	
CMCKCD														15	
CMCKDF														15	
CMCNMR														15	
CMCONT														15	
CMDISC														15	
CMHGEN														15	
CMHWRT														15	
CMHCON														15	
CMRD2A														15	
CMRD2B														15	
CMRD2C														15	
CMRD2D														15	
CMRD2E														15	
CMRD2F														15	
CMRD2G														15	
CMRD2Z														15	
CMRD?														15	
CMRELS														15	
CMSEIL														15	
CMSEFO														15	
CMTIMU										11					
CMTOC														15	
CMTRCE														15	
CNORM1										11					
CNORM										11					
CNSTPC	2														
CMRAIN				5											
CMRAN	1														
CMRA1														15	

Figure 19. Alphabetical Deck Name - Link Table (Continued)

CONRA2X																15
CONRA2																15
CONNECT		2														
CONM12							7		9	10	11					15
CONNDAD	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	15
CONNDAS	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	15
CONNE					5											
CONMSG	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	15
CONM10								8								
CONM1S								8								
CONM2D								8								
CONM2S								8								
CONTOR		2														
COPYZZ							7									
COPY							7									
CORSZ										10						
CPYFIL	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	15
CPYSTR	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	15
CRDRD2				4												
CPDRD				4												
CREATE		2														
CRIGGP				4												
CRSUA																15
CSQPTX											11					
CSUB											11					
CSUMM											11					
CTRNSP							7		9	10	11					15
CURCAS												12				
CURVC1													13			
CURVC2													13			
CURVC3													13			
CURVIT													13			
CURVPS													13			
CURVTB													13			
CURVXX													13			
CURV1													13			
CURV2													13			
CURV3													13			
CURV													13			
CXLONP							7		9	10	11					15
CXTRNY											11					
CYCT1Z							7									
CYCT1							7									
CYCT2A							7									
CYCT2B							7		9	10	11	12				
CYCT2X							7									
CYCT2							7									
DACHAR	1															

Figure 19. Alphabetical Deck Name - Link Table (Continued)

NAME	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DADD5							7								
DADD				4			7								
DADDA				4			7								
DADPTR			3		5			8							
DARCNS								8							
DAXR			3		5			8							
DRAP													13		
DCD													13		
DCOMPX				4	5	6	7		9	10	11	12			15
DCONF													13		
DDCMPS							7								
DDCOMP							7								
DDRA1												12			
DDRB1												12			
DDRMC1												12			
DDRMMA												12			
DDRMMP												12			
DDRMMS												12			
DDRMMY												12			
DDRMM1												12			
DDRMM2												12			
DDRMM												12			
DDR1A												12			
DDR1R												12			
DDR1X												12			
DDR1												12			
DDR2												12			
DDR								8							
DDUM1													13		
DDUM2													13		
DDUM3													13		
DDUM4													13		
DDUM5													13		
DDUM6													13		
DDUM7													13		
DDUM8													13		
DDUM9													13		
DECPDE								8					13		
DECOMP				4	5	6	7			10	11		13		15
DECP5X							7								
DECP1X							7								
DECP2X							7								
DECP3X							7								
DEFILE	1														
DELETE															
DFLKLS								8							15
DELSFT		2	3		5			8					13		
DELTKL			3					8							

Figure 19. Alphabetical Deck Name - Link Table (Continued)

5.4-66 (12/29/78)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DESCPP															
DETCKX								8							
DETCK			3												
DETDEF						6									
DETDX						6									
DETFPS						6									
DETMX						6									
DETM1						6									
DETM3						6									
DETM4						6									
DETM5						6									
DETM						6									
DFBS1X							7								
DFBS2X							7								
DFBS							7								
DIAGON															15
DIAGXX															15
DIHEX													13		
DISPLA		2													
DKINT			3					8							
DKI			3					8							
DKLS								8							
DKL			3					8							
DK100			3					8							
DK211			3					8							
DK89			3					8							
DLAMBY									9						
DLAMG									9						
DLAXX									9						
DLRDY									9						
DLRPT2									9						
DLRXX									9						
DLCONM									9						
DLM									9						
DLONP				4	5	6	7			10	11				
DLPT2									9						
DLP2X									9						
DMATRX			3					8							
DMFGR		2													
DMINT			3												
DMI			3												
DMPFIL									9	10					15
DMPY							7								
DMPYAD							7								
DMPYX							7								
DM100			3												
DM211			3												
DM89			3												

Figure 19. Alphabetical Deck Name - Link Table (Continued)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DDOWN						6									
DPDAA						6									
DPDCDM						6									
DPDCNR						6									
DPD1						6									
DPD2						6									
DPD3						6									
DPD4						6									
DPD5						6									
DPD						6									
DPLNT		2													
DPLTST		2													
DPPS									9						
DPPSR									9						
DPZY									9						
DQDMEM													13		
DQUAD													13		
DRAW		2													
DRDD													13		
DPWCHR		2													
DRWDAT		2													
DSCHKX							7								
DSCHK							7								
DSHEAR													13		
DSMG1													13		
DSMG2X				4											
DSMG2				4											
DSTRDY															15
DS1AXX													13		
DS1A													13		
DS1AAA													13		
DS1ADP													13		
DS1AET													13		
DS1A													13		
DS1ETD													13		
DS1ETT													13		
DS1X													13		
DS1													13		
DTRF													13		
DTRANP							7								
DTRANX				4	5	6	7		9						
DTRASC													13		
DTRIA													13		
DTRMEM													13		
DTSHLD													13		
DTSHLS													13		
DUMERG							7								
DUMOD1							7								

Figure 19. Alphabetical Deck Name - Link Table (Continued)

DUMDD2								7							
DUMDD3								7							
DUMDD4								7							
DUMPER	1														
DUMPRM														15	
DUM1XX								7							
DUM1					5										
DUM2XX								7							
DUM2					5										
DUM3XX								7							
DUM3					5										
DUM4XX								7							
DUM4					5										
DUM5					5										
DUM6					5										
DUM7					5										
DUM8					5										
DUM9					5										
DUPART								7							
DVECTR		2													
DYPZ										9					
DZPY										9					
DZYMAT										9					
DZY										9					
EADD						6									
ECTLNC		2						8							
EDIT														15	
EDTL					5										
EGNVCT															
EJDUM2	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
EJECT		2						8	9			12	13		
EKTRBD								8							
EKTRBS								8							
EKTRMD								8							
EKTRMS								8							
FLFLRL		2													
ELIMX				4											
FLIM				4											
FMAATO								8							
FMASTO								8							
FMAXXX								8							
FMA1D								8							
FMA1S								8							
FMA1XX								8							
FMA1								8							
FMA								8							
FMGCNG								8							
FMGCOR								8							

Figure 19. Alphabetical Deck Name - Link Table (Continued)

EMGDIC	8
EMGEST	8
EMGFIL	8
EMGFIN	8
EMGNLD	8
EMGNUT	8
EMGPRM	8
EMGPRO	8
EMGSNC	8
EMGTAB	8
EMGTRX	8
EMGXXX	8
EMGX01	8
EMGX02	8
EMGX03	8
EMGX04	8
EMGX05	8
EMGX06	8
EMGX07	8
EMGX08	8
EMGX09	8
EMGX10	8
EMGX11	8
EMGX12	8
EMGX13	8
EMGX14	8
EMGX17	8
EMGX18	8
EMGX19	8
EMGX20	8
EMGX21	8
EMGX22	8
EMGX23	8
EMGX24	8
EMGX25	8
EMGX26	8
EMGX27	8
EMGX28	8
EMGX29	8
EMGX30	8
EMGX31	8
EMGX32	8
EMGX33	8
EMGX34	8
EMGX35	8
EMGX36	8
EMGX37	8
EMGX38	8

Figure 19. Alphabetical Deck Name - Link Table (Continued)

FPGX39								8								
FMGX40								8								
FMGX41								8								
FMGX42								8								
FMGX43								8								
FMGX44								8								
FMGX45								8								
FMGX46								8								
FMGX47								8								
FMGX48								8								
FMG1BX								8								
FMG1R								8								
FMG								8								
FMPCOR						6										
FMSG									9							
EMTPRD								8								
EMTRAS								8								
ENCODE																
ENDSSS	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
ENDSYS	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
EOMCKM												12				
EOMCKS												12				
FOMCK												12				
FOMKS												12				
EONUT1												12				
FOSCOD															15	
FOSOF															15	
FOSOUT															15	
FPRMKN															15	
FSFA	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
ESORT	1														15	
ESTOUT					5											
EXFOPR																
EXIN1X															15	
EXIN1															15	
EXIN2F															15	
EXIN2P															15	
EXIN2X															15	
FXI02															15	
EXIN															15	
EXI2															15	
FXLVL															15	
FX02															15	
EXTFRN					5										15	
FACTOR				4	5	6										
FACTRU											11					
FACTRX				4	5	6					11					
FAIKXX											11					

Figure 19. Alphabetical Deck Name - Link Table (Continued)

FA1K						11		
FA1KF						11		
FA1PKA						11		
FA1PKC						11		
FA1PKE						11		
FA1PKI						11		
FA1PKV						11		
FA1PKY						11		
FA1XX						11		
FA1						11		
FA2X						11		
FA2						11		
FBSINT						11		
FRSI	4	5		7	10	11	12	
FBSX	4	5		7		11	12	15
FBS1	4	5		7	10	11	12	15
FBS21						11		
FBS2								15
FBS3	4	5		7	10	11	12	15
FBS4	4	5		7		11	12	15
FRS	4	5		7	10	11	12	15
FCNTL			6					
FCTRUX						11		
FCUPL		5						
FDIT								15
FDNAME								15
FDSUR								15
FDVECT			6					
FEERAA						11		
FEERCX			6					
FFERXC						11		
FEERXX			6					
FEERX			6					
FEERZC						11		
FEERZ1						11		
FEERZ2						11		
FEERZ3						11		
FEERZ4						11		
FEER1X			6					
FFEP1			6					
FFER2X			6					
FEEP2			6					
FFER3X			6					
FEFR3			6					
FEER4X			6					
FEER4			6					
FF100		5						
FILCOR			6					

Figure 19. Alphabetical Deck Name - Link Table (Continued)

FILSWI				4	5	6	7		9	10	11	12			15
FIND		2													
FINDC				4	5	6	7		9	10	11				15
FINDER															15
FLLD									9						
FMDI															15
FNAME	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FNDGRD															15
FNDLVL															15
FNDNYL															15
FNDPAR		2	3	4	5	6	7	8	9	10	11	12	13	14	15
FNDPLT	1	2													
FNDPNT					5										
FNDSFT		2													
FNXTVC															
FNXTV						6									
FNXT						6									
FORFIL		2												14	15
FORMAT							8								
FORMGG				4											
FORMG2				4											
FORM12															
FORMJ											11				
FORM22											11				
FORM2											11				
FPONT					5										
FPT					5										
FOPWV															
FORW						6									
FRBK2						6									
FRBK						6									
FRD2AX						6									
FRD2A										10					
FRD2RX										10					
FRD2B										10					
FRD2CX										10					
FRD2C										10					
FRD2OX										10					
FRD2D										10					
FRD2FX										10					
FRD2E										10					
FRD2I										10					
FRFAD	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FPLG										10					
FRMAX										10					
FRMLTA						6									
FRMLTD						6									
FRMLTX						6									

Figure 19. Alphabetical Deck Name - Link Table (Continued)

FRML T	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FRRDA1						6				10					
FRRDR1										10					
FPRDC1										10					
FRRDC2										10					
FRRDC3										10					
FRRDD1										10					
FRRDD2										10					
FRRDF1										10					
FPRDST										10					
FPRD1A										10					
FPRD1B										10					
FPRD1C										10					
FPRD1D										10					
FPRD1F										10					
FPRD2X										10					
FRRD2										10					
FPRD										10					
FPR1A1										10					
FRSW2						6									
FPSW						6									
FWMW									9						
FZY2									9						
F6211					5								13		
F89					5								13		
GENDSB									9						
GFND									9						
GENELX				4											
GENFLY				4											
GENVEC				4	5	6	7		9	10	11				15
GETRLK															15
GETDEF		2													
GFRSX				4	5		7		9	10	11				15
GFRS				4	5		7		9	10	11				15
GICNM									9						
GIGGKS									9						
GIGGX									9						
GIGTKA									9						
GIGTKG									9						
GINOX	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
GIPSST									9						
GIPSY									9						
GIVM						6									
GI									9						
GKADA1										10					
GKAD1A										10					
GKAD1C										10					
GKAD										10					

Figure 19. Alphabetical Deck Name - Link Table (Continued)

5.4-74 (12/29/78)

GKAM1A										10					
GKAM1R										10					
GKAM1X										10					
GKAM										10					
GMMATC								9		10					
GMMATD		2	3	4	5			8		10			13		
GMMATS		2	3	4	5			8	9	10	11	12	13		15
GMMERG															15
GMPRTN															15
GNFIAT	1														
GNFIST		2	3	4	5	6	7	8	9	10	11	12	13	14	15
GOPFN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
GN									9						
GPA1		2													
GPA2		2													
GPCYCX							7								
GPCYC							7								
GPFORX													13		
GPFDR													13		
GPSP				4											
GPSPA				4											
GPSPP				4											
GPSPC				4											
GPSPO				4											
GPTA1		2	3		5			8				12	13		
GPTLRL		2													
GPTSVM		2													
GPWGA1				4								12			
GPWGR1				4											
GPWG1A				4								12			
GPWG1B				4											
GPWG1C				4											
GPWG				4											
GP1		2													
GP2		2													
GP3A		2													
GP3B		2													
GP3COM		2													
GP3CNR		2													
GP3C		2													
GP3D		2													
GP3		2													
GP4CNR				4											
GP4FIL				4											
GP4PRM				4											
GP4PRT				4											
GP4				4											
GRAV					5										

Figure 19. Alphabetical Deck Name - Link Table (Continued)

NASTRAN - OPERATING SYSTEM INTERFACES

[illegible]

Figure 19. Alphabetical Deck Name - Link Table (Continued)

5.4-76 (12/29/78)

IFP3Z7	1																		
IFP3	1																		
IFP3RD	1																		
IFP3R	1																		
IFP3LV	1																		
IFP4Z7	1																		
IFP4	1																		
IFP4A	1																		
IFP4R	1																		
IFP4C	1																		
IFP4E	1																		
IFP4F	1																		
IFP4G	1																		
IFP5Z7	1																		
IFP5	1																		
IFP5A	1																		
IFP	1																		
IFS1P	1																		
IFS2P	1																		
IFS3P	1																		
IFS4P	1																		
IFS5P	1																		
IFTE2										10									
IFTE4										10									
IFTG										10									
IFTX										10									
IFT										10									
IHEXSD			3		5			8							13				
IHEXSS															13				
IHEX					5														
IHNCSARS													11						
IHNCSAS									9				11						
IHNCSST													11						
IHNFCXPI													11						
IHNFDXPD			3					8					11						
IHNFDXPI	1	2	3		5	6		8					11					15	
IHNFIXPI	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15				
IHNFRXPI		2	3		5	6		8	9		11		13	14					
IHNFRXPR		2	3	4	5	6	7	8	9	10	11	12	13	14	15				
IHNLATN2							7				11								
IHNLFXP			3					8			11								
IHNLLNG			3		5	6		8			11		13						15
IHNLSGN			3				7	8											15
IHNLSORT		2	3	4	5	6	7	8	9	10	11		13						15
IHNLSASN								8					13						
IHNLSATN2		2	3	4	5	6	7	8	9	10	11	12	13	14	15				
IHNSEXP		2	3	4	5	6	7	8	9	10	11	12	13	14	15				
IHNSLNG	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15				

Figure 19. Alphabetical Deck Name - Link Table (Continued)

INQSSCNH											11				
INQSSCN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
INQSSOPT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
INQSTNCT		2	3	4	5	6	7	8	9	10	11	12	13	14	15
INCORE										10	11				
INCRQ									9						
INFRSX						6					11				
INITL2											11				
INITL											11				
INITX											11				
INPTT1		2													
INPTT2		2													
INPTT3		2													
INPTT4		2													
INPUTA		2													
INPUTX		2													
INPUT		2													
INP1XX		2													
INP2XX		2													
INTERB											11				
INTLST		2													
INTPRT								8							
INTVFC		2													
INT2AL															
INVERD		2	3	4	5			8				13			
INVERS				4	5			8				13			
INVERT								8	9	10	11	13		15	
INVFBS						6									
INVPWP						6					11				
INVPWX						6									
INVPXX						6									
INVPX						6									
INVP1X						6									
INVP1						6									
INVP2V						6									
INVP2X						6									
INVP2						6									
INVP3X						6									
INVP4X						6									
INVRTX						6									
INVTRX						6									
INVTR						6									
ISFT	1														
ITCNOE															15
ITEMDT															15
ITMPRT															15
ITTYPE															15
IUNION		2													15

Figure 19. Alphabetical Deck Name - Link Table (Continued)

5.4-78 (12/29/78)

KRAP	3																		
KCONEX	3							8											
KCONEXY	3							8											
KCONEXZ	3							8											
KCONE	3							8											
KDS									9										
KDUM1	3							8											
KDUM2	3							8											
KDUM3	3							8											
KDUM4	3							8											
KDUM5	3							8											
KDUM6	3							8											
KDUM7	3							8											
KDUM8	3							8											
KDUM9	3							8											
KFLAS	3							8											
KFLUD2	3							8											
KFLUD3	3							8											
KFLUD4	3							8											
KHBDY	3							8											
KIHFX	3																		
KORSZ	3	1	2	4	5	6	7	8	9	10	11	12	13	14	15				
KPANFL	3							8											
KPLTST	3							8											
KODMEM	3							8											
KODMM1	3																		
KODM2D	3																		
KODPLT	3							8											
KRON	3																		
KSLNT	3							8											
KSOLID	3							8											
KTETRA	3							8											
KTORDR	3							8											
KTPZ	3																		
KTRAPR	3							8											
KTRBSC	3							8											
KTRIA	3																		
KTRI0D	3							8											
KTRIPG	3							8											
KTRMEM	3							8											
KTRM6D								8											
KTRM6S								8											
KTPPLD								8											
KTRPLS								8											
KTRPLT	3							8											
KTSHLD								8											
KTSHLS								8											
KTUBF	3																		

Figure 19. Alphabetical Deck Name - Link Table (Continued)

LAMXXX									9										
LAMX									9										
LD01	1																		
LD02	1																		
LD03	1																		
LD04	1																		
LD05	1																		
LD06	1																		
LD07	1																		
LD08	1																		
LD09	1																		
LD10	1																		
LD11	1																		
LD12	1																		
LD13	1																		
LD14	1																		
LD15	1																		
LD21	1																		
LD22	1																		
LD23	1																		
LD45	1																		
LD46	1																		
LINFL		2																	
LINF10		2																	
LINE3		2																	
LINE4		2																	
LINE		2																	
LINKNSXX	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15				
LOADS					5														
LOADX					5														
LOADAPP																15			
LOADAPX																15			
LSPLIN									9	10	11								
MABC					5														
MAR			3		5			8					13						
MAGPHA												12	13						
MAKMCB	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15				
MAPENS	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15				
MAPSET		2																	
MASSD			3					8											
MASSTO			3					8											
MATR					5								13						
MATDUM								8											
MATGEN							7												
MATGNX							7												
MATGPR								8											
MATIN		2	3		5			8				12	13						
MATOUT		2	3		5			8				12	13						

Figure 19. Alphabetical Deck Name - Link Table (Continued)

MATPRG				10	
MATPPN					
MATPPT				8	
MATVC2				8	
MATVEC					11
MATWRT					11
MA1XX					
MRAMGX				8	15
MPAHG					
MRAR	3			9	
MBRSLJ					
MRCAP				9	
MRCTR1				9	
MRCTR2				9	
MRDPDH				9	
MRGAF				9	
MRGATE				9	
MRGAU				9	
MRGENO				9	
MRMNOF				9	
MRDXA				9	
MRDNC				9	
MRPLDT				9	
MRPRII				9	
MRREG				9	
MCRAP	3				
MCEA1		4			
MCEB1		4			
MCEC1		4			
MCFD1		4			
MCE1		4			
MCE1A		4			
MCF1R		4			
MCE1C		4			
MCF1D		4			
MCE2		4			
MCONF	3				
MCONMX	3			8	
MCPPO	3			8	
MDUM1	3				
MDUM2	3				
MDUM3	3				
MDUM4	3				
MDUM5	3				
MDUM6	3				
MDUM7	3				
MDUM8	3				
MDUM9	3				

Figure 19. Alphabetical Deck Name - Link Table (Continued)

5.4-81 (12/29/78)

MFRGFD								9		11					
MFRGF1						7									
MFRGF					5	6	7	9	10	11	12			15	
MESSAGE	1	2	3	4	5	6	7	9	10	11	12	13	14	15	
MFLUD2			3					8							
MFLUD3			3					8							
MFLUD4			3					8							
MREF			3					8							
MHROY			3					8							
MIHEX			3												
MINMAX		2													
MINTRP									10	11					
MINV			3		5			8				13			
MODA							7								
MODACC											12				
MODACX											12				
MODAC1											12				
MODAC2											12				
MODAC3											12				
MODR							7								
MODC							7								
MODDMP	1														
MODEL	1														
MPLPRT	1														
MPRTX								8							
MPYADX				4	5	6	7		10	11	12			15	
MPYADZ				4	5	6	7	9	10	11	12			15	
MPYAD				4	5	6	7	9	10	11	12			15	
MPYA1D							7								
MPYA2D							7								
MPYA3D							7								
MPYL					5										
MPYO				4	5	6	7	9	10	11	12			15	
MPY3A							7							15	
MPY3R							7							15	
MPY3C							7							15	
MPY3DR							7							15	
MPY3NU							7							15	
MPY3NC							7							15	
MPY3P							7							15	
MPY3TL							7							15	
MPY3Z7							7							15	
MPY3							7								
MPY3CP							7							15	
MPY3IC							7							15	
MPZDA			3												
MODPLT			3												
MREDZZ								8							15

Figure 19. Alphabetical Deck Name - Link Table (Continued)

MPED1A																			15
MPED1R																			15
MPED1C																			15
MPED1D																			15
MPED1E																			15
MPED1																			15
MPED2A																			15
MPED2R																			15
MPED2C																			15
MPED2D																			15
MPED2F																			15
MPED2F																			15
MPED2G																			15
MPED2H																			15
MPED2I																			15
MPED2J																			15
MPED2L																			15
MPED2M																			15
MPED2N																			15
MPED2O																			15
MPED2P																			15
MPED2Z																			15
MPED2																			15
MPGE								8											
MRGFDX									9			11							
MPING			3					8											
MROD			3																
MSB																			
MSGCOM	1	2	3	4	5	6	7	8	9	10	11	12	13						15
MSGWRT	1	2	3	4	5	6	7	8	9	10	11	12	13	14					15
MSGX	1	2	3	4	5	6	7	8	9	10	11	12	13	14					15
MSLOT			3					8											
MSQLID			3					8											
MSTRIA			3																
MTIMSU						6													
MTMSU1						6													
MTORDR			3					8											
MTRAPR			3					8											
MTPRSC			3					8											
MTRI00			3					8											
MTRIRG			3					8											
MTRPLT			3					8											
MTRXIN											10								
MTRXIX																			15
MTRYI																			15
MTRX0X																			15
MTRX0																			15
MTRXXX										10									

Figure 19. Alphabetical Deck Name - Link Table (Continued)

MTURE			3												
MXCID5							7								
MXCID							7								
NAMES	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
NASCAP	1						7					12			
NORM11						6									
NORM1						6									
NTIMEX	1														
NTIME	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
QCTBCD											11				
QDUM1											11				
QDUM2														14	
QDUM3														14	
QDUM4														14	
QDUM5														14	
QDUM6														14	
QDUM7														14	
QDUM8														14	
QDUM9														14	
QDUM														14	
QFPR9														14	
QFPRD1														14	
QFPRD5														14	
QFPR1														14	
QFPR2														14	
QFPR35														14	
QFPR3														14	
QFPR4														14	
QFPR5														14	
QFPR6														14	
QFPRB7S														14	
QFPRB7														14	
QFPRB8														14	
QFPCF1														14	
QFPCF2														14	
QFPCPM														14	
QFPCS1														14	
QFPCS2														14	
QFPMIS														14	
QFPPNT														14	
QFPPUN														14	
QFPRF1														14	
QFPRF2														14	
QFPRS1														14	
QFPRS2														14	
QFPSN1														14	
QFPSS1														14	
QFPXXX														14	

Figure 19. Alphabetical Deck Name - Link Table (Continued)

OFPI																	14
OFPIA																	14
OFPI																	14
OFRF2S																	14
OFRS2S																	14
OFSEN1																	14
OFSS1																	14
ONETWO				4	5	6	7			10	11						14
OPINV						6											14
OPMFSG		2															14
OPTPR1		2															
OPTPR2								8									
OPTPW1		2															
OPTPW2								8									
OPTPX1		2															
OPTPX		2															
OPTP1A		2															
OPTP1B		2															
OPTP1C		2															
OPTP1D		2															
OPT2A								8									
OPT2B								8									
OPT2C								8									
OPT2D								8									
ORDER		2															
ORTCK						6											
ORTHO											11						
OSCENT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		
OSCXRF	1																
OUTPT1																	14
OUTPT2																	14
OUTPT3																	14
OUTPT4																	14
OUTPT																	14
OUTPUT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		
OUT1XX																	14
OUT2XX																	14
OUT3XX																	14
PABS											11						
PACKX	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		
PAGF	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		
PARAML		2	3	4	5	6	7	8	9	10	11	12	13	14	15		
PARAM		2															
PARMEG				4	5	6	7		9	10	11	12					15
PARMLX		2	3	4	5	6	7	8	9	10	11	12	13	14	15		
PARTN1							7										
PARTN2							7										
PARTN3							7										

Figure 19. Alphabetical Deck Name - Link Table (Continued)

PARTN				4	5		7		9	10	11	12			15
PASSFR	1														
PATX				4	5	6	7		9	10	11	12			15
PERMUT					5										
PERPEC		2													
PEXIT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PHDMIA														14	
PHDMTX														14	
PKRAR															
PKOAN1													13		
PKOAN2													13		
PKODMS													13		
PKQDM1													13		
PKODM													13		
PKQDPL													13		
PKRQD													13		
PKT01													13		
PKT02													13		
PKTRBS													13		
PKTRI1													13		
PKTRI2													13		
PKTRMS													13		
PKTRM1													13		
PKTRM													13		
PKTRPL													13		
PKTRQD													13		
PKTRQ2													13		
PLAGP													13		
PLAMAT													13		
PLA1			3												
PLA2X													13		
PLA2													13		
PLA3ES													13		
PLA3UV													13		
PLA31X													13		
PLA31													13		
PLA32C													13		
PLA32F													13		
PLA32S													13		
PLA32X													13		
PLA32													13		
PLA3													13		
PLA4R													13		
PLA4ES													13		
PLA41IV													13		
PLA41X													13		
PLA41													13		
PLA42C													13		

Figure 19. Alphabetical Deck Name - Link Table (Continued)

PLA42D																	13
PLA42E																	13
PLA42S																	13
PLA42X																	13
PLA42																	13
PLA4																	13
PLQAD3					5												
PLQAD					5												
PLQT		2															
PLTDAT		2															
PLTMGX																	15
PLTMRG																	15
PLTOPR		2															
PLTRN1								8									
PLTSCR		2															
PLTSFT		2															
PLTTRA								8									
PREFIX																	15
PPELOC	1	2	3	4	5	6	7	8	9	10	11	12	13	14			15
PREMAT		2	3		5			8				12	13				
PRESAX					5												
PRFTAR			3		5			8		10	11		13	14			15
PRETRD		2	3	4				8					13				
PRFTRS			3	4	5			8	9			12	13				15
PRINT		2															
PROCFS		2															
PRINT								8									
PRTMRG							7										
PRTMSC		2															
PRTPRM								8									
PSRAR																	
PSQAD1													13				
PSQAD2													13				
PSQDM1													13				
PSQDM													13				
PSQPL1													13				
PSRND													13				
PSTA																	
PSTAMG									9								
PSTONC									9								
PSTONX									9								
PSTPL1																	
PST01													13				
PST02													13				
PSTRB1													13				
PSTR11													13				
PSTR12													13				
PSTRM1													13				

Figure 19. Alphabetical Deck Name - Link Table (Continued)

PSTPM													13		
PSTPQ2													13		
PTMGZZ							7								
PULL	1														
PUSH	1													15	
PVECA									9						
PVECA									9						
PVECKX									9						
PVECK									9						
PVECONM									9						
PVECPY									9						
PVECP									9						
PVECSX									9						
PVECS									9						
PVECO5									9						
PVEC10									9						
PVEC20									9						
PVEC									9						
PXIT36	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
QDMEM					5										
QDMM10								8							
QDMM15								8							
QDMM1					5										
QDMM20								8							
QDMM25								8							
QDMM2					5										
QDPLT					5										
QHRDY					5										
QLOADL					5										
OPARAM		2	3	4	5	6	7	8	9	10	11	12	13	14	15
OPARMR		2	3	4	5	6	7	8	9	10	11	12	13	14	15
ORITER						6									
OVECT					5										
OVNL					5										
Q2BCD			3												
Q2BCS					5								13		
Q2TRMD			3												
Q2TRMS					5								13		
RANDMX														14	
RANDNM														14	
RAND1														14	
RAND2														14	
RAND3														14	
RAND5														14	
RAND6														14	
RAND7														14	
RAND8														14	
RRMG1						4								14	

Figure 19. Alphabetical Deck Name - Link Table (Continued)

5.4-88 (12/29/78)

(Handwritten signature)

RRMG2	4		
RRMG3	4		
RRMG4	4		
RCARO	1		
RCNVAX			15
RCOVA			15
RCNVR			15
PCNVCM			15
RCNVCR			15
RCNVCX			15
PCNVC			15
RCNVDS			15
RCOVEM			15
RCOVEX			15
RCNVE			15
PCNVIM			15
RCNVLS			15
PCNVMS			15
PCNVNX			15
PCOVN			15
RCOVOV			15
PCNVR3			15
PCNVR			15
PCNVSL			15
PCNVSS			15
RCNVUI			15
RCOVUO			15
RCNVVA			15
PCNV3X			15
PDMNDX	2		
PEAD1A		6	
PEAD1		6	
PEAD2A		6	
PEAD2		6	
PEAD3		6	
PEAD4		6	
PEAD6X		6	
PEAD6		6	
PEAD7		6	
PFDU	1		
PEDUCE			15
PEDZZ7			15
REGEAN		6	
REIG		6	
PEIGA		6	
PEIGKR		6	
RENAME			15
PETBLK			15

Figure 19. Alphabetical Deck Name - Link Table (Continued)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RETURN															
RF2AL								8			11				
RFORCE					5										
RMG777					5										
RMG					5										
RPODD								8							
RPODS								8							
RPOD					5										
RPM8DK			3					8							
RPM8ER					5								13		
ROTAT		2													
ROTAX						6									
POWDYZ									9						
RSORT											11				
RSTXXX		2													
RHLEP				4	5	6	7		9	10	11	12			15
SADDX				4		6	7		9	10					15
SADD				4		6	7		9	10	11				15
SADOTB			3		5			8	9				13		
SAXB			3		5			8	9				13		
SAXIF1													13		
SAXIF2													13		
SBAR1													13		
SBAR2													13		
SBSPL2													13		
SCALAR															15
SCALED								8							
SCALEX				4											
SCALXX															15
SCE1				4											
SCONF1													13		
SCONF2													13		
SCONF3													13		
SCRLM													13		
SDCTNS							7								
SDCTN				4	5	6	7			10	11				15
SDCMMX							7								
SDCMM							7								
SDCMP5							7								
SDCMO							7								
SDCOMP				4	5	6	7			10	11				15
SDCOMX				4	5	6	7			10	11				15
SDCOM1				4	5	6	7				11				15
SDCOM2										10					15
SDCOM3				4	5	6	7			10	11				15
SDCOM4				4	5	6	7			10	11				15
SDCOM				4	5	6	7			10	11				
SDCOUT				4	5	6	7			10	11				15

Figure 19. Alphabetical Deck Name - Link Table (Continued)

SDHTFF					13	
SDHTF1					13	
SDHTF2					13	
SDRA1					12	
SDRA2						13
SDRP1	5	6	7		12	
SDRCHK						13
SDRC2						13
SDRFTD						13
SDRFTT						13
SDRHTZ						13
SDRHT						13
SDR1					12	
SDR1A					12	
SDR1R	5	6	7		12	
SDR2AA						13
SDR2A						13
SDR2R						13
SDR2C						13
SDR2DE						13
SDR2D						13
SDR2E						13
SDR2XX						13
SDR2X1						13
SDR2X2						13
SDR2X4						13
SDR2X5						13
SDR2X6						13
SDR2X7						13
SDR2X8						13
SDR2X9						13
SDR2						13
SDR37Z						13
SDR3						14
SDR3A						14
SDUM11						14
SDUM12					13	
SDUM21					13	
SDUM22					13	
SDUM31					13	
SDUM32					13	
SDUM41					13	
SDUM42					13	
SDUM51					13	
SDUM52					13	
SDUM61					13	
SDUM62					13	
SDUM71					13	

Figure 19. Alphabetical Deck Name - Link Table (Continued)

SDUM72																		13
SDUM81																		13
SDUM82																		13
SDUM91																		13
SDUM92																		13
SD2RHD																		13
SFFMAT		2																13
SFFMTX		2																
SFLAS1																		
SFLAS2																		13
SFLCAM		2																13
SFMINT	1																	
SFMRN	1																	
SFM	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15			
SETEO																		15
SETEND												12	13					15
SETINP		2																15
SFTLVL																		
SETTIM	1																	15
SFTUP	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15			
SETVAL		2	3	4	5	6	7	8	9	10	11	12	13	14	15			
SFACT				4	5	6	7											
SFFTCH												11						15
SGENZZ																		15
SGEN																		15
SGFNA																		15
SGENR																		15
SGENM																		15
SGINN1		2																15
SGINN		2																
SHAPE		2																
SHEARD								8										
SHEARS								8										
SICOX						6												
SIHFX1																		
SIHFX2													13					
SJUMP													13					
SKPERM		2																15
SKPREC	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15			
SMAR3				4														
SMAC3				4														
SMA1X			3															
SMA1			3															
SMA1A			3															
SMA1RK			3					8										
SMA1R			3															
SMA1CL			3					8										
SMA1OP			3					8										

Figure 19. Alphabetical Deck Name - Link Table (Continued)

SMA1ET			3						8								
SMA1HT			3						8								
SMA1IO			3						8								
SMA2X			3						8								
SMA2			3														
SMA2A			3														
SMA2BK			3														
SMA2R			3						8								
SMA2CL			3														
SMA2DP			3						8								
SMA2ET			3						8								
SMA2HT			3						8								
SMA2IO			3						8								
SMA3				4													
SMA3A				4													
SMA3B				4													
SMA3C				4													
SMLFIG						6											
SMMATS													13				
SMPYAD							7										
SMP1				4													
SMP2				4													
SMSG																	
SNPDF																	15
SNFCLS									9								
SNFCOM	1	2	3	4	5	6	7	8	9	10	11	12	13	14			15
SNFINT																	15
SNFIOF																	15
SNFINI	1																15
SNFIN																	
SNFI																	15
SNFOPN																	15
SNFN																	15
SNFS17																	15
SNFTOC																	15
SNFTRL																	15
SNFUTX																	15
SNFUT																	15
SNF																	15
SOLID						5											15
SOLVER				4													
SOLVF1																	
SOLVE													13				
SOLVRX									7								
SOLV1X				4													
SOLV2X									7								
SOLV3X									7								
SOLV4X									7								

Figure 19. Alphabetical Deck Name - Link Table (Continued)

5.4-93 (12/29/78)

[Handwritten signature]

SOLV5X							7								
SPRT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SORTCM															15
SOUT													13		
SPANL1													13		
SPANL2													13		
SPLT10															15
SODME1													13		
SODM11													13		
SODM12													13		
SODM21													13		
SODM22													13		
SODPL1													13		
SCRTM						6									
SPND1													13		
SPND2													13		
SSGA1X					5										
SSGA2					5				9	10	11	12			
SSGA3			4		5							12			
SSG81X					5										
SSG82			4		5	6	7		9	10	11	12			15
SSGC2			4						9	10	11				15
SSGFTD					5										
SSGETT					5										
SSGHTP					5										
SSGHTZ					5										
SSGHT1					5										
SSGHT2					5										
SSGHT					5										
SSGKHI					5										
SSGSLT					5										
SSGTRI					5										
SSGWRK					5										
SSG1					5										
SSG1A					5										
SSG2A					5				9	10	11	12			
SSG2Y					5										
SSG2					5										
SSG2P			4		5	6	7		9	10	11	12			15
SSG2C			4						9	10	11				15
SSG3					5										
SSG3A			4		5										
SSG4					5							12			
SSLDT1													13		
SSLDT2													13		
SSOLD1													13		
SSOLD2													13		
SSPLIN									9		11		13		

Figure 19. Alphabetical Deck Name - Link Table (Continued)

5.4-94 (12/29/78)

SSWTCB	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
STAPID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
STFP2											11				
STEP											11				
STIME	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
STOR01													13		
STOR02													13		
STPAIC									9						
STPAX1													13		
STPAX2													13		
STPAX3													13		
STPRG									9						
STPRS0									9						
STPRS1									9						
STPDA									9						
STPK									9						
STPLNT		2													
STPPHI									9						
STPPT2									9						
STOME2													13		
STRAP1													13		
STRAP2													13		
STRAX1													13		
STPAX2													13		
STRAX3													13		
STRPS1													13		
STRIPC									9						
STRIPX									9						
STRIR1													13		
STRIR2													13		
STRMF1													13		
STRM61													13		
STRM62													13		
STRPL1													13		
STRPTS													13		
STRP11													13		
STRP12													13		
STR001													13		
STR002													13		
STRSLV													13		
STRSL1													13		
STRSL2													13		
STUBF1													13		
SUBA									9						
SUBI									9						
SUBP2															
SUBP									9						
SUBPA									9						

Figure 19. Alphabetical Deck Name - Link Table (Continued)

NAME	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SUADPH1															
SUP1						6									15
SUP						6									
SUMM						6									
SUMPH1															
SUPLT		2							9						
SUREAD															
SUWRT															15
SWITCH							7								15
SWSPT	1														
SYMBLS		2													
SYMBOL		2													
SYSTEM	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SYS															
TAA1		2													15
TAA2		2													
TAREMT								8							
TARFTX								8							
TARFTZ								8							
TARPCB									9						
TARPHX									9						
TARPRF									9						
TARPRT															
TARPEX								8						14	15
TARPT								8						14	15
TAC1		2						8							
TAC1AX		2													
TAPBIT	1	2													
TAPSWI		2												14	15
TAL1A		2												14	
TAL1AR		2													
TAL1ACM		2													
TAL1B		2													
TAL1C		2													
TAL1CA		2													
TAL1ETD		2													
TAL1ETT		2													
TAL1H		2													
TAL1		2													
TETPA					5										
TIMAL3									9						
TIMEEO			4	5	6	7			9	10	11				15
TIMTST									9						
TIMTS1									9						
TIMTS2									9						
TIMTS3									9						
TIMTS4									9						
TIMTS5									9						

Figure 19. Alphabetical Deck Name - Link Table (Continued)

TIMTS6									9								
TIMTS7									9								
TIMTS8									9								
TIM1XX									9								
TIM2XX									9								
TIM3XX									9								
TIM4XX									9								
TIM5XX									9								
TIM6XX									9								
TIM7XX									9								
TIM8XX									9								
TIPF		2															
TKER									9								
TKT7TK			3										13				
TLDDM6					5												
TLDDSL					5												
TLDDT1					5												
TLDDT2					5												
TLDDT3					5												
TMTGGD	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		
TPREPX									9								
TPSWIT		2												14			
TPZTEM					5												
TPAILE									9								
TPANFM																	
TRANPJ				4	5	6			9				13				
TRANSP				4	5	6	7										
TEANX					5					10	11						
TRAPAD								8									
TRAPAX								8									
TFBSCD								8									
TPBSCS								8									
TRBSC					5												
TRDA1											11						
TFDC1											11						
TRDD1											11						
TPDF1											11						
TRDXX											11						
TRD1A2											11						
TRD1A											11						
TRD1C2											11						
TRD1C											11						
TRD1C2											11						
TRD1D											11						
TRD1E											11						
TRD1X											11						
TRD											11						
TRHTX											11						

Figure 19. Alphabetical Deck Name - Link Table (Continued)

TPHT1A										11						
TPHT1B										11						
TPHT1C										11						
TPHT										11						
TRIAAD								8								
TPIAAX								8								
TPIDI						6										
TPIMEM				5												
TPJMFY				5												
TPIOD				5												
TPFLGA				5												
TPLGIC				5												
TPLG				5												
TPLGA				5												
TPLGC				5												
TPLCD				5												
TRMEMD								8								
TRMEMS								8								
TRNSPX			4	5		6	7		9						15	
TRNSP			4	5		6	7		9						15	
TPPLT				5												
TRTTEM				5												
TSPL1D								8								
TSPL1S								8								
TSPL2D								8								
TSPL2S								8								
TSPL3D								8								
TSPL3S								8								
TTLPGE	1															
TTORDR				5												
TTAPR				5												
TTIRG				5												
TIURED								8								
TIURFS								8								
TVNR									9							
TWISTD								8								
TWISTS								8								
TWO	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
TYPE10		2														
TYPE3		2														
TYPE	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
TYPEFLT		2														
TYPEINT		2														
TYPEOUT		2														
UMFEND	1													14		
UMFYXX	1															
UMFZZZ	1															
UNPAKY	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	

Figure 19. Alphabetical Deck Name - Link Table (Continued)

5.4-98 (12/29/78)

IPARTY				4			7			10					
UPART				4			7			10					
USRMSG	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VALVEC						6									
VDRA												12			
VDRB												12			
VDRCON												12			
VDRCONR												12			
VDR												12			
VECPRT								8							
VECXXX							7								
VEC							7								
VISCN								8							
WALTIM	1														15
WILVEC						6									
WPLT10		2													
WPLT3		2													
WPLT4		2													
WRMSG		2													
WRTPT		2													15
WRTTRL	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XCFITR	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XCFI	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XCHK	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XCLEAN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XCSA	1														
XCSABF	1														
XCSM					5										
YDPH	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
YDPL	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XFADJ1	1														
XF1AT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XF1ST	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XFLDEF	1														
XFLORD	1														
XFLS7D	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XGP1AS	1														
XGP1C	1														
XGP1DG	1														
XGP1D	1														
XGP11	1														
XGP12X	1														
XGP12	1														
XGP13	1														
XGP14	1														
XGP15	1														
XGP16	1														
XGP17	1														

Figure 19. Alphabetical Deck Name - Link Table (Continued)

YGPIR	1														
YGPI	1														
XIHFY							8								
XIPFL	1														
XLINK	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
YLKSPC	1														
XLNKHD	1														
XMDMSK	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
YNSTRN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
YNLDPT	1														
XDSGFN	1														
XPAPAM	1														
XPFIIST	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XPOLCK	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XPUMP	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XPURGF	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XPCARD	1														
XPECPS	1														
XRGDFM	1														
XSAVE		2	3	4	5	6	7	8	9	10	11	12	13	14	15
XSRSET	1														
YSCNDM	1														
YSEMO1	1														
XSEM02		2													
YSEMO3			3												
XSEM04				4											
XSEM05					5										
XSEM06						6									
XSEM07							7								
XSEM08								8							
XSEM09									9						
XSEM10										10					
XSEM11											11				
XSEM12												12			
YSEM13													13		
XSEM14														14	
XSEM15															15
XSFA1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XSFA		2	3	4	5	6	7	8	9	10	11	12	13	14	15
XSNPT	1														
XDSGN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XSRTBD	1														
XTPNSY						6									
XTRNY1						6									
XVPS	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XYFIAT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XXMPRT								8							
XXOPT1		2													

Figure 19. Alphabetical Deck Name - Link Table (Continued)

XYOPTZ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XXPARM		2						8							
XXPLINT		2													
XXPMSC		2													
XXPRTI								8							
XXPSET		2													
XXVLVC						6									
XYCHAR															
XYDUMP															14
XYFIND															14
XYGRAF															14
XYLONG															14
XYOUT															14
XYPLIN		2													14
XYPLINT		2													
XYPLXX		2													
XYPPPP															
XYPRPL															14
XYPRPT															14
XYTICS															14
XYTRAN															14
XYTRZZ															14
XYWRK															14
ZAPD															14
ZPLPKX	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ZFRNC									9	10	11				
ZJ									9						
ZNTPKX	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Figure 19. Alphabetical Deck Name - Link Table (Continued)

5.4-101 (12/29/78)

5.5 NASTRAN ON THE CDC 6000/CYBER (NØS AND NØS/BE)

5.5.1 Introduction

NASTRAN operates as 15 separate programs, or links, on the CDC 6000/CYBER computers under the NØS and NØS/BE operating systems. NASTRAN is created using the Segmentation Loader, which is operating system independent. Each link is segmented to minimize central memory requirements. A small resident loader loads segments dynamically.

Link switching is performed using the TCS (Translate Control Statement) macro on NØS, and CCL (CYBER Control Language) under NØS/BE. This, and differences in library formats between CDC operating systems, necessitate different control card setups for NØS and NØS/BE. NØS 1.3 and NØS/BE 1.3 require minor modifications to the link switching code. (See Section 5.5.3)

All matrix operations are done in single precision (60 bit) arithmetic. Double precision is not supported.

5.5.2 Input/Output

NASTRAN input/output is performed in four ways: (1) card input, printed output, punched output, and binary output generated from some of the utility modules use standard FØRTRAN formatted input and output statements; (2) all other input/output (except plots and substructure I/O) uses the GINØ routines which in turn call IØ6600 (see Section 5.5.6) to perform reads and writes; (3) plot output uses SGINØ (see Section 3.4.51); and (4) substructuring I/O uses SØFIØ (see Section 3.6.29).

The number of files available to GINØ is a function of the MAXFILES parameter in the SYSTEM common block (see Section 2.4.1.8). All NASTRAN files which have not been assigned to tape are assigned to disk as computed index files. Disk file relative PRU addresses are computed in IØ6600 using the GINO block number and block size. Pointers to the current and largest disk address are maintained in common block /GINØX/. While the file is open word 13 of the File Environment Table (FET) is used as a pseudo-index for system communication. Actual input/output is accomplished through XIØRTNS (see Section 5.5.6), which is a machine dependent FØRTRAN subprogram. XIØRTNS communicates with NØS and NØS/BE through subroutine PPCALL (see Section 5.5.6), which issues requests to Combined Input/Output (CIØ).

NASTRAN - OPERATING SYSTEM INTERFACES

SGINØ on the CDC functions independently of other I/O. Its point of commonality with other I/O routines is XIØRTNS, which is used to perform the physical writing of data.

Figure 1 depicts a complete buffer as used by GINØ, IØ6600 and XIØRTNS.

5.5.3 NASTRAN on the Segmentation Loader

5.5.3.1 Overview of the Segmentation Loader

Level 17.5 NASTRAN uses the Segmentation Loader, a CDC product which provides most of the capabilities needed to structure NASTRAN for efficient operation in a relatively small memory region. The Segmentation Loader replaces the NASTRAN Linkage Editor, which is no longer maintained. The Segmentation Loader and its usage are described in detail in the CDC CYBER Loader Reference Manual, CDC Publication Number 60429800.

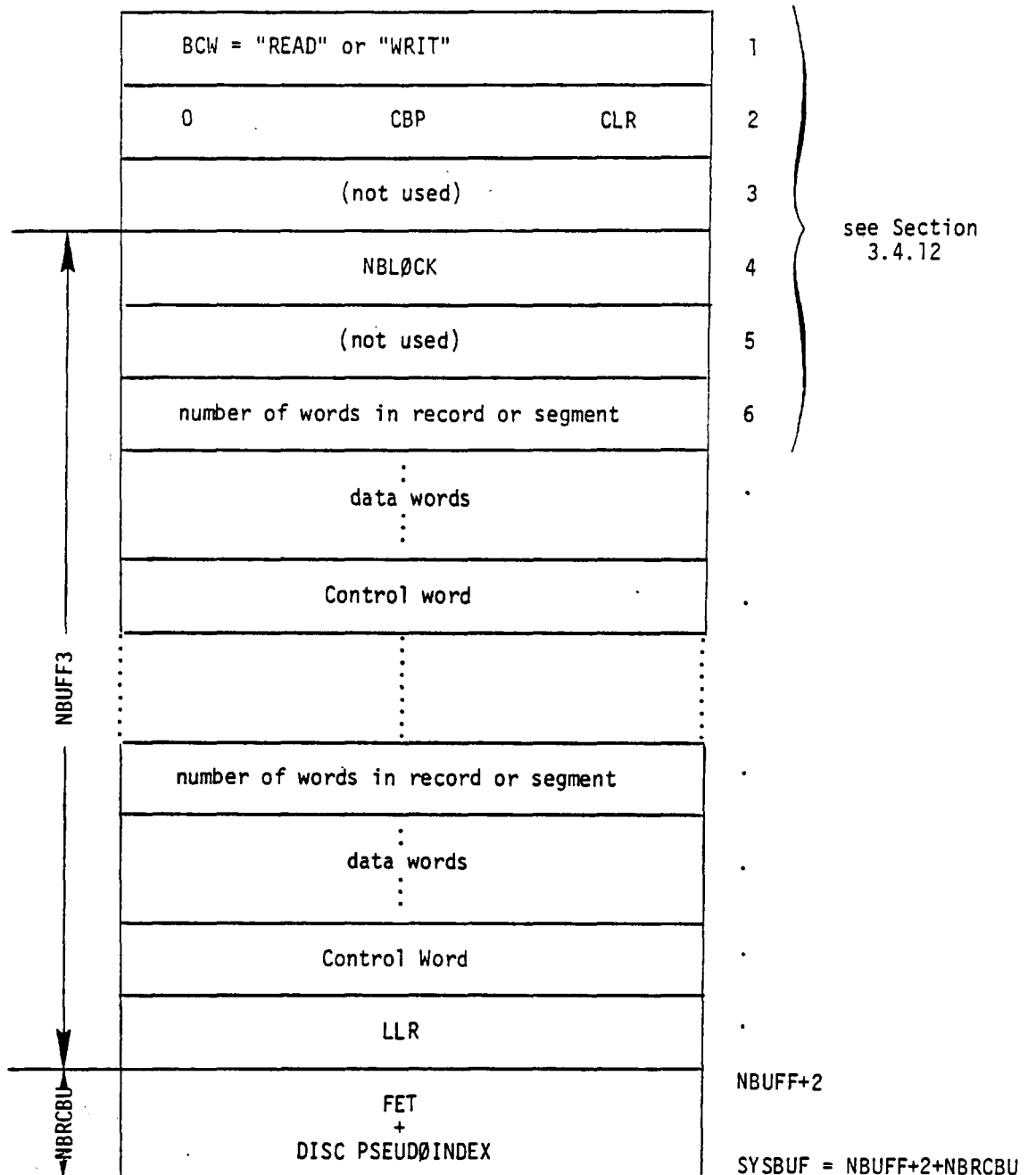
The Segmentation Loader (not to be confused with the CDC SCØPE Segment Loader) shares the following features with the NASTRAN Linkage Editor.

- The segmentation structure is defined at program generation time by user directives.
- Segments are loaded dynamically by a call to any entry point within the segment.

The Segmentation Loader has the following restrictions which affect NASTRAN.

- The Segmentation Loader cannot load all of NASTRAN at once. Therefore NASTRAN has been broken down into fifteen separate programs. The programs correspond to links one through fifteen of Level 17 NASTRAN.
- Link switching within NASTRAN now involves program chaining as currently done in the Level 17 UNIVAC version of NASTRAN. No practical method for program chaining could be found that would work on both the NØS and NØS/BE operating systems. Therefore two separate CDC NASTRAN control card setups must be maintained.
- Loading NASTRAN as 15 programs requires that the object (relocatable) decks be in library format. Every subprogram to be included in the library must have an entry point table. The FTN compiler generates entry point tables for all subprograms except block data subprograms. A special utility was written to add an entry point table to block data subprograms.
- RENAME of entry points must be done by job control cards rather than by loader directives.

NASTRAN ON THE CDC 6000/CYBER



NBUFF3 = NBUFF-2

NBUFF = SYSBUF-2-NBRCBU

SYSBUF and NBRCBU are defined in the SYSTEM common block (see Section 2.4.1.8)

Figure 1. GINØ buffer on the CDC 6000/CYBER.

5.5.3.2 Using the Segmentation Loader in the NASTRAN Environment

An overlay structure has been developed for NASTRAN on the Segmentation Loader which closely parallels the Linkage Editor overlay structure. However, there are significant differences in terminology, organization and formatting of directives, and organization of the Update Program Library containing the directives.

To illustrate this, a typical tree (overlay) structure has been abstracted from Link 4; the structure and directive cards to create it are shown in Figure 2. Segments are created using INCLUDE cards, and organized using TREE directives. Segment names may be subprogram names. Common blocks that are referenced from more than one segment must be declared GLOBAL. The first field of the GLOBAL directive defines the owning segment; only this segment may preset the referenced common via a block data subprogram or DATA statements. Open core is specified by placing an open core common block in its own segment. The LEVEL directive serves the same purpose as the Linkage Editor REGION statement.

Tree directives use commas to separate parallel segments that cannot coexist in core, and dashes for segments that can coexist. Tree directives may be labeled, and trees may consist of subprograms, segments and other trees.

The segmentation directives need not list all subprograms and common blocks. A common block not specified as global will be linked only to subprograms within each segment it occurs in and is not available to subprograms in other segments. Also a program not specified on a directive will be placed by the loader in the lowest level segment accessible to all segments referencing it.

This latter feature of the loader is very convenient, but can lead to inefficient memory utilization. Consider the tree structure of Figure 3. If subprogram z is called by segments g and h, but z is not mentioned on a directive, it will "float" to segment d. But if z is also called from segment l, it will float to segment a. Now, if the chain a-b-e-j is the longest path, memory requirements can be reduced by explicitly including z in segments d and f. This behaviour cropped up frequently during conversion of NASTRAN to the Segmentation Loader, and as a result the INCLUDE directives are often more explicit than may seem necessary.

Figure 4 shows an actual NASTRAN Segmentation Loader directive stream in Update format. The decks EXECMOD and EXCTREE are segments and trees for the NASTRAN executive and will not normally be modified by the user.

Figure 5 illustrates a subtle segmentation error which was acceptable to the NASTRAN Linkage Editor. It is desired that segments a, b, and c be callable from d and e. Initially, segment d is called from the root, and d calls a, which is not in its call chain. But upon completion, a returns control directly to d, and d returns control to the root successfully. Then e calls c, and c calls f, another subroutine in segment e. The call from c to f causes segment e to be reloaded, and the return address from e to the root is destroyed. This problem can be resolved only by such means as combining segments a, b, and c, or moving f to the root or to a segment common to a, b, and c.

5.5.3.3 Using NASTRAN with the Segmentation Loader

The following examples describe the use of NASTRAN with the Segmentation Loader. Actual control card setups are shown in Section 5.5.5.

1. Execution from Absolute Files - The control cards required to execute the Segmentation Loader version of NASTRAN under NOS are shown in example 1 of Section 5.5.5. These control cards access the fifteen absolute (executable) links and execute LINK1. The other links are executed dynamically as needed.
2. Modification of Source Code and Segment Structure - Figure 6 shows the control cards required to make a source code alter run on the NOS operating system. This job may be considered as four steps. The first two steps process source code and segment alters, and create the files needed by the Segmentation Loader. The third step invokes the loader to create new executables for those links being modified. The final step invokes execution.

The first step is very much like a linkage editor alter run except that it produces an object library for input to the Segmentation Loader. Note that Block Data subprograms require exceptional handling. If an object deck file is used as direct input to the Segmentation Loader, all common blocks in block data routines not explicitly named on Segmentation Loader directives are floated to the root. This problem could be avoided by setting up in each link a dummy tree consisting only of block data routines not used in that link. This would create maintenance problems and significantly increase the disc space needed to store executables.

The following alternative was implemented for NASTRAN. First, all block data routines are named; e.g., BLOCK DATA SEMDBD. Then after compilation the object decks are passed

NASTRAN - OPERATING SYSTEM INTERFACES

through the NASTRAN utility BLKENTR (Figure 7) which inserts entry point tables in the Block Data routines. The modified object deck file can now be merged into the old object deck file and the result processed into a library using the LIBGEN utility.

Note that the NX (no cross-reference) parameter on the LIBGEN card must be turned-off (set non-zero). The Segmentation Loader creates its own cross-reference table, and may not properly process the library cross-reference table.

The next steps are to update the link overlay structure and to load the links modified. These steps must be planned in parallel, because the link update step writes each modified link directive list as a record on the SUBSYS file. Each SEGLØAD control card reads one record from this file and creates a tree structure table. The associated LIBLØAD control card fills in the tree structure with NASTRAN object decks. Thus there must be a one-to-one relationship between updated links and segment loads, or erroneous loads will result. If links 1, 3 and 8 are updated, links 1, 3 and 8 must be loaded; no more, no less.

Each link to be loaded requires, at a minimum, a SEGLØAD card, a LIBLØAD* card, and a NØGØ card. The SEGLØAD card specifies the file to which the executable is to be written. The LIBLØAD card specifies which of the 15 main programs, NAST01 to NAST15, is to be loaded from NASTLIB to begin filling in the tree structure. The NØGØ card causes circular search of all libraries, starting with NASTLIB, to satisfy externals.

Link 8 requires a special load procedure, shown in Section 5.5.5, to effect renaming of non-existent element routines to EMGØLD.

The card LINK1, INPUT. starts execution of NASTRAN.

*SLØAD for NØS/BE

NASTRAN ON THE CDC 6000/CYBER

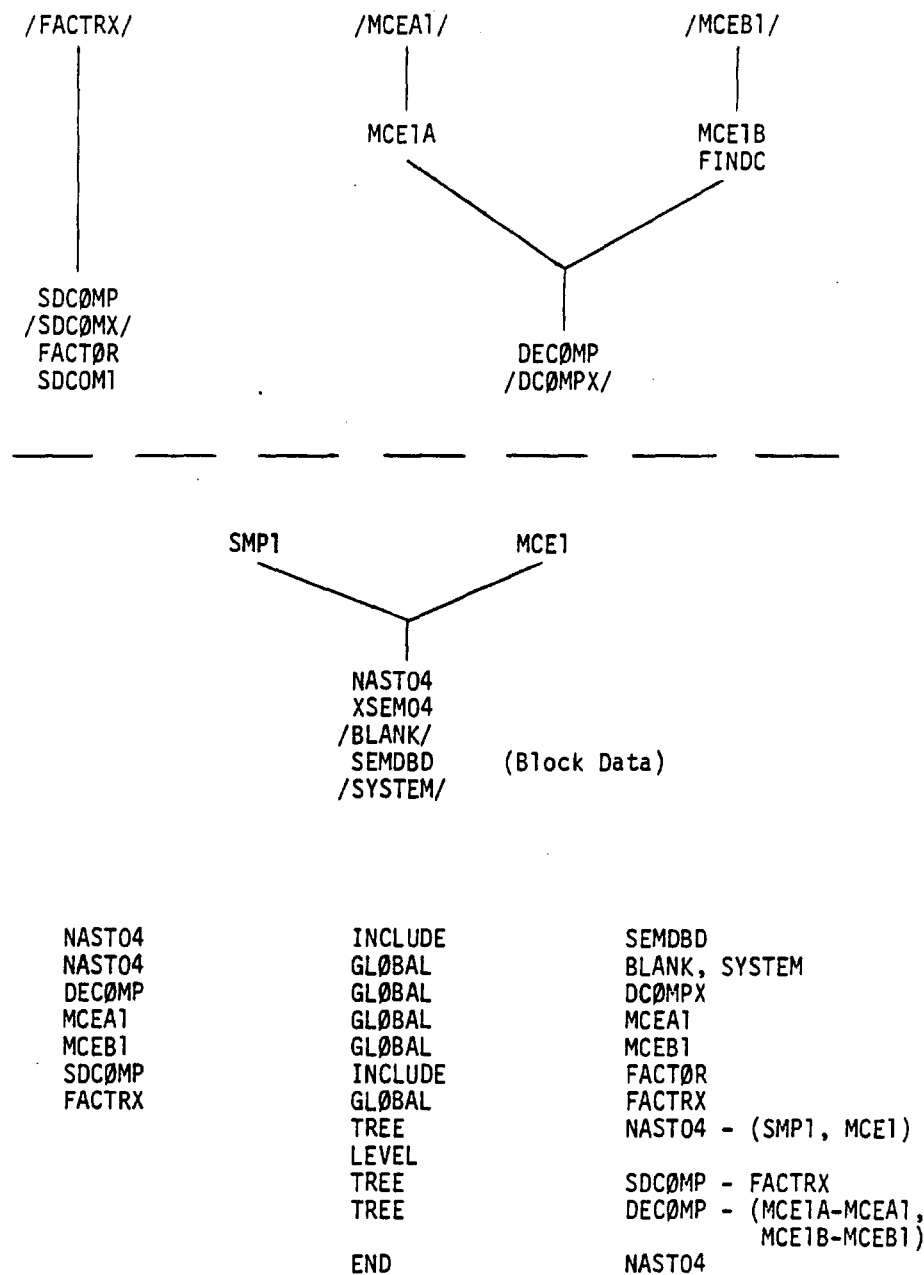


Figure 2. NASTRAN Tree Structure and Segmentation Directives (Typical)

NASTRAN - OPERATING SYSTEM INTERFACES

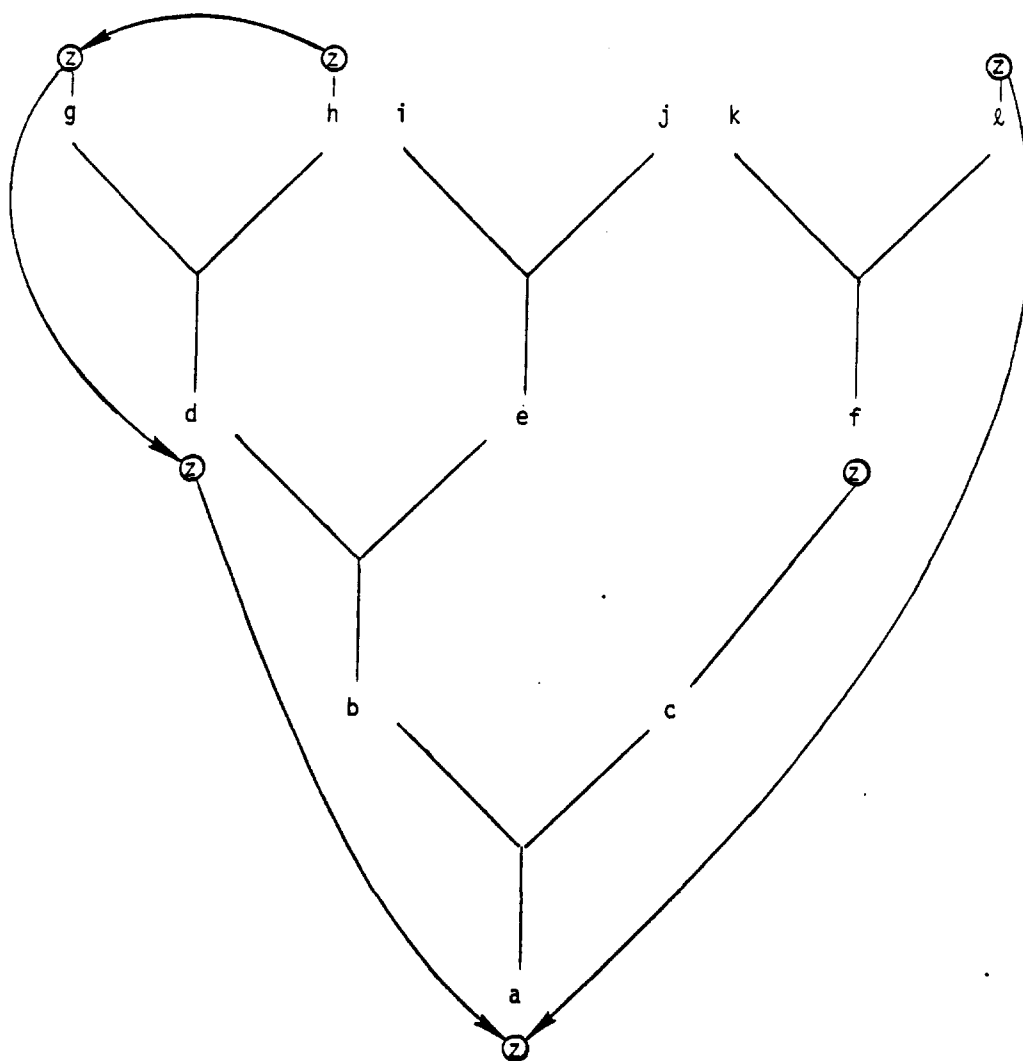


Figure 3. Floating of a Subroutine by the Segmentation Loader

SEGLOAD DIRECTIVES.

```

***** LINK12 2
* LINK12 3
* *** L I N K 1 2 *** LINK12 4
* LINK12 5
* THE FOLLOWING COMMON BLOCKS ARE NEEDED FOR THE ROOT EXECMOD 2
* EXECMOD 3
* EXECMOD 4
* GLOBAL XNSTRN,SYSTEM,GIND66,GINDX,XFIST,XFIAT,XXFIAT EXECMOD 5
* GLOBAL ZBLPKX,ZNTPKX,PACKX,UNPAKX,CONDAD,CONDAS,NTIME EXECMOD 6
* GLOBAL SQFCOM,TYPE,XPFIST,MSGX,TWO,XVPS,STIME,BITPOS,SETUP EXECMOD 7
* GLOBAL NAMES,OSCENT,XDPL,STAPID,XCEITB,XMDMSK,OUTPUT EXECMOD 8
* GLOBAL BLANK,SEM,XLINK,DESCRP EXECMOD 9
* EXECMOD 10
* THE FOLLOWING INCLUDES ARE FOR EXECUTIVE MODULES THAT ARE CALLED BY EXECMOD 11
* ALL LINKS EXECMOD 12
* EXECMOD 13
* MSGWRT INCLUDE MSGWRT,USRMSG EXECMOD 14
* ENDSYS INCLUDE BTSTRP,BGNSYS EXECMOD 15
* XSFA INCLUDE RPDABD,XPURGE,XSOSGN,XDPH,XFILPS,XPOLCKZ EXECMOD 16
* ENDSSS GLOBAL ENDSSS EXECMOD 17
* PARMLX GLOBAL PARMLX EXECMOD 18
* ESFA GLOBAL ESFA EXECMOD 19
* SETVAL INCLUDE PARAML,QPARAMR,QPARAM EXECMOD 20
* EXECMOD 21
* THE FOLLOWING INCLUDES ARE FOR THIS LINK EXECMOD 22
* EXECMOD 23
* THE FOLLOWING INCLUDES ARE FOR LINK 12 LINK12 7
* LINK12 8
* LINK12 9
* NAST12 INCLUDE XSEM12,CN55BD,GIND66,SEMDBD LINK12 10
* NAST12 GLOBAL PATX,MATIN,MATOUT,EQMK1 LINK12 11
* DDRA1 GLOBAL DDRA1 LINK12 12
* DDRB1 GLOBAL DDRB1 LINK12 13
* DDRMM INCLUDE SETFND,GHMATS,MAGPHA,DDRMMS,DDRMMMP,BISLOC,MAT,PREMAT, LINK12 14
* ,PREMATZ,GPTABD LINK12 15
* DDRMM GLOBAL DDRMC1,CLS TRS,GPTA1 LINK12 16
* DDRMMX GLOBAL DDRMMX LINK12 17
* DDR1X GLOBAL DDR1X,SSGA2 LINK12 18
* FBSC INCLUDE FBS,FBS1,FBS2,FBS3,FBS4,SSG3A LINK12 19
* FBSC GLOBAL FBSX LINK12 20
* MODACX GLOBAL MODACX LINK12 21
* RULERC INCLUDE RULERC,CALCV,EQMKM LINK12 22
* RULERC GLOBAL PARMEG LINK12 23
* SDR1 GLOBAL SDR1 LINK12 24
* SDRB1 GLOBAL SDRB1,SSGB2 LINK12 25
* SDR1A INCLUDE SDR1D,SDR1AZZ LINK12 26
* SDR1BB INCLUDE SDR1C,MERGE,SDR1BZZ,SDR1B LINK12 27
* SORTC INCLUDE SORT LINK12 28
* SSGA3 GLOBAL SSGA3 LINK12 29
* SSG2AC INCLUDE SSG2A,PARTN,CURCAS,MXCID,BISHEL LINK12 30
* SSG2BC INCLUDE SSG2B,MPYAD,MPYQ,FILSWI LINK12 31
* SSG2BC GLOBAL MPYADX LINK12 32
* VDR INCLUDE VDR8D LINK12 33
* VDRCDR GLOBAL VDRCDR LINK12 34
* LINK12 35
* THE FOLLOWING INCLUDES KLUDGE AN ERROR IN SOME_SEG. LOADER VERSIONS LINK12 36
* DDRMM INCLUDE DDRMM LINK12 37
* SDR1A INCLUDE SDR1A LINK12 38
* VDR INCLUDE VDR LINK12 39
* LINK12 40
* TREE NAST12-(SDR1,DDR1,DDR2,EQMK) LINK12 41
* LEVEL LINK12 42
* EXEC TREE 2
* THE FOLLOWING TREES ARE FOR EXECUTIVE MODULES THAT ARE CALLED BY ALL EXEC TREE 3
* LINKS EXEC TREE 4
* EXEC TREE 5
* TREE ENDSYS-(ENDSSS) EXEC TREE 6
* TREE SETVAL-(PARMLX) EXEC TREE 7
* TREE XSAVE EXEC TREE 8
* TREE XCEI EXEC TREE 9
* TREE XCHK EXEC TREE 10
* TREE XSFA-(ESFA) EXEC TREE 11
* TREE MSGWRT EXEC TREE 12
* LINK12 44
* THE FOLLOWING TREES ARE FOR LINK 12 LINK12 45
* LINK12 46
* TREE SDR1A-(SDR1) LINK12 47
* TREE CYCT2B-(DDR1B-(DDR1),MODACC-(MODACX)) LINK12 48
* TREE SORTC-(SSG2BTC,VDR-(VDRCDR)) LINK12 49
* SSG2BTC TREE SSG2BC-(FBSC-(SSGA3),RULERC,DDR1A-(DDR1),DDRMM) LINK12 50
* RULERC TREE RULERC-(SDR1B-(SDR1),SSG2AC-(DDR1X)) LINK12 51
* DDRMM TREE DDRMM-(DDRMM2-DDRMMX,DDRMM1) LINK12 52
* END NAST12 LINK12 53

```

Figure 4. NASTRAN Segmentation Directives for Link 12

NASTRAN - OPERATING SYSTEM INTERFACES

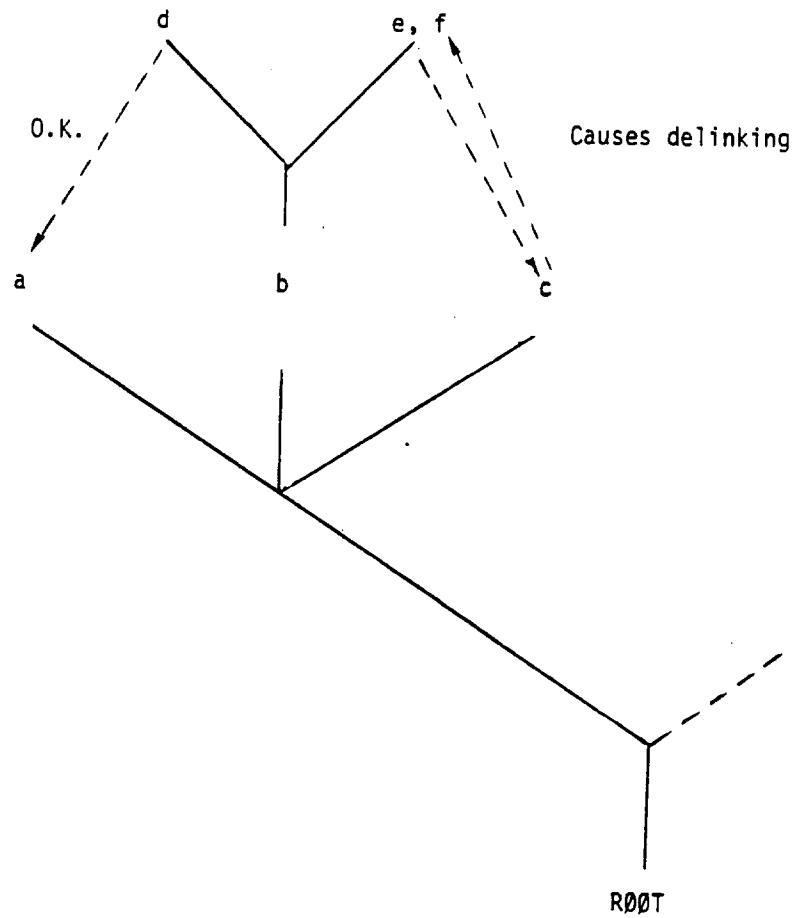


Figure 5. Inadvertent Delinking of Segments

NASTRAN ON THE CDC 6000/CYBER

CONTROL CARDS FOR THE SEGMENT LOADER VERSION OF NASTRAN

JOB,T10000,CM220000.
USER,069305C.
CHARGE,100334,LRC.

--UPDATE THE SOURCE PL AND COMPILE--

ATTACH(NSD=NSD1705/NA) GET THE SOURCE PL FROM DISC PACK.
UPDATE(P=NSD,Q,L=A1234,I=INPUT) UPDATE THE SOURCE PROGRAM LIBRARY.
FTN(B=LGO,R=3,I,L=OUTPUT,DPT=2,A,PL=300000) COMPILE MODIFIED ROUTINES.

--INSERT ENTRY TABLES IN BLOCK DATAS--

GET(BLKENTR/UN=069305C,NA) CREATE ENTRY POINT TABLES (368)
BLKENTR(LGO,OBJ) SO THE BLOCK DATA ROUTINES CAN
RETURN(BLKENTR,LGO) BE LIBGEN-ED.

--MERGE NEW BINARIES INTO OBJECT FILE--

ATTACH(NOB=NOB1705/NA) ATTACH THE OLD OBJECT DECKS.
REWIND(NOB,OBJ)
COPYL(NOB,OBJ,MASTOBJ,A) MERGE NEW OBJECT DECKS INTO OLD.
RETURN(NOB,OBJ)

--LIBGEN THE OBJECT DECKS--

REWIND(MASTOBJ)
RFL(150000)
LIBGEN(F=MASTOBJ,P=MASTLIB,NX=1) CREATE A NASTRAN OBJECT LIBRARY.
RETURN(MASTOBJ)

--UPDATE THE LINK STRUCTURE (SUBSYS)--

GET(NLK=NLK1705/NA) GET THE LINK STRUCTURE SOURCE PL.
UPDATE(P=NLK,Q,L=A1234,I=INPUT,C=SUBSYS) .PROCESS LINK STRUCTURE MODS.
--GET OLD SEG LOADER LINKS--
GET,LINK1,LINK2,LINK3,LINK4,LINK5,LINK6,LINK7,LINK8,LINK9/NA.
GET,LINK10,LINK11,LINK12,LINK13,LINK14,LINK15/NA.

--SEGLOAD THE LINKS TO BE CHANGED--

LIBRARY(MASTLIB,FORTRAN,SYSID)
RFL(150000)
SEGLOAD(I=SUBSYS,R=LINK2,LD=DT) LOAD SEQUENCE FOR LINK 2 (EXAMPLE)
LOSET(MAP=SB)
LIBLOAD(MASTLIB,MAST02) LOAD FIRST FROM LIBRARY MASTLIB.
NOGO.

...
(ALL OTHER LINKS ARE THE SAME EXCEPT LINK8. SEE ATTACHMENT)
...

--EXECUTE NASTRAN--

RFL(220000)
REDUCE(-)
LINK1,INPUT.
--EXECUTE ANOTHER CASE--
LINK1,INPUT.
/EOR
(SOURCE CODE MODS)
/EOR
(LINK STRUCTURE MODS)
/EOR
(INPUT DATA FOR CASE 1)
/EOR
(INPUT DATA FOR CASE 2)
/EOF

Figure 6. Control Cards for Segmentation Loader Alters and Execution

NASTRAN - OPERATING SYSTEM INTERFACES

```

PROGRAM BLKDAT(TAPE1=0,TAPE2=0,OUTPUT,TAPE6=OUTPUT)
INTEGER N(10003),B(10000)
EQUIVALENCE (N(4),B(1))
DATA IENTRY / 300 /

C
REWIND 1
REWIND 2
75 CONTINUE
BUFFER IN (1,1) (4(1),B(10000))
IF(UNIT(1)) 100,1000,1000

C
C DETERMINE THE LENGTH OF THIS SUBPROGRAM
C
100 LEN = LENGTH(1)
ICNT = 1
125 CONTINUE

C
C SHIFT TO PICK UP THE TABLE ID (ICLK) AND WORD COUNT (IK)
C ICNT WILL BE THE ADDRESS OF THE NEXT TABLE.
C
LICNT = ICNT-1
ICLK = SHIFT(B(ICNT),5)
IK = SHIFT(ICLK,-42) + 1
ICLK = AND(ICLK,77B)
ICNT = ICNT + IK

C
C IF PREFIX OR LDSET TABLE, SKIP TO NEXT TABLE
C
IF(ICLK .EQ. 77B) GO TO 125
IF(ICLK .EQ. 70B) GO TO 125

C
C DETERMINE IF THERE IS AN ENTRY POINT TABLE IN THIS SUBPROGRAM
C
IF(ICLK - IENTRY) 125,150,250

C
C ENTRY POINT TABLE PRESENT - SWAP PROGRA. OUT
C
150 BUFFER OUT (2,1) (B(1),B(LEN))
IF(UNIT(2)) 75,1000,1000

C
C NO ENTRY POINT TABLE - SHUFFLE TABLES PROCESSED SO FAR
C UP THREE WORDS TO MAKE ROOM FOR ONE
C
C
250 DO 260 I=1,LICNT
N(I) = B(I)
260 CONTINUE

C
C INSERT ENTRY POINT TABLE
C
LICNT = LICNT+1
N(LICNT) = SHIFT(3600 0002B , 36)
N(LICNT+1) = N(2)
N(LICNT+2) = 100 0001B

C
C SWAP OUT SUBPROGRAM AND PROCESS NEXT
C
KLEN = LEN + 3
BUFFER OUT (2,1) (N(1),N(KLEN))
IF(UNIT(2)) 75,1000,1000
1000 STOP
END

```

Figure 7. Program BLKENTR Source Code

5.5.3.4 NASTRAN on NØS 1.3 and NØS/BE 1.3

NASTRAN executables created under NØS 1.2 will run under NØS 1.3. If code alter runs are made under NØS 1.3, however, NASTRAN will fail for the following reasons.

During link switching, NASTRAN must (1) flush output buffers, and (2) test for user overrides of default file names. Under NØS 1.2 NASTRAN subroutine LINK calls NASTRAN subroutine FLUSH to flush buffers. FLUSH locates the FETs for ØUTPUT, PUNCH and TAPE11 using the RA+2 file list; then tests for information in the buffer, and issues an RA+1 WRITE TØ EØR call if necessary. Then LINK uses the RA+2 default file list (which is that file list specified on the PRØGRAM card), to locate the FET to pick up any file name overrides, in order to format a control card for the next LINK call.

Under NØS 1.3 the RA+2 file list does not exist, and the FET is restructured. Subroutines LINK and FLUSH must be modified accordingly by a systems programmer.

5.5.4 Physical Deliverables

The CDC version of NASTRAN is delivered on six (6) tapes. All delivery tapes are System Internal (SCØPE compatible) format, nine track, 1600 bpi, phase encoded tapes.

NASTØ1 - EXECUTABLE

This tape contains the executables of the fifteen links and the Segmentation Loader Load Maps.

File 1 - Link 1

⋮

File 15 - Link 15

File 16 - Load maps

NASTØ2 - SOURCE AND OBJECT DECKS

This tape contains source code for the CDC NASTRAN program, Segmentation Loader directives, BLKENTR block data entry point program, and NASTPLT plot-postprocessor, all in UPDATE sequential program library format, and NASTRAN and BLKENTR relocatable decks.

File 1 - NASTRAN ØLDPL (1 record)

Machine-independent decks appear in alphabetical order, followed by machine-dependent subprograms in alphabetical order, and terminated by the main program NASTRAN and the link main programs NASTØ1 to NAST15.

NASTRAN - OPERATING SYSTEM INTERFACES

File 2 - Segmentation Loader Directives (1 record)

Directives for each of the 15 links are terminated by a *WEØR card.

File 3 - BLKENTR Source (1 record)

File 4 - Plot-postprocessor Source (1 record)

Machine-independent source, followed by machine dependent source, terminated by Langley Research Center dependent source for example use.

File 5 - ØBJECT Decks (1678 records)

This contains the relocatable decks obtained by compiling File 1 above under FTN 4.6 - PSR Level 452.1*, ØPT=2. (Because of a compiler bug, routine INCØRE had to be compiled under FTN 4.6 - PSR Level 452, ØPT=2 and routines DTSHLD and KTSHLD could not be compiled.)

File 6 - BLKENTR ØBJECT Deck (1 record)

This contains the relocatable deck obtained by compiling File 3 above under FTN 4.6 - PSR Level 452.

NAST03 - DEMO ITEMS

This tape contains the User Master File, the Demonstration Problem Driver Decks, and the UMF verification.

File 1 - User Master File (408 data records produced by NASTRAN)

File 2 - Demonstration Problem Driver Decks (1 record, representing 82 Demos, in UPDATE format)

Each driver deck name is adapted from the UMF pid.

The deck names are DDC---- without the trailing 0. For example, the deck name on a *CØMPILE card for Problem 1-4-1 is DDC1041. Restart deck names are appended with an A (for the first restart) and B (for the second restart). The deck name for Problem 1-1-1A is thus DDC1011A.

File 3 - Print of UMF verification (1 record in print file format)

NAST04 - DEMO PRINT FILES

One file of demonstration problem executions.

Records 1 through 82 - 1 execution per record

*FTN 4.6 - PSR Level 452.1 is being used at Langley Research Center and is a modified version of FTN 4.6 - PSR Level 452.

NASTRAN ON THE CDC 6000/CYBER

NAST05 - COMPILATION LISTINGS

This tape contains the CDC compilation listings of subroutines A-M.

Record 1 - Routines ADR through DTRMEM

Routines DTSHLS through IHEXSS

Routines INCRØ through KTRPLT

Routines KTSHLS through MXCIDS

Record 2 - Routine INCØRE

NAST06 - COMPILATION LISTINGS

This tape contains the CDC compilation listings of subroutines N-Z plus machine dependent routines.

File 1 - Routines NASCAR through ZJ followed by all machine-dependent routines.

NASTRAN - OPERATING SYSTEM INTERFACES

5.5.5 Example Control Card Setups

The following examples show control card setups for using NASTRAN under the CDC Segmentation Loader. Accounting, tape requests and plot library control cards may vary at different installations. The first four deck setups illustrate usage on NØS 1.2, the last two on NØS/BE 1.2. NØS 1.3 - NØS/BE 1.3 differences are discussed in Section 5.5.4 above.

SCOPE users can execute problems from the Segmentation Loader executable tape, but cannot make code alter runs. They can use the Level 17.5 source and relocatables with the Level 17.0 Linkage Editor by changing all BLOCK DATA (name) cards to unnamed BLOCK DATA. Linkage Editor directives for incorporating the Level 17.5 Modal Synthesis and Strain Recovery capabilities are not available.

- Example 1 - To load and execute a single NASTRAN problem.

```

NASTRAN,T500.
USER,069305C.
CHARGE,100334,LRC.
DELIVER.0    B11
*--DUMP NASTRAN FROM TAPE TO DISC
LAFRL(TAPE,NT,TAPE,POWER,LS=K0,FES)NON-LPC=NASTRAN
REWIND,TAPE.
COPYBF,TAPE,LINK1.
COPYBF,TAPE,LINK2.
COPYBF,TAPE,LINK3.
COPYBF,TAPE,LINK4.
COPYBF,TAPE,LINK5.
COPYBF,TAPE,LINK6.
COPYBF,TAPE,LINK7.
COPYBF,TAPE,LINK8.
COPYBF,TAPE,LINK9.
COPYBF,TAPE,LINK10.
COPYBF,TAPE,LINK11.
COPYBF,TAPE,LINK12.
COPYBF,TAPE,LINK13.
COPYBF,TAPE,LINK14.
COPYBF,TAPE,LINK15.
RETURN,TAPE.
*--EXECUTE NASTRAN
RFL(220000)
REDUCE(-)
LINK1,INPUT.
/EOR
      (NASTRAN INPUT DATA)
/EOF

```

NASTRAN ON THE CDC 6000/CYBER

- Example 2 - To correct source decks common to all links, generate a new set of executables, and execute a test problem

```

NASTRAN,TS00.
USER,069305C.
CHARGE,100334,LRC.
DELIVER,0 B11
*--RECREATE NASTRAN FROM TAPE
LAPL(TAPE,RT,DEPE,PO=2,LS=RO,FE=ST)NON-LRC=TEST12
REWIND,TAPE.
COPYBF,TAPE,NSO.
COPYBF,TAPE,NSU.
COPYBF,TAPE,DUM,2.
COPYBF,TAPE,NOB.
COPYBF,TAPE,BLKENTR.
RETURN,TAPE.
REWIND,NSO,NSU,NOB,BLKENTR.
*--PROCESS SOURCE MODS
UPDATE,P=NSO,Q,U.
RETURN,NSO.
FTN,A,I,OPT=2,R=3,PL=300000.
RETURN,COMPILE.
*--PUT ENTRY POINTS IN BLOCK DATAS.
BLKENTR,LGO,OBJ.
RETURN,BLKENTR,LGO.
*--MERGE NEW RELOCATABLES INTO OLD.
REWIND,OBJ,NOB.
COPYL,NOB,OBJ,NASTOBJ.
RETURN,NOB,OBJ.
*--CREATE LIBRARY *NASTLIB*
RFL(150000)
LIBRARY(NASTLIB,FORTRAN,SYSIO)
LIBGEN(F=NASTOBJ,P=NASTLIB,NX=1)
LIBRARY(NASTLIB,FORTRAN,SYSIO)
RETURN,NASTOBJ.
*--PROCESS SEGMENTATION DIRECTIVE MODS.
UPDATE,P=NSU,Q,U,C=SUBSYS.
*--SEGMENTATION LOAD
RFL(220000)
LDSET(MAP=SB)
SEGLOAD(I=SUBSYS,B=LINK1)
LIBLOAD(NASTLIB,NAST01)
NOGO.
SEGLOAD(I=SUBSYS,B=LINK2)
LIBLOAD(NASTLIB,NAST02)
NOGO.
SEGLOAD(I=SUBSYS,B=LINK3)
LIBLOAD(NASTLIB,NAST03)
NOGO.
SEGLOAD(I=SUBSYS,B=LINK4)
LIBLOAD(NASTLIB,NAST04)
NOGO.
SEGLOAD(I=SUBSYS,B=LINK5)
LIBLOAD(NASTLIB,NAST05)
NOGO.
SEGLOAD(I=SUBSYS,B=LINK6)
LIBLOAD(NASTLIB,NAST06)

```


NASTRAN - OPERATING SYSTEM INTERFACES

```

NOGO.
SEGLOAD(I=SUBSYS,B=LINK7)
LIBLOAD(NASTLIB,NAST07)
NOGO.
SEGLOAD(I=SUBSYS,B=LINK8)
LDSET(SUBST=AXIF2D-EMGOLD/AXIF2S-EMGOLD/AXIF3D-EMGOLD/AXIF3S-EMGOLD)
LDSET(SUBST=AXIF4D-EMGOLD/AXIF4S-EMGOLD/CONED-EMGOLD/CONES-EMGOLD)
LDSET(SUBST=FLMASD-EMGOLD/FLMASS-EMGOLD/FLUD2D-EMGOLD/FLUD2S-EMGOLD)
LDSET(SUBST=FLUD3D-EMGOLD/FLUD3S-EMGOLD/FLUD4D-EMGOLD/FLUD4S-EMGOLD)
LDSET(SUBST=HBDYD-EMGOLD/HBDYS-EMGOLD/HEXA1D-EMGOLD/HEXA1S-EMGOLD)
LDSET(SUBST=HEXA2D-EMGOLD/HEXA2S-EMGOLD/PLOTLD-EMGOLD/PLOTLS-EMGOLD)
LDSET(SUBST=QDMEMD-EMGOLD/QDMEMS-EMGOLD/QDMM3D-EMGOLD/QDMM3S-EMGOLD)
LDSET(SUBST=QDPLTD-EMGOLD/QDPLTS-EMGOLD/QUAD1D-EMGOLD/QUAD1S-EMGOLD)
LDSET(SUBST=QUAD2D-EMGOLD/QUAD2S-EMGOLD/SLOT3D-EMGOLD/SLOT3S-EMGOLD)
LDSET(SUBST=SLOT4D-EMGOLD/SLOT4S-EMGOLD/TETRAD-EMGOLD/TETRAS-EMGOLD)
LDSET(SUBST=TRIARD-EMGOLD/TRIARS-EMGOLD/TRIA1D-EMGOLD/TRIA1S-EMGOLD)
LDSET(SUBST=TRIA2D-EMGOLD/TRIA2S-EMGOLD/TRPLTD-EMGOLD/TRPLTS-EMGOLD)
LDSET(SUBST=TRAPRD-EMGOLD/TRAPRS-EMGOLD/WEDGED-EMGOLD/WEDGES-EMGOLD)
LDSET(SUBST=SMA1B-EMG1B/SMA2B-EMG1B)
LDSET(OMIT=KBAR/KTORDR/KTPZ/KTRIA/MBAR/MCBAR/MPZDA/MSTRIA/MTORDR)
LIBLOAD(NASTLIB,NAST08)
NOGO.
SEGLOAD(I=SUBSYS,B=LINK9)
LIBLOAD(NASTLIB,NAST09)
NOGO.
SEGLOAD(I=SUBSYS,B=LINK10)
LIBLOAD(NASTLIB,NAST10)
NOGO.
SEGLOAD(I=SUBSYS,B=LINK11)
LIBLOAD(NASTLIB,NAST11)
NOGO.
SEGLOAD(I=SUBSYS,B=LINK12)
LIBLOAD(NASTLIB,NAST12)
NOGO.
SEGLOAD(I=SUBSYS,B=LINK13)
LIBLOAD(NASTLIB,NAST13)
NOGO.
SEGLOAD(I=SUBSYS,B=LINK14)
LIBLOAD(NASTLIB,NAST14)
NOGO.
SEGLOAD(I=SUBSYS,B=LINK15)
LIBLOAD(NASTLIB,NAST15)
NOGO.
*--EXECUTE NASTRAN.
RFL(220000)
REDUCE(-)
LINK1,INPUT.
/EDR
*/ (NASTRAN SOURCE CODE MODS)
/EDR
*/ (NASTRAN LINK DIRECTIVE MODS)
%C LINK1.LINK15
/EDP
(NASTRAN INPUT DATA)
/EDF

```

NASTRAN ON THE CDC 6000/CYBER

Example 3 - To edit three links, execute a problem, and plot using the NASTPLT plot-processor and the Langley Research Center CALCOMP plotter.

```

NASTRAN,T500.
USER,069305C.
CHARGE,100334,LRC.
DELIVER,0      ALL
*--MODIFY SELECTED NASTRAN LINKS
LABEL(TAPE,NT,)=PE,PO=7,LE=10,F=ST)NON-LOC=NAST02
REWIND,TAPE.
COPYBF,TAPE,NS0.
COPYBF,TAPE,NSU.
COPYBF,TAPE,DUM.
COPYBF,TAPE,POSTPLT.
COPYBF,TAPE,NOB.
COPYBF,TAPE,BLKENTR.
RETURN,TAPE.
REWIND,NS0,NSU,NOB,BLKENTR.
*--PROCESS SOURCE MODS
UPDATE,P=NS0,Q,U.
RETURN,NS0.
FTN,A,I,OPT=2,R=3,PL=300000.
RETURN,COMPILE.
*--PUT ENTRY POINTS IN BLOCK DATAS.
BLKENTR,LGO,OBJ.
RETURN,BLKENTR,LGO.
*--MERGE NEW RELOCATABLES INTO OLD.
REWIND,OBJ,NOB.
COPYL,NOB,OBJ,NASTOBJ.
RETURN,NOB,OBJ.
*--CREATE LIBRARY *NASTLIB*
RFL(150000)
LIBGEN(F=NASTOBJ,P=NASTLIB,NX=1)
LIBRARY(NASTLIB,FORTRAN,SYSIO)
RETURN,NASTOBJ.
*--GET OLD EXECUTABLE FROM TAPE.
LABEL(TAPE,NT,)=PE,PO=7,LE=10,F=ST)NON-LOC=NAST01
REWIND,TAPE.
COPYBF,TAPE,LINK1.
COPYBF,TAPE,LINK2.
COPYBF,TAPE,LINK3.
COPYBF,TAPE,LINK4.
COPYBF,TAPE,LINK5.
COPYBF,TAPE,LINK6.
COPYBF,TAPE,LINK7.
COPYBF,TAPE,LINK8.
COPYBF,TAPE,LINK9.
COPYBF,TAPE,LINK10.
COPYBF,TAPE,LINK11.
COPYBF,TAPE,LINK12.
COPYBF,TAPE,LINK13.
COPYBF,TAPE,LINK14.
COPYBF,TAPE,LINK15.
RETURN,TAPE.
*--PROCESS SEGMENTATION DIRECTIVE MODS.

```

NASTRAN - OPERATING SYSTEM INTERFACES

```

UPDATE,P=NSU,Q,U,C=SUBSYS.
*--SEGMENTATION LOAD.
RFL(220000)
LDSET(MAP=SB)
SEGLOAD(I=SUBSYS,B=LINK1)
LIBLOAD(NASTLIB,NAST01)
NOGO.
SEGLOAD(I=SUBSYS,B=LINK7)
LIBLOAD(NASTLIB,NAST07)
NOGO.
SEGLOAD(I=SUBSYS,B=LINK14)
LIBLOAD(NASTLIB,NAST14)
NOGO.
*--EXECUTE NASTRAN.
RFL(220000)
REDUCE(-)
LINK1,INPUT.
*--EXECUTE THE PLOT POSTPROCESSOR.
UPDATE,F,P=POSTPLT,I=U.
FTN,I.
REWIND,PLT2.
*--PLOT USING THE LANGLEY SYSTEM SOFTWARE
ATTACH,PLOT,LRCGOSF/UN=LIBRARY.
RETURN(NASTLIB)
LDSET(LIB=LRCGOSF)
LGO.
PLOT,CALPOST,12 // BALL POINT PEN //
/EOR
*/ (NASTRAN SOURCE CODE MODS)
/EOR
*/ (MODS TO LINKS 1, 7, 14)
/EOR
NASTRAN FILES=PLT2
      (NASTRAN INPUT DATA)
/EOR
$INPUT$
/EOF

```

NASTRAN ON THE CDC 6000/CYBER

Example 4 - To execute Demonstration Problems 1-1-1 and 1-1-1A. These problems test Statics Rigid Format #1 and the restart code, and use the User Master File to obtain bulk data. Note that the PUNCH file, which contains restart table data, must be PACKed to eliminate record marks (7-8-9 cards) within the file.

```

NASTRAN,T500.
USER,069305C.
CHARGE,100334,LRC.
DELIVER,0,11
LABEL(TAPE,NT,DEP,PO=7,LD=40,F=50)NAME=LRC=NASTRAN
REWIND,TAPE.
COPYBF,TAPE,LINK1.
COPYBF,TAPE,LINK2.
COPYBF,TAPE,LINK3.
COPYBF,TAPE,LINK4.
COPYBF,TAPE,LINK5.
COPYBF,TAPE,LINK6.
COPYBF,TAPE,LINK7.
COPYBF,TAPE,LINK8.
COPYBF,TAPE,LINK9.
COPYBF,TAPE,LINK10.
COPYBF,TAPE,LINK11.
COPYBF,TAPE,LINK12.
COPYBF,TAPE,LINK13.
COPYBF,TAPE,LINK14.
COPYBF,TAPE,LINK15.
RETURN,TAPE.
*--GET THE UMF AND DRIVER DECKS
LABEL(TAPE,NT,DEP,PO=7,LD=40,F=50)NAME=LRC=NASTRAN
COPYBF,TAPE,UMF.
COPYBF,TAPE,OLDPL.
REWIND,UMF.
RETURN,TAPE.
*--UPDATE A DRIVER DECK AND EXECUTE
UPDATE,Q,C=NASDATA.
RFL(220000)
REDUCE(-)
LINK1,NASDATA.
REDUCE.
*--PROCESS FILES FOR RESTART
REWIND,NPTP,PUNCH,UMF.
PACK,PUNCH.
RETURN,POUL.
RENAME,OPTP=NPTP.
*--UPDATE SECOND DRIVER DECK AND EXECUTE
UPDATE,Q,C=NASDATA.
RFL(220000)
REDUCE(-)
LINK1,NASDATA.
RETURN(PUNCH)
/EOF
*C DDC1011
/EOF
*C DDC1011A
*I DDC1011A.3
*READ PUNCH
/EOF

```

NASTRAN - OPERATING SYSTEM INTERFACES

Example 5 - To execute a problem under N0S/BE. This deck set-up reads in a Cyber Control Language Procedure File (CCL Proc) and transfers control to it. When SYSTEM(76) is non-zero, NASTRAN writes a procedure file to TAPE99 which implements link switching.

```

NASTRAN,TS00.
USER,069305C.
CHARGE,100334,LRC.
DELIVER,1,BILL
LIFE,RTS00. T.REF.,NAME,LINEO.FEND; OF-LEAD-TIME 1
REWIND,TAPE.
COPYHF,TAPE,LINK1.
COPYHF,TAPE,LINK2.
COPYHF,TAPE,LINK3.
COPYHF,TAPE,LINK4.
COPYHF,TAPE,LINK5.
COPYHF,TAPE,LINK6.
COPYHF,TAPE,LINK7.
COPYHF,TAPE,LINK8.
COPYHF,TAPE,LINK9.
COPYHF,TAPE,LINK10.
COPYHF,TAPE,LINK11.
COPYHF,TAPE,LINK12.
COPYHF,TAPE,LINK13.
COPYHF,TAPE,LINK14.
COPYHF,TAPE,LINK15.
RETURN,TAPE.
***--READ AND EXECUTE NASTRAN PROC.
COPYCR,INPUT,NASTRAN.
BEGIN,NASTRAN,NASTRAN.
/EOR
.PROC,NASTRAN,RFL=200000,INPUT=,OUTPUT=,PUNCH=.
#RFL,RFL.
LINK1,INPUT,OUTPUT,PUNCH.
#MILE,RIG,EQ.0,REPEAT.
BEGIN,NEXT,TAPE99.
END*(REPEAT)
REVERT.
EXIT.
REVERT(ABORT)
/EOR
NASTRAN SYSTEM(76)=1
      (NASTRAN INPUT DATA)
/EOR

```

NASTRAN ON THE CDC 6000/CYBER

Example 6 - To relink all of NASTRAN under NØS/BE and execute a problem. Because of the way NØS/BE processes FØRTRAN files, the NASTRAN link main programs are segregated on a separate file and the SLOAD directive used. Note that the libraries FØRTRAN and SYSIO must be replaced on the LIBRARY card with their local installation equivalents. Note also that some NØS/BE installations put limits on the size of user libraries. In such cases, NASTRAN must be split into two or more libraries. The local CDC Systems Engineer should be consulted for details.

```

NASTRAN,T500.
USER,069305C.
CHARGE,100334,LRC.
DELIVER,0 -11
*--RECREATE NASTRAN FROM TAPE
LABEL(TAPE,NT,REFR,PQ=1,LD=NO,TF=5) DON-LRCE,1,1,12
REWIND,TAPE.
COPYBF,TAPE,NSO.
COPYBF,TAPE,NSU.
COPYBF,TAPE,DUM,2.
COPYBF,TAPE,NOB.
COPYBF,TAPE,BLKENTR.
RETURN,TAPE.
REWIND,NSO,NSU,NOB,BLKENTR.
*--PROCESS SOURCE MODS
UPDATE,P=NSO,Q,U.
RETURN,NSO.
FTN,A,I,OPT=2,R=3,PL=300000.
RETURN,COMPILE.
*--PUT ENTRY POINTS IN BLOCK DATAS.
BLKENTR,LGO,OBJ.
RETURN,BLKENTR,LGO.
*--MERGE NEW RELOCATABLES INTO OLD.
REWIND,OBJ,NOB.
COPYL,NOB,OBJ,NASTOBJ.
RETURN,NOB,OBJ.
*--CREATE LIBRARY *NASTLIB*
EDITLIB.
LIBRARY(NASTLIB,FORTTRAN,SYSIO)
*--CREATE FILE OF MAIN PROGRAMS
COPYBF,NASTOBJ,NASMAIN.
RETURN,NASTOBJ.
*--PROCESS SEGMENTATION DIRECTIVE MODS.
UPDATE,P=NSU,Q,U,C=SUBSYS.
*--SEGMENTATION LOAD
RFL(220000)
LDSET(MAP=SB)
SEGLOAD(I=SUBSYS,B=LINK1)
SLOAD(NASMAIN,NAST01)
NOGO.
SEGLOAD(I=SUBSYS,B=LINK2)
SLOAD(NASMAIN,NAST02)
NOGO.
SEGLOAD(I=SUBSYS,B=LINK3)
SLOAD(NASMAIN,NAST03)
NOGO.
SEGLOAD(I=SUBSYS,B=LINK4)
SLOAD(NASMAIN,NAST04)

```

NASTRAN - OPERATING SYSTEM INTERFACES

```

NOGO.
SEGLOAD(I=SUBSYS,B=LINK5)
SLOAD(NASMAIN,NAST05)
NOGO.
SEGLOAD(I=SUBSYS,B=LINK6)
SLOAD(NASMAIN,NAST06)
NOGO.
SEGLOAD(I=SUBSYS,B=LINK7)
SLOAD(NASMAIN,NAST07)
NOGO.
SEGLOAD(I=SUBSYS,B=LINK8)
LDSET(SUBST=AXIF2D-EMGOLD/AXIF2S-EMGOLD/AXIF3D-EMGOLD/AXIF3S-EMGOLD)
LDSET(SUBST=AXIF4D-EMGOLD/AXIF4S-EMGOLD/CONED-EMGOLD/CONES-EMGOLD)
LDSET(SUBST=FLMASD-EMGOLD/FLMASS-EMGOLD/FLUD2D-EMGOLD/FLUD2S-EMGOLD)
LDSET(SUBST=FLUD3D-EMGOLD/FLUD3S-EMGOLD/FLUD4D-EMGOLD/FLUD4S-EMGOLD)
LDSET(SUBST=HBDYD-EMGOLD/HBDYS-EMGOLD/HEXA1D-EMGOLD/HEXA1S-EMGOLD)
LDSET(SUBST=HEXA2D-EMGOLD/HEXA2S-EMGOLD/PLOTLD-EMGOLD/PLOTLS-EMGOLD)
LDSET(SUBST=QDMEMD-EMGOLD/QDMEMS-EMGOLD/QDMM3D-EMGOLD/QDMM3S-EMGOLD)
LDSET(SUBST=QDPLTD-EMGOLD/QDPLTS-EMGOLD/QUAD1D-EMGOLD/QUAD1S-EMGOLD)
LDSET(SUBST=QUAD2D-EMGOLD/QUAD2S-EMGOLD/SLOT3D-EMGOLD/SLOT3S-EMGOLD)
LDSET(SUBST=SLOT4D-EMGOLD/SLOT4S-EMGOLD/TETRAD-EMGOLD/TETRAS-EMGOLD)
LDSET(SUBST=TRIARD-EMGOLD/TRIARS-EMGOLD/TRIA1D-EMGOLD/TRIA1S-EMGOLD)
LDSET(SUBST=TRIA2D-EMGOLD/TRIA2S-EMGOLD/TRPLTD-EMGOLD/TRPLTS-EMGOLD)
LDSET(SUBST=TRAPRD-EMGOLD/TRAPRS-EMGOLD/WEDGED-EMGOLD/WEDGES-EMGOLD)
LDSET(SUBST=SMA1B-EMG1B/SMA2B-EMG1B)
LDSET(OMIT=KBAR/KTORDR/KTPZ/KTRIA/MBAR/MCBAR/MPZDA/MSTRIA/MTORDR)
SLOAD(NASMAIN,NAST08)
NOGO.
SEGLOAD(I=SUBSYS,B=LINK9)
SLOAD(NASMAIN,NAST09)
NOGO.
SEGLOAD(I=SUBSYS,B=LINK10)
SLOAD(NASMAIN,NAST10)
NOGO.
SEGLOAD(I=SUBSYS,B=LINK11)
SLOAD(NASMAIN,NAST11)
NOGO.
SEGLOAD(I=SUBSYS,B=LINK12)
SLOAD(NASMAIN,NAST12)
NOGO.
SEGLOAD(I=SUBSYS,B=LINK13)
SLOAD(NASMAIN,NAST13)
NOGO.
SEGLOAD(I=SUBSYS,B=LINK14)
SLOAD(NASMAIN,NAST14)
NOGO.
SEGLOAD(I=SUBSYS,B=LINK15)
SLOAD(NASMAIN,NAST15)
NOGO.
*--READ AND EXECUTE NASTRAN PROC.
COPYCR,INPUT,NASTRAN.
BEGIN,NASTRAN,NASTRAN.
/EOP

```

NASTRAN ON THE CDC 6000/CYBER

```
LIBRARY(NASTLIB,NEW)
ADD(*XTRACE1,NASTOBJ)
DELETE(FBSI)
DELETE(MPY3T)
DELETE(SDCOM)
FINISH.
ENDRUN.
/EOB
*C LINK1.LINK15
/EOB
.PROC,NASTRAN,RFL=200000,INPUT=,OUTPUT=,PUNCH=.
#RFL,RFL.
LINK1,INPUT,OUTPUT,PUNCH.
SET,DSC=1.
WHILE,R1G.EQ.0,REPEAT.
BEGIN,NEXT,TAPE99.
ENDW(REPEAT)
REVERT.
EXIT.
REVERT(ABORT)
/EOB
NASTRAN  SYSTEM(76)=1
      (NASTRAN INPUT DATA)
/EOF
```


5.5.6 Machine Dependent Routines

The following utility routines necessary to NASTRAN operation must, by their nature, be implemented in a machine dependent manner. Certain of the routines have been written in COMPASS language, and the remainder are in FORTRAN language.

5.5.6.1 MAPFNS (COMPASS)

The MAPFNS deck embodies a set of 20 utility functions and routines. These are as follows:

Logical Functions

ANDF (logical product of two words)
 ORF (logical sum of two words)
 XORF (logical difference of two words)
 COMPLF (complement of a word)
 LSHIFT (left shift of a word)
 RSHIFT (right shift of a word)

Utility Functions

CORSZ (returns length of "open core")
 CORWDS (returns difference of two addresses + 1)
 LWORDS (returns difference of two addresses)
 LOCF (returns the value of an address)
 INSTAL (returns installation name, left justified, blank fill)

Utility Routines

XSTORE (stores an array at an absolute position in core)
 XFETCH (fetches an array from an absolute position in core)
 XJUMP (transfers control to an absolute location in core)
 ZAP (stores zeros between specified locations in core)
 LINK20. (provides a special call to Link 20, the Level 17.0 NASTRAN exit link)
 FIELDLN (returns the field length assigned to the job)
 TDATE (returns month, day, year, time)
 KLOCK (returns current value of CPU clock)
 DAYTIME (returns current time on wall clock)

Dummy Routines (These do nothing but return to the calling program. They are left in NASTRAN for compatibility with earlier versions.)

XCØMMØN

DMPXXX

SET66

5.5.6.2 CØNMSG (FØRTRAN)

The CØNMSG routine enters messages in the DAYFILE denoting the times of the initiation and completion of modules during NASTRAN execution.

5.5.6.3 GNFIAT (FØRTRAN)

GNFIAT initializes the XFIAT and FIAT tables, and clears the PØØL and NPTP files on restart.

5.5.6.4 DUMP and PDUMP (FØRTRAN)

A call to PDUMP produces a trace back listing and, if a DIAG1 card appeared in the Executive Control Deck, a memory dump. PDUMP returns to the calling program. DUMP terminates with a CALL EXIT.

5.5.6.5 PPCALL

PPCALL issues RA+1 requests to the Combined Input/Output (CIØ) Peripheral Processor routine.

5.5.6.6 SGINØ (FØRTRAN)

SGINØ is used by the plot routines to write plot tapes. SGINØ issues calls to XIORTNS.

5.5.6.7 XIØRTNS (FØRTRAN)

The XIØRTNS deck embodies a set of utility routines which communicate with the NØS Operating System through CIØ to accomplish various file operations. XIØRTNS sets up the File Environment Table for the file operation, then calls PPCALL to issue a CIØ request.

Entry Points

XØPEN (constructs FET and initiates activity for file)

XCLØSE (terminates activity for file)

XEVICT (evicts space assigned to a disk file)

NASTRAN - OPERATING SYSTEM INTERFACES

XREWIND (repositions file to load point)
XBKREC (repositions file one logical record backward)
XBKPREC (repositions file one physical record backward)
XFRDREC (repositions file one logical record forward)
READX (reads a complete logical record)
WRITEX (writes a complete logical record)

5.5.6.8 GINØ66 (FØRTRAN)

GINØ66 is a BLØCK DATA subprogram which initializes the GINØ66 common block with names of each of the possible NASTRAN files.

5.5.6.9 NASTØ1-NAST15 (FØRTRAN)

These are Link main programs that establish FØRTRAN buffers for the files INPUT, ØUTPUT, PUNCH, and TAPE11 before calling the link driver (XSEM) routines.

5.5.6.10 MPYQ (CØMPASS)

MPYQ is a compass routine written to increase the speed of MPYAD's inner loops (see Section 3.5.12.5).

5.5.6.11 WALTIM (FØRTRAN)

WALTIM calls DAYTIM for time of day and then calculates the elapsed time in seconds after midnight.

5.5.6.12 IØ66ØØ (FØRTRAN)

IØ66ØØ is an interface routine between GINØ and XIØRTNS. It maintains disk address information for the file being operated on and issues open and close messages as required by DIAG 15. The calling sequence is:

CALL IØ66ØØ (ØPCØDE,BUFF),RETURNS(RETURN1)

IØ66ØØ formats calls to appropriate entry points in XIØRTNS according to ØPCØDE. The requests and calls are:

NASTRAN ON THE CDC 6000/CYBER

<u>ØPCØDE</u>	<u>REQUEST</u>	<u>CALL</u>
1	rewind	XREWIND
2	write	WRITEX
3	read	READX
4	backspace	XBKREC*
5	forward space	READX*
6	open	XØPEN
7	close	XCLØSE

*Sequential files only. For direct access files the disc pointer is modified.

5.5.6.13 CN66BD (FØRTRAN)

CN66BD is a BLØCK DATA subprogram that provides single and double precision values for several arithmetic constants.

5.5.6.14 SØFIØ (FØRTRAN)

SØFIØ directs the input/output operations for substructuring between core and the random access storage device. The calling sequence is:

CALL SØFIØ (IRW,IBLKNM,IBUFF)

SØFIØ writes (IRW=1) or reads (IRW=2) block number IBLKNM from/to the buffer IBUFF.

5.5.6.15 XTRACE (FØRTRAN)

XTRACE directs the error traceback.

5.5.6.16 XTRACE1 (CØMPASS)

XTRACE1 is called by XTRACE to generate an error traceback.

5.5.6.17 SETBR (CØMPASS)

SETBR converts a FTN calling sequence of up to six arguments to a RUN FØRTRAN calling sequence. It is used to minimize CØMPASS code conversion effort.

5.5.6.18 STØREB (CØMPASS)

STØREB converts a RUN FORTRAN calling sequence of up to six arguments to a FTN calling sequence.

NASTRAN - OPERATING SYSTEM INTERFACES

5.5.6.19 XFLSTS (FØRTRAN)

XFLSTS maintains GINØ file block count statistics.

5.5.6.20 REINDX (CØMPASS)

REINDX is used by the Link Editor for managing its file. It is present only for compatibility with the Level 17.0 NASTRAN Linkage Editor.

5.5.6.21 READX1 (CØMPASS)

READX1 is an interface routine between the Level 17.0 NASTRAN Linkage Editor and Level 17.5 NASTRAN.

MODULE - DMAP NAME - MODULE ENTRY - LINKS MODULE RESIDES IN ON 6600
 INDEX OF MODULE POINT NAME

3	CHKPNT	XCHK	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
5	REPT	XCEI	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
6	JUMP	XCEI	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
7	CEND	XCEI	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
8	SAVE	XSAVE		2	3	4	5	6	7	8	9	10	11	12	13	14	15
9	PURGE	XPURGE	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
10	EQUIV	XEQUIV	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
11	END	XCEI	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
12	EXIT	XCEI	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
14	ADD	DADD				4			7								
15	ADD5	DADD5							7								
16	AMG	AMG									9						
17	AMP	AMP									9						
18	APD	APD									9						
19	RMG	RMG										10					
20	CASE	CASE										10					
21	CYCT1	CYCT1							7								
22	CYCT2	CYCT2							7								
23	CEAD	CEAD											11				
24	CURV	CURV															
26	DDR	DDR								8							
27	DDR1	DDR1												12			
28	DDR2	DDR2												12			
29	DDRM	DDRM												12			
30	DECOMP	DDECOMP							7								
31	DIAGONAL	DIAGON															15
32	DPD	DPD						6									
33	DSCHK	DSCHK							7								
34	DSMG1	DSMG1															
35	DSMG2	DSMG2				4									13		
37	DUMMOD1	DUMMOD1							7								
38	DUMMOD2	DUMMOD2							7								
39	DUMMOD3	DUMMOD3							7								
40	DUMMOD4	DUMMOD4							7								
42	EMA1	EMA1								8							
43	EMG	EMG								8							
44	FA1	FA1											11				
45	FA2	FA2											11				

NASTRAN ON THE CDC 6000/CYBER

Figure 19. Module Link Specification Table

5.5-31 (12/29/78)

MODULE INDEX	DMAP NAME OF MODULE	MODULE ENTRY POINT NAME	LINKS	MODULE RESIDES IN	ON	6600			
46	FBS	DFBS			7				
47	FRLG	FRLG						10	
48	FRRD	FRRD						10	
50	GI	GI					9		
51	GKAD	GKAD						10	
52	GKAM	GKAM						10	
53	GP1	GP1	2						
54	GP2	GP2	2						
55	GP3	GP3	2						
56	GP4	GP4		4					
57	GPCYC	GPCYC			7				
58	GPFDR	GPFDR							13
59	GPSP	GPSP		4					
60	GPWG	GPWG		4					
62	INPUT	INPUT	2						
63	INPUTT1	INPTT1	2						
64	INPUTT2	INPTT2	2						
65	INPUTT3	INPTT3	2						
66	INPUTT4	INPTT4	2						
67	MATGEN	MATGEN			7				
68	MATGPR	MATGPR					8		
69	MATPRN	MATPRN					8		
70	MATPRT	PRTINT					8		
71	MCE1	MCE1		4					
72	MCE2	MCE2		4					
73	MERGE	MERGE1			7				
75	MODA	MODA			7				
76	MODACC	MODACC							12
77	MODB	MODB			7				
78	MODC	MODC			7				
79	MPYAD	DMPYAD			7				
80	MTRXIN	MTRXIN						10	
81	QFP	QFP							14
82	OPTPR1	OPTPR1	2						
83	OPTPR2	OPTPR2					8		
85	OUTPUT	OUTPT							14
86	OUTPUT1	OUTPT1							14
87	OUTPUT2	OUTPT2							14

Figure 19. Module Link Specification Table (Continued)

5.5-32 (12/29/78)

[illegible]

Figure 19. Module Link Specification Table (Continued)

5.5-33 (12/29/78)

15
15
15
15
15
15
15
15
15
15
15
15

14
14

5.5-34 (12/29/78)

MODULE - DMAP NAME - MODULE ENTRY - LINKS MODULE RESIDES IN ON 6600
 INDEX OF MODULE POINT NAME

175	SDCMPS	DDCMPS					
176	LODAPP	LODAPP		7			
177	GPSPC	GPSPC					15
178	EOMCK	EOMCK	4				
179	ADR	ADR				12	
180	FRRD2	FRRD2				10	
181	GUST	GUST				10	
182	IFT	IFT				10	
183	LAMX	LAMX				10	
184	MTRXTEST	MTRXTS			9		
185	EMA	EMA		8			14

NASTRAN ON THE CDC 6000/CYBER

Figure 19. Module Link Specification Table (Continued)

MATERIAL PREVIOUSLY ON PAGES 5.5-36 THROUGH 5.5-54 HAS BEEN DELETED.

ADRI				10	
ADRPRT				10	
ADRX				10	
ADR				10	
AIS	5				13
AI	5				13
ALLMAT					
AMATRX					11
AMGRFS					13
AMGMN				9	
AMGP2				9	
AMGR00				9	
AMGSBA				9	
AMGX				9	
AMG				9	
AMPA1X				9	
AMPA				9	
AMPR1X				9	
AMPR1				9	
AMPR2X				9	
AMPR2				9	
AMPR				9	
AMPCOM				9	
AMPC1X				9	
AMPC1				9	
AMPC2				9	
AMPC				9	
AMPD1X				9	
AMPD				9	
AMPFX				9	
AMPE				9	
AMPFX				9	
AMPE				9	
AMP				9	
APDCS				9	
APDF				9	
APDDE				9	
APDR				9	
APD12C				9	
APD1				9	
APD1C				9	
APD1D				9	
APD12				9	
APD2				9	
APD3				9	
APD4				9	
APD5				9	

Figure 20. Alphabetical Deck Name - Link Table

APD								9												
ARRM						6														
ASCM01	1																			
ASCM02	1																			
ASCM03	1																			
ASCM04	1																			
ASCM05	1																			
ASCM06	1																			
ASCM07	1																			
ASCM08	1																			
ASCM09	1																			
ASCM10	1																			
ASCM11	1																			
ASCM12	1																			
ASCM13	1																			
ASDRD	1																			
ASDMAP	1																			
ASDZZZ	1																			
ASPRD	1																			
ATEIG													11							
AUTDCK	1																			
AUTDCM	1																			
AUTDHD	1																			
AUTDSM	1																			
AUTDSV	1																			
AXIS10		2																		
AXIS3		2																		
AXIS		2																		
BARD									8											
BARS									8											
BAR							5													
BASGLR							5													
RCB			3				5		8											
RDAT01																13				15
RDAT02																				15
RDAT03																				15
RDAT04																				15
RDAT05																				15
RDAT06																				15
RFSMAT										9										
RINT							5													
RISHFL		2														13				
RISLNC	1	2	3				5	7	8	9	10		12							
RITPAT													12	13						
RITPOS	1	2	3	4	5	6	7	8	9	10	11	12	13	14						15
RLANK	1	2	3	4	5	6	7	8	9	10	11	12	13	14						15
BMGTNS																				
BMGZZZ																				

Figure 20. Alphabetical Deck Name - Link Table (Continued)

RMG										10					
RORDER		2													
RTSTPP	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RUG	1								9	10					15
RVISC			3					8		10					15
CALCV				4	5	6	7		9	10	11	12			15
CASCOR										10					
CASE										10					
CDCMPX							7		9	10	11	12			15
CDCOMP							7		9	10	11				15
COETMX											11				
COETM2											11				
COETM											11				
COIFBS											11				
COIVID											11				
COIFBS											11				
CFADA1											11				
CFAD1A											11				
CFAD1X											11				
CEAD											11				
CENTRE		2									11				
CFACTR									9	10	11				
CFACTX									9	10	11				
CFBSOR									9	10	11				
CFBSRX									9	10	11				
CFCNTL										10	11				
CFEER1											11				
CFEER2											11				
CFEER3											11				
CFEER4											11				
CFER3D											11				
CFEP3S											11				
CFE1AN											11				
CFE1MY											11				
CFE2AD											11				
CFE2MY											11				
CFNOR1											11				
CFNOR2											11				
CF1FBS											11				
CF1ORT											11				
CF2FBS											11				
CF2ORT											11				
CHARQ4		2									11				
CHKOPN															
CHRDW		2												15	
CINFBS											11				
CINFRX											11				
CINVPR											11				

Figure 20. Alphabetical Deck Name - Link Table (Continued)

NASTRAN - OPERATING SYSTEM INTERFACES

CINVPX											11			
CINVP1											11			
CINVP2											11			
CINVP3											11			
CINVPX											11			
CINVIX											11			
CINV2X											11			
CINV3X											11			
CLSTAR	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CLSTRS	2	3		5			8				12	13		
CLVEC										11				
CMAUTO														15
CMBFND														15
CMR777														15
CMR001														15
CMR002														15
CMR003														15
CMR004														15
CMCASE														15
CMCKCD														15
CMCKDF														15
CMCDMB														15
CMCNT														15
CMDISC														15
CMHGEN														15
CMIWRT														15
CMMCNN														15
CMRD2A														15
CMRD2B														15
CMRD2C														15
CMRD2D														15
CMRD2E														15
CMRD2F														15
CMRD2G														15
CMRD2Z														15
CMRD2														15
CMRELS														15
CMSEIL														15
CMSOFO														15
CMTIMU										11				
CMTQC														15
CMTRCF														15
CNORM1										11				
CNORM										11				
CNSTRC	2													
COMAIN														
COMRO	1				5									
COMR1														15

Figure 20. Alphabetical Deck Name - Link Table (Continued)

CDMA2X															
CDMA2															15
CDMECT		2													15
CDM12							7		9	10	11				
CDNDAD	1	2	3	4	5	6	7	8	9	10	11				15
CDNDAS	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CDNE					5						11	12	13	14	15
CDNMSG	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CDNM1D								8							
CDNM1S								8							
CDNM2D								8							
CDNM2S								8							
CONTOR															
COPYZZ		2													
COPY							7								
CDPSZ							7								
CPYFIL	1	2	3	4	5	6	7	8	9	10					
CPYSTR	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CPORD2				4											
CPORD				4											
CPREATE		2													
CPIGGP				4											
CRSUB															
CSORTX															15
CSUR											11				
CSUMM											11				
CTRNSP											11				
CURCAS							7		9	10	11				15
CURVC1												12			
CURVC2													13		
CURVC3													13		
CURVIT													13		
CURVPS													13		
CURVTR													13		
CURVXX													13		
CURV1													13		
CURV2													13		
CURV3													13		
CURV													13		
CXLDDP							7		9	10	11				
CXTRNY											11				15
CYCT1Z							7				11				
CYCT1							7								
CYCT2A							7								
CYCT2B							7								
CYCT2X							7		9	10	11	12			
CYCT2							7								
DACHAR							7								

Figure 20. Alphabetical Deck Name - Link Table (Continued)

NAME	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DADD5							7								
DADD				4			7								
DADDA				4			7								
DADPTR			3		5			8							
DARCOS								8							
DAXR			3		5			8							
DEAP													13		
DCD													13		
DCOMPX				4	5	6	7		9	10	11	12			15
DCOHF													13		
DDCMPS							7								
DDCOMP							7								
DDRA1												12			
DDRR1												12			
DDRMC1												12			
DDRMMA												12			
DDRMMP												12			
DDRMMS												12			
DDRMMX												12			
DDRMM1												12			
DDRMM2												12			
DDRMM												12			
DDR1A												12			
DDR1R												12			
DDR1X												12			
DDR1												12			
DDR2												12			
DDR								8							
DDUM1													13		
DDUM2													13		
DDUM3													13		
DDUM4													13		
DDUM5													13		
DDUM6													13		
DDUM7													13		
DDUM8													13		
DDUM9													13		
DECDDE								8					13		15
DECOMP				4	5	6	7			10	11				
DECP5X							7								
DFCP1X							7								
DFCP2X							7								
DFCP3X							7								
DEFILF	1														
DELFTE															15
DELKLS								8							
DELSET		2	3		5			8					13		
DFLTKL			3					8							

Figure 20. Alphabetical Deck Name - Link Table (Continued)

DESCRIPT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DETCXX								8							
DETCX			3												
DETDET						6									
DETDX						6									
DETFRS						6									
DETMX						6									
DETM1						6									
DETM3						6									
DETM4						6									
DETM5						6									
DETM						6									
DEBS1X						6									
DEBS2X							7								
DEBS							7								
DIAGDN							7								
DIAGXX															15
DIHEX															15
DISPLA		2										13			
DKINT			3												
DKI			3					8							
DKLS								8							
DKL			3					8							
DK100			3					8							
DK211			3					8							
DK89			3					8							
DLAMBY									9						
DLAMG									9						
DLAXX									9						
DLBDY									9						
DLRPT2									9						
DLRXX									9						
DLCOM									9						
DLH									9						
DLDDP				4	5	6	7			10	11				
DLPT2										9					
DLP2X										9					
DMATRX			3												
DMFGR		2						8							
DMINT			3												
DMI			3												
DMPEIL															
DMPY									9	10					15
DMPYAD							7								
DMPYX							7								
DM100			3				7								
DM211			3												
DM89			3												

Figure 20. Alphabetical Deck Name - Link Table (Continued)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DDWN						6									
DDPAA						6									
DDPCOM						6									
DDPCOR						6									
DDP01						6									
DDP02						6									
DDP03						6									
DDP04						6									
DDP05						6									
DDP0						6									
DDPLNT		2													
DDPLTST		2													
DDPPS									9						
DDPPS8									9						
DDPZY									9						
DDDMEM													13		
DDUAD													13		
DDRAW		2													
DDRND													13		
DDRWCHR		2													
DDWDAT		2													
DDSCHKX							7								
DDSCHK							7								
DDSHFAR													13		
DDSMG1													13		
DDSMG2X				4											
DDSMG2				4											
DDSTRDY															15
DDSLAXX													13		
DDSLA													13		
DDSLAAA													13		
DDSLADP													13		
DDSLAET													13		
DDSLR													13		
DDSLFTD													13		
DDSLFTT													13		
DDSLX													13		
DDSL													13		
DDTBF													13		
DDTRANP							7								
DDTRANX				4	5	6	7		9						
DDTPRSC													13		
DDTRIA													13		
DDTRMEM													13		
DDTSHLD													13		
DDTSHLS													13		
DDUMERG							7								
DDUMPD1							7								

Figure 20. Alphabetical Deck Name - Link Table (Continued)

DUMD02							7								
DUMD03							7								
DUMD04							7								
DUMPER	1														
DUMPRM															
DUM1XX							7							15	
DUM1					5										
DUM2XX							7								
DUM2					5										
DUM3XX							7								
DUM3					5										
DUM4XX							7								
DUM4					5										
DUM5					5										
DUM6					5										
DUM7					5										
DUM8					5										
DUM9					5										
DUPART							7								
DVECTR		2													
DYPZ															
DZPY									9						
DZYMAT									9						
DZY									9						
EADD						6									
ECTLNC		2													
EDIT							8								
EDTL					5									15	
EGNVCT															
FJDUM2	1	2	3	4	5	6	7	8	9	10	11	11	12	13	14
EJECT		2						8	9				12	13	15
EKTRBD								8							
EKTRBS								8							
EKTRMD								8							
EKTRMS								8							
ELFLBL		2						8							
FLIMX				4											
FLIM				4											
FMADTO								8							
EMASTO								8							
EMAXXX								8							
FMA10								8							
EMA15								8							
FMA1XX								8							
EMA1								8							
FMA								8							
EMGCNG								8							
FMGCOR								8							

Figure 20. Alphabetical Deck Name - Link Table (Continued)

NASTRAN - OPERATING SYSTEM INTERFACES

[illegible]

FMGNDIC
FMGFRST
FMGFIL
FMGFIN
FMGOLD
FMGOUT
FMGPRM
FMGPPN
FMGSNO
FMGTAR
FMGTRX
FMGXXX
FMGX01
FMGX02
FMGX03
FMGX04
FMGX05
FMGX06
FMGX07
FMGX08
FMGX09
FMGX10
FMGX11
FMGX12
FMGX13
FMGX14
FMGX17
FMGX18
FMGX19
FMGX20
FMGX21
FMGX22
FMGX23
FMGX24
FMGX25
FMGX26
FMGX27
FMGX28
FMGX29
FMGX30
FMGX31
FMGX32
FMGX33
FMGX34
FMGX35
FMGX36
FMGX37
FMGX38

Figure 20. Alphabetical Deck Name - Link Table (Continued)

FMGX39								8								
FMGX40								8								
FMGX41								8								
FMGX42								8								
FMGX43								8								
FMGX44								8								
FMGX45								8								
FMGX46								8								
FMGX47								8								
FMGX48								8								
FMG1RX								8								
FMG1R								8								
FMG								8								
FMPCDP						6										
EMSG									9							
EMTRRD								8								
FMTRBS								8								
ENCDDF																
ENDSSS	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
ENDSYS	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
EOMCKM												12				
EOMCKS												12				
EOMCK												12				
EOMKS												12				
EONUT1												12				
EOSCOD																15
EOSOF																15
EOSOUT																15
ERRMKN																15
FSFA	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
FSORT	1															15
ESTOUT					5											
EXFORT																
EXIN1X																15
EXIN1																15
EXIN2F																15
EXIN2P																15
EXIN2X																15
FXI02																15
EXIN																15
FXI2																15
FXLVL																15
EXN2																15
EXTERN					5											15
FACTOR				4	5	6										
FACTRU											11					
FACTPX				4	5	6					11					
FA1KXX											11					
											11					

Figure 20. Alphabetical Deck Name - Link Table (Continued)

FA1K						11			
FA1KF						11			
FA1PKA						11			
FA1PKC						11			
FA1PKF						11			
FA1PKI						11			
FA1PKV						11			
FA1PKX						11			
FA1XX						11			
FA1						11			
FA2Y						11			
FA2						11			
FRSINT						11			
FRSI	4	5	7		10	11	12		
FRSX	4	5	7			11	12		15
FRS1	4	5	7		10	11	12		15
FRS21						11			
FRS2									15
FRS3	4	5	7		10	11	12		15
FRS4	4	5	7			11	12		15
FRS	4	5	7		10	11	12		15
FCNTL				6					
FCTRUX						11			
FCUPL		5							
FDIT									15
FDNAME									15
FDSUB									15
FDFVCT				6					
FEERAA						11			
FEERCX				6					
FEEPXC						11			
FEERXX				6					
FEERY				6					
FEER7C						11			
FEER71						11			
FEER72						11			
FEER73						11			
FEER74						11			
FEER1X				6					
FEER1				6					
FEER2X				6					
FEER2				6					
FEER3Y				6					
FEER3				6					
FEER4X				6					
FEER4				6					
FF100		5							
FILCOR				6					

Figure 20. Alphabetical Deck Name - Link Table (Continued)

5.5-66 (12/29/78)

FILSWI				4	5	6	7		9	10	11	12		15
FIND		2												
FINDC				4	5	6	7		9	10	11			15
FINDER														15
FLLD									9					15
FMDI														
FNAME	1	2	3	4	5	6	7	8	9	10	11	12	13	14
FNDGRD														15
FNDLVL														15
FNDNXL														15
FNDPAR		2	3	4	5	6	7	8	9	10	11	12	13	14
FNDPLT	1	2												15
FNDPNT					5									
FNDSET		2												
FNXTVC														
FNXTV						6								
FNXT						6								
FORFIL		2												15
FORMAT								8					14	
FORMGG				4										
FORMG2				4										
FORM12														
FORM1											11			
FORM22											11			
FORM2											11			
FPONT					5						11			
FPT					5									
FORWV														
FORW						6								
FPRK2						6								
FPRK						6								
FRD2AX						6								
FRD2A										10				
FRD2RX										10				
FRD2B										10				
FRD2CX										10				
FRD2C										10				
FRD2DX										10				
FRD2D										10				
FRD2EX										10				
FRD2E										10				
FRD2I										10				
FREAD	1	2	3	4	5	6	7	8	9	10	11	12	13	14
FPLG										10				15
FRMAX										10				
FRMLTA						6								
FRMLTD						6								
FRMLTX						6								

Figure 20. Alphabetical Deck Name - Link Table (Continued)

FPMLT					6										
FPPDA1										10					
FPRDA1										10					
FPRDC1										10					
FPRDC2										10					
FPPDC3										10					
FPRDD1										10					
FPRDD2										10					
FPRDF1										10					
FPRDST										10					
FRRD1A										10					
FRRD1B										10					
FRRD1C										10					
FRRD1D										10					
FRRD1F										10					
FPRD2X										10					
FPRD2										10					
FPRD										10					
FPR1A1										10					
FPSW2					6										
FPSW					6										
FWMW									9						
FZY2									9						
F6211				5									13		
FR9				5									13		
GENDSP									9						
GFND									9						
GENFLX			4												
GENFLY			4												
GENVEC			4	5	6	7			9	10	11			15	
GFTBLK														15	
GETDFF		2													
GFRSX			4	5		7			9	10	11			15	
GFBS			4	5		7			9	10	11			15	
GICDM									9						
GJGGKS									9						
GIGGY									9						
GIGTKA									9						
GIGTKG									9						
GINDX	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
GIPSSY									9						
GIPSY									9						
GIVN						6									
GJ									9						
GKADA1										10					
GKAD1A										10					
GKAD1C										10					
GKAD										10					

Figure 20. Alphabetical Deck Name - Link Table (Continued)

Figure 20. Alphabetical Deck Name - Link Table (Continued)

5.5-69 (12/29/78)

[illegible]

[illegible]

Figure 20. Alphabetical Deck Name - Link Table (Continued)

Figure 20. Alphabetical Deck Name - Link Table (Continued)

5.5-71 (12/29/78)

[illegible]

NASTRAN - OPERATING SYSTEM INTERFACES

THOSSCNH											11				
THOSSCN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
THOSSOPT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
THOSTNCT		2	3	4	5	6	7	8	9	10	11	12	13	14	15
INCRPF										10	11				
INCRD									9						
INFRSX						6					11				
INITL2											11				
INITL											11				
INITX											11				
INPTT1		2													
INPTT2		2													
INPTT3		2													
INPTT4		2													
INPUA		2													
INPUTX		2													
INPUIT		2													
INP1XX		2													
INP2XX		2													
INTFBS											11				
INTLST		2													
INTPPT								8							
INTVEC		2													
INT2AL														13	
INVERD		2	3	4	5			8					13		
INVERS				4	5			8					13		
INVFR						6			9	10	11		13		15
INVFB											11				
INVPWR						6									
INVPWX						6									
INVPXX						6									
INVPX						6									
INVP1X						6									
INVP1						6									
INVP2V						6									
INVP2X						6									
INVP2						6									
INVP3X						6									
INVP4X						6									
INVPTX						6									
INVTRX						6									
INVTR						6									
ISFT	1														
ITCDEF															15
ITEMDT															15
ITMPRT															15
ITTYPE															15
IUNINN		2													15

Figure 20. Alphabetical Deck Name - Link Table (Continued)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
KPAR			3												
KCONFY			3					8							
KCONFY			3					8							
KCONFZ			3					8							
KCONE			3					8							
KDS									9						
KDUM1			3					8							
KDUM2			3					8							
KDUM3			3					8							
KDUM4			3					8							
KDUM5			3					8							
KDUM6			3					8							
KDUM7			3					8							
KDUM8			3					8							
KDUM9			3					8							
KELAS			3					8							
KFLUID2			3					8							
KFLUID3			3					8							
KFLUID4			3					8							
KHBDY			3					8							
KIHFX			3					8							
KORSZ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
KPANFL			3					8							
KPLTST			3					8							
KODMEM			3					8							
KODMM1			3												
KODM2D			3												
KODPLT			3					8							
KRND			3												
KSLNT			3					8							
KSOLID			3					8							
KTFTRA			3					8							
KTORDR			3					8							
KTPZ			3												
KTRAPR			3					8							
KTRBSC			3					8							
KTRIA			3												
KTRIQD			3					8							
KTRIRG			3					8							
KTRMEM			3					8							
KTRM6D								8							
KTRM6S								8							
KTRPLD								8							
KTRPLS								8							
KTRPLT			3					8							
KTSHLD								8							
KTSHLS								8							
KTURF			3					8							

9 9

5.5-74 (12/29/78)

NAME	UNIT	STATUS	REMARKS
MATPRG		10	
MATPPN	8		
MATPRT	8		
MATVFC2		11	
MATVFC		11	
MATWRT			
MA1XX	8		15
MBAMGX		9	
MRAMG		9	
MRAR	3		
MRRSLJ		9	
MRCAP		9	
MRCR1		9	
MRCR2		9	
MRDPDH		9	
MRGAF		9	
MRGATE		9	
MRGAW		9	
MRGFND		9	
MRMNDP		9	
MRDPA		9	
MRDPC		9	
MRPLOT		9	
MRPRIT		9	
MRREG		9	
MCBAR	3		
MCEA1		4	
MCEB1		4	
MCEC1		4	
MCE01		4	
MCE1		4	
MCE1A		4	
MCE1B		4	
MCE1C		4	
MCE1D		4	
MCF2		4	
MCONF	3		
MCONMX	3	8	
MCRDD	3	8	
MDUM1	3		
MDUM2	3		
MDUM3	3		
MDUM4	3		
MDUM5	3		
MDUM6	3		
MDUM7	3		
MDUM8	3		
MDUM9	3		

Figure 20. Alphabetical Deck Name - Link Table (Continued)

MFRGFD								9		11					
MERGE1						7									
MERGE					5	6	7	9	10	11	12			15	
MESSAGE	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MFLUID2			3					8							
MFLUID3			3					8							
MFLUID4			3					8							
MFRFE			3					8							
MHRDY			3					8							
MIHFX			3												
MINMAX		2													
MINTRP										10	11				
MINV			3		5			8					13		
MODA							7								
MODACC												12			
MODACX												12			
MODAC1												12			
MODAC2												12			
MODAC3												12			
MODR							7								
MODC							7								
MODOMP	1														
MODEL	1														
MPLPRT	1														
MPRTX								8							
MPYADX				4	5	6	7			10	11	12			15
MPYADZ				4	5	6	7		9	10	11	12			15
MPYAD				4	5	6	7		9	10	11	12			15
MPYA1D							7								
MPYA2D							7								
MPYA3D							7								
MPYL					5										
MPY0				4	5	6	7		9	10	11	12			15
MPY3A							7								15
MPY3R							7								15
MPY3C							7								15
MPY3DR							7								15
MPY3NU							7								15
MPY3OC							7								15
MPY3P							7								15
MPY3TL							7								15
MPY3ZZ							7								15
MPY3							7								
MPY3CP							7								15
MPY3IC							7								15
MPZDA			3												
MODPLT			3					8							
MPEDZ7															15

Figure 20. Alphabetical Deck Name - Link Table (Continued)

MTRXXX

(d)

NASTRAN - OPERATING SYSTEM INTERFACES

MTURE			3				7								
MXCIDS							7								
MXCID							7					12			
NAMES	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
NASCAR	1														
NORM11						6									
NORM1						6									
NTIMEX	1														
NTIME	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OCIBCD											11				
ODUM1														14	
ODUM2														14	
ODUM3														14	
ODUM4														14	
ODUM5														14	
ODUM6														14	
ODUM7														14	
ODUM8														14	
ODUM9														14	
ODUM														14	
OFPR9														14	
OFPRD1														14	
OFPRD5														14	
OFPR1														14	
OFPR2														14	
OFPR3S														14	
OFPR3														14	
OFPR4														14	
OFPR5														14	
OFPR6														14	
OFPRB7S														14	
OFPR7														14	
OFPR8														14	
OFPCF1														14	
OFPCF2														14	
OFPCOM														14	
OFPCS1														14	
OFPCS2														14	
OFPMIS														14	
OFPPNT														14	
OFPPUN														14	
OFPRF1														14	
OFPRF2														14	
OFPRS1														14	
OFPRS2														14	
OFPSN1														14	
OFPS1														14	
OFPSXX														14	

Figure 20. Alphabetical Deck Name - Link Table (Continued)

NAME	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PCP1															
PCP1A															14
PCP															14
PCRF2S															14
PCRS2S															14
PCSNJ															14
PCSS1															14
ONEIWD				4	5	6	7			10	11				14
PRINV						6									
PRMSG		2													
PRTPR1		2													14
PRTPR2								8							
PRTPW1		2													
PRTPW2								8							
PRTPX1		2													
PRTPX		2													
PRTP1A		2													
PRTP1R		2													
PRTP1C		2													
PRTP1D		2													
PRT2A								8							
PRT2R								8							
PRT2C								8							
PRT2D								8							
PRDER		2													
PRCK						6									
PRTHD															
QSCENT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
QSCXRF	1										11				
QUTPT1															
QUTPT2															14
QUTPT3															14
QUTPT4															14
QUTPT															14
QUTPUT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
QUT1XX															
QUT2XX															14
QUT3XX															14
PARS															14
PACKX	1	2	3	4	5	6	7	8	9	10	11				
PAGE	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PARAML		2	3	4	5	6	7	8	9	10	11	12	13	14	15
PARAM		2													
PARMEG															
PARMLX		2	3	4	5	6	7	8	9	10	11	12			15
PARTN1							7	8	9	10	11	12	13	14	15
PARTN2							7								
PARTN3							7								

Figure 20. Alphabetical Deck Name - Link Table (Continued)

5.5-79 (12/29/78)

PARTN				4	5		7		9	10	11	12			15
PASSER	1														
PATX				4	5	6	7		9	10	11	12			15
PERMUT					5										
PERPEC		2													
PFXIT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PHDMIA														14	
PHDMIX														14	
PKRAP														14	
PKQAD1													13		
PKQAD2													13		
PKQDMS													13		
PKQDM1													13		
PKQDM													13		
PKQDPL													13		
PKRND													13		
PKT01													13		
PKT02													13		
PKTRRS													13		
PKTRI1													13		
PKTRI2													13		
PKTRMS													13		
PKTPM1													13		
PKTRM													13		
PKTRPL													13		
PKTROD													13		
PKTP02													13		
PLAGP													13		
PLAMAT													13		
PLA1			3												
PLA2X													13		
PLA2													13		
PLA3ES													13		
PLA3UV													13		
PLA31X													13		
PLA31													13		
PLA32C													13		
PLA32F													13		
PLA32S													13		
PLA32X													13		
PLA32													13		
PLA3													13		
PLA4R													13		
PLA4FS													13		
PLA4UV													13		
PLA41X													13		
PLA41													13		
PLA42C													13		

Figure 20. Alphabetical Deck Name - Link Table (Continued)

[illegible]

Figure 20. Alphabetical Deck Name - Link Table (Continued)

[illegible]

Figure 20. Alphabetical Deck Name - Link Table (Continued)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RETURN															
PF2AL								8			11				
PFDRCE					5										
RMGZZZ					5										
RMG					5										
RNDN								8							
RNDS								8							
RND					5										
RDMRDK			3					8							
RDMRER					5								13		
RDTAT		2													
RDTAX						6									
RWDYZZ									9						
RSORT											11				
PSTYXX		2													
PULFR				4	5	6	7		9	10	11	12			15
SADNX				4		6	7		9	10					15
SADD				4		6	7		9	10	11				15
SADQTR			3		5			8	9				13		
SAXB			3		5			8	9				13		
SAXIF1													13		
SAXIF2													13		
SPAR1													13		
SPAR2													13		
SPSPL2													13		
SCALAR															15
SCALED								8							
SCALFY				4											
SCALXX															15
SCE1				4											
SCONE1													13		
SCONE2													13		
SCONE3													13		
SCRLM													13		
SDCINS							7								
SDCIN				4	5	6	7			10	11				15
SDCMMX							7								
SDCMM							7								
SDCMPS							7								
SDCMO							7								
SDCNMP				4	5	6	7			10	11				15
SDCNMX				4	5	6	7			10	11				15
SDCNM1				4	5	6	7				11				15
SDCNM2										10					15
SDCNM3				4	5	6	7			10	11				15
SDCNM4				4	5	6	7			10	11				15
SDCNM				4	5	6	7			10	11				
SDCPUT				4	5	6	7			10	11				15

Figure 20. Alphabetical Deck Name - Link Table (Continued)

[illegible]

Figure 20. Alphabetical Deck Name - Link Table (Continued)

SDUM72																	13
SDUM81																	13
SDUM82																	13
SDUM91																	13
SDUM92																	13
SD2RHD																	13
SEEMAT		2															
SEEMTX		2															
SELAS1																	13
SFLAS2																	13
SELCAM		2															
SEMINI	1																
SEMTRN	1																
SFM	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		
SFTFO																	15
SETFND												12	13				15
SETINP		2															
SETIVL																	15
SETTIM	1																
SFTUP	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		
SETVAL		2	3	4	5	6	7	8	9	10	11	12	13	14	15		
SFACT				4	5	6	7				11						15
SFETCH																	15
SGENZZ																	15
SGEN																	15
SGFNA																	15
SGENB																	15
SGENM																	15
SGINQ1		2															
SGINQ		2															
SHAPE		2															
SHEARD								8									
SHEARS								8									
SICDX						6											
SIHEX1																	13
SIHEX2																	13
SJUMP																	15
SKPFPM		2															
SKPREC	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		
SMAR3				4													
SMAC3				4													
SMA1X			3														
SMA1			3														
SMA1A			3														
SMA1RK			3					8									
SMA1R			3														
SMA1CL			3					8									
SMA1DP			3					8									

Figure 20. Alphabetical Deck Name - Link Table (Continued)

SMA1ET			3					8									
SMA1HT			3					8									
SMA1IN			3					8									
SMA2X			3														
SMA2			3														
SMA2A			3														
SMA2PK			3					8									
SMA2P			3														
SMA2CL			3					8									
SMA2DP			3					8									
SMA2ET			3					8									
SMA2HT			3					8									
SMA2IO			3					8									
SMA3				4													
SMA3A				4													
SMA3B				4													
SMA3C				4													
SMLFIG					6												
SMMATS																	
SMPYAD							7					13					
SMP1				4													
SMP2				4													
SMSG																	
SNPDF								9								15	
SNFCLS																15	
SNFCOM	1	2	3	4	5	6	7	8	9	10	11	12	13	14		15	
SNFINT																15	
SNFIOF																15	
SNFINI	1															15	
SNFIN																	
SNFI																15	
SNFOPN																15	
SNFN																15	
SNFSIZ																15	
SNFTOC																15	
SNFTRL																15	
SNFUTX																15	
SNFUT																15	
SNF																15	
SOLID					5											15	
SOLVER				4													
SOLVE1																	
SOLVE													13				
SOLVRX							7										
SOLV1X				4													
SOLV2X							7										
SOLV3X							7										
SOLV4X							7										

Figure 20. Alphabetical Deck Name - Link Table (Continued)

SOLV5X							7								
SNRT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SNRTCM															15
SNUT															
SPANL1													13		
SPANL2													13		
SPLT10													13		
SODME1															15
SODM11													13		
SODM12													13		
SODM21													13		
SODM22													13		
SODPL1													13		
SOPTM						6							13		
SRND1													13		
SRND2													13		
SSGA1X					5										
SSGA2					5				9	10	11	12			
SSGA3			4		5							12			
SSGR1X					5										
SSGR2			4		5	6	7		9	10	11	12			15
SSGC2			4						9	10	11				15
SSGETD					5										
SSGETT					5										
SSGHTP					5										
SSGHTZ					5										
SSGHT1					5										
SSGHT2					5										
SSGHT					5										
SSGKHI					5										
SSGSLT					5										
SSGTRI					5										
SSGWRK					5										
SSG1					5										
SSG1A					5										
SSG2A					5				9	10	11	12			
SSG2X					5										
SSG2					5										
SSG2B			4		5	6	7		9	10	11	12			15
SSG2C			4						9	10	11				15
SSG3					5										
SSG3A			4		5										
SSG4					5							12			
SSL0T1													13		
SSL0T2													13		
SSOLD1													13		
SSOLD2													13		
SSPLIN									9		11		13		

Figure 20. Alphabetical Deck Name - Link Table (Continued)

SSWTC	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
STAPID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
STFP2											11		13	14	15
STFP											11				
STIME	1	2	3	4	5	6	7	8	9	10	11				
STOPD1											11	12	13	14	15
STOPD2													13		
STPAIC													13		
STPAX1									9						
STPAX2													13		
STPAX3													13		
STPRG													13		
STPRSO									9						
STPBS1									9						
STPDA									9						
STPK									9						
STPLOT		2							9						
STPOHI															
STPDT2									9						
STOME2									9						
STRAP1													13		
STRAP2													13		
STRAX1													13		
STRAX2													13		
STRAX3													13		
STRAS1													13		
STRIPC													13		
STRIPX									9						
STRIR1									9						
STRIR2													13		
STRME1													13		
STPM61													13		
STPM62													13		
STRPL1													13		
STPPTS													13		
STRP11													13		
STRP12													13		
STROD1													13		
STROD2													13		
STRSLV													13		
STRSL1													13		
STRSL2													13		
STURE1													13		
SURR													13		
SURJ									9						
SUBP2									9						
SURP															
SUBPR									9						

Figure 20. Alphabetical Deck Name - Link Table (Continued)

[illegible]

Figure 20. Alphabetical Deck Name - Link Table (Continued)

TIMTS6									9											
TIMTS7									9											
TIMTSA									9											
TIM1XX									9											
TIM2XX									9											
TIM3XX									9											
TIM4XX									9											
TIM5XX									9											
TIM6XX									9											
TIM7XX									9											
TIM8XX									9											
TIPE		2							9											
TKFP									9											
TKTZTK			3																	
TLNOM6					5								13							
TLNDSL					5															
TLNDT1					5															
TLNDT2					5															
TLNDT3					5															
TMTDGN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15					
TPREFPX									9											
TPSWIT		2																		
TPZTFM					5									14						
TRAILF									9											
TPANEM																				
TRANP1				4	5	6			9				13							
TRANSP				4	5	6	7			10	11									
TPANX					5															
TPAPAD								8												
TPAPAX								8												
TPBSCD								8												
TPBSCS								8												
TPRSC					5															
TPDA1																				
TRDC1											11									
TRDD1											11									
TRDF1											11									
TRDYX											11									
TRD1A2											11									
TRD1A											11									
TRD1C2											11									
TRD1C											11									
TRD1D2											11									
TRD1D											11									
TRD1E											11									
TRD1Y											11									
TRD											11									
TPHTX											11									

Figure 20. Alphabetical Deck Name - Link Table (Continued)

Variable	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TRHT1A															
TRHT1B											11				
TRHT1C											11				
TPHT											11				
TRIAAD											11				
TRIAAX								8							
TRIDI						6		8							
TRIMEM					5										
TPIMEX					5										
TPIOD					5										
TPLGA					5										
TRLG1C					5										
TPLG					5										
TPLGB					5										
TRLGC					5										
TPLGD					5										
TPMEMD								8							
TPMEMS								8							
TRNSPX				4	5	6	7		9						15
TPNSP				4	5	6	7		9						15
TRPLT					5										
TRTTEM					5										
TSPL1D								8							
TSPL1S								8							
TSPL2D								8							
TSPL2S								8							
TSPL3D								8							
TSPL3S								8							
TTLPGE	1							8							
TTORDR					5										
TTRAPR					5										
TTRIPG					5										
TURED								8							
TURES								8							
TVOR									9						
TWISTD								8							
TWISTS								8							
TWO	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TYPE10		2													
TYPE3		2													
TYPE	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TYPELT		2													
TYPINT		2													
TYPOUT		2													
UMFFDT	1													14	
UMFXYX	1														
UMFZZZ	1														
UNPAKX	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Figure 20. Alphabetical Deck Name - Link Table (Continued)

UPARTX				4			7			10					
UPART				4			7			10					
USRMSG	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VALVFC						6									
VDRA												12			
VDRB												12			
VDRCONM												12			
VDRCONR												12			
VDR												12			
VECPRT								8							
VFCXXX							7								
VFC							7								
VISCD								8							
WALTIM	1														15
WJLVFC						6									
WPLT10		2													
WPLT3		2													
WPLT4		2													
WPTMSG		2													
WPTPRT		2													15
WPTTRL	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XCEITR	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XCEI	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XCHK	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XCLEAN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XCSA	1														
XCSARF	1														
XCSTM					5										
XDPH	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XDPL	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XFADJ1	1														
XFIAI	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XFIST	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XFLDEF	1														
XFLORD	1														
XFLSZD	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XGPIPS	1														
XGPIC	1														
XGPIDG	1														
XGPI0	1														
XGPI1	1														
XGPI2X	1														
XGPI2	1														
XGPI3	1														
XGPI4	1														
XGPI5	1														
XGPI6	1														
XGPI7	1														

Figure 20. Alphabetical Deck Name - Link Table (Continued)

YGP1P	1														
XGPI	1														
YTHEY							8								
YIPFL	1														
XLINK	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XLKSPC	1														
XLNKHD	1														
XMDMSK	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XNCTPN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XNLDPT	1														
XNSGFN	1														
YPAPAM	1														
XPFIET	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XPOLCK	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XPUMP	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XPURGF	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XRCARD	1														
XPFCPS	1														
YPGDFM	1														
XSAVE		2	3	4	5	6	7	8	9	10	11	12	13	14	15
XSRSET	1														
XSCNDM	1														
XSEM01	1														
XSEM02		2													
XSEM03			3												
XSEM04				4											
XSEM05					5										
XSEM06						6									
XSEM07							7								
XSEM08								8							
XSEM09									9						
XSEM10										10					
XSEM11											11				
XSEM12												12			
XSEM13													13		
XSEM14														14	
XSEM15															15
XFA1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
YFA		2	3	4	5	6	7	8	9	10	11	12	13	14	15
YNOPT	1														
XNSGN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XSTRD	1														
XTRNSY						6									
XTPNY1						6									
XVPS	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
YXFIAT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XYMPRT								8							
XYOPT1		2													

Figure 20. Alphabetical Deck Name - Link Table (Continued)

XXOPT2								8									
XXPARM		2															
XXPLNT		2															
XXPM5G		2															
XXPRTI								8									
XXPSFT		2															
XXVLVC						6											
XYCHAP																14	
XYDUMP																14	
XYFIND																14	
XYGRAF																14	
XYLOG																14	
XYQUIT																14	
XYPLIN		2														14	
XYPLNT		2															
XYPLXX		2															
XYPPPP																	
XYPRPL																14	
XYPRPT																14	
XYTICS																14	
XYTRAN																14	
XYTP2Z																14	
XYWORK																14	
ZAPD																14	
ZBLPKX	1	2	3	4	5	6	7	8	9		10	11	12	13	14	15	
ZEROC									9		10	11					
ZJ									9								
ZNTPKX	1	2	3	4	5	6	7	8	9		10	11	12	13	14	15	

Figure 20. Alphabetical Deck Name - Link Table (Continued)

5.5-95 (12/29/78)

6. MODIFICATIONS AND ADDITIONS TO NASTRAN

6.1 INTRODUCTION

Modifications and additions are continuously made to large programming systems. NASTRAN will not be an exception to this rule. Section 6.2 presents the FORTRAN IV language restrictions that must be followed in order to produce equivalent object code across the computing machines on which NASTRAN operates. The remaining sections discuss areas of the program which experience, gained during program development, has shown to be those areas most subject to modifications and additions.

6.2 FØRTRAN IV LANGUAGE RESTRICTIONS

NASTRAN was developed in the FØRTRAN IV programming language to the greatest extent possible in order to simplify the task of conversion from the development machine, which, for the majority of program development, was the IBM 7094/7040(44) DCS, to third generation computing systems. The same FØRTRAN IV code can execute differently across computing machines. This fact is all but too well known to those who have labored through the task of conversion from one computing system to another. For this reason, modifications and additions to NASTRAN must be accomplished with FØRTRAN code that will produce equivalent object code across computing machines. The basic set of rules governing programming in FØRTRAN IV for NASTRAN is incorporated in the following manual in the IBM Systems Reference Library: IBM 7090/7094 IBSYS Operating System, Version 13, FØRTRAN IV Language, File No. 7090-25, Form C28-6390-3. The following is a list of exceptions to the rules set forth in this manual.

1. An integer constant may not be greater than $2^{31}-1$.
2. Subscripted variables should contain no more than 3 subscripts.
3. A reference to the first variable in a subscripted array must contain the subscript 1, e.g., $A(1) = 0.0$.
4. A CØNTINUE statement requires a FØRTRAN statement number.
5. The PAUSE statement is not to be used.
6. The NAMELIST statement is not to be used.
7. Implied DØ's in DATA statements are not allowed.
8. The last statement of a DØ loop may not be a logical IF statement. It is recommended that it be a CØNTINUE statement. It is also recommended that each DØ loop have its own CØNTINUE statement.
9. BLØCK DATA subprograms may contain only type (e.g., REAL, INTEGER), DIMENSION, CØMMØN, DATA and comment statements.
10. All Hollerith data should be defined in the form 4H.....
11. Do not use octal (Ø) or hexadecimal (Z) in DATA or FØRMAT statements.
12. Specification statements should precede any executable statement.

MODIFICATIONS AND ADDITIONS TO NASTRAN

13. The order of specification statements should be as follows:

COMPLEX

DOUBLE PRECISION

REAL

INTEGER

LOGICAL

DIMENSION

COMMON

EQUIVALENCE

EXTERNAL

DATA

Arithmetic Statement Functions

14. The variables in blank COMMON or a block of COMMON should be ordered as follows:
complex, double precision, real, integer and logical.
15. Variables stored as single precision cannot be referenced as double precision variables (via the FORTRAN EQUIVALENCE statement) because of the different internal word storage format for single and double precision words on the Univac 1108.
16. Caution must be exercised to insure that types (REAL, INTEGER, etc.) of FORTRAN function values agree in the function subprogram and in the calling program. This agreement between types is necessary for machines (e.g., IBM S/360) on which REAL and INTEGER values of FORTRAN functions are returned in different registers.
17. Do not attempt to extend the length of arrays through the EQUIVALENCE statement.
18. Caution must be exercised when using the EQUIVALENCE statement. One should not use the EQUIVALENCE statement to give different variable names to the same cell, since modern compilers, because of their optimization techniques, do not guarantee that the values of the equivalenced variables will be the same. Hence EQUIVALENCE should be used only between variables which have non-intersecting use spans in a program.

FORTRAN IV LANGUAGE RESTRICTIONS

19. Nonstandard returns in a SUBROUTINE statement must immediately follow the left parenthesis which starts the names of the subroutine's arguments, e.g. SUBROUTINE XYZ (*,*,A,B) is the correct form; SUBROUTINE XYZ (*,A,*,B) is not acceptable. On the CDC computers, use SUBROUTINE XYZ(A,B),RETURNS(RETURN1,RETURN2).
20. There must be agreement with respect to the number of arguments and the type of each argument in the argument list of a calling program and the subprogram called.
21. For consistency with current NASTRAN practice, deck (or member) names for subroutines should agree with the primary entry point names. Deck names for Block Data subprograms should end with the characters "BD".
22. FUNCTION subprograms whose type is not implicit must be typed in the FUNCTION statement. For example, use

DOUBLE PRECISION FUNCTION ABC(X)

and not

FUNCTION ABC(X)

DOUBLE PRECISION ABC
23. The name of a FUNCTION subprogram must appear somewhere within the subprogram.
24. All subscripted variables appearing in EQUIVALENCE statements must be subscripted. E.g., use EQUIVALENCE (A(1),X(1)) instead of EQUIVALENCE (A,B).
25. DO loop indices may not be greater than $2^{17}-1$ (131,071). (This is a CDC 6600 restriction.)

THE EXECUTIVE CONTROL DECK

6.3 THE EXECUTIVE CONTROL DECK

The capabilities of the Executive Control Deck may be changed or increased by modifying existing control card functions or addition completely new card types. Executive control cards are processed within two modules of the Preface: XCSA, (Executive Control Section Analysis, Section 4.2) and XGPI (Executive General Problem Initialization, Section 4.7). Some cards are handled completely within XCSA, while others are only partially checked by XCSA and then passed to XGPI, via the Executive Control Table (Section 2.4.2.5), for final processing.

To modify the content or function of an existing control card, first locate the proper section within module XCSA. The block of FØRTRAN statements related to the processing of each type of card is appropriately commented. Also, it can be determined from these statements whether part of the processing is being passed to module XGPI. The required modifications can then be made within XCSA and/or XGPI.

To add a new control card type to those currently acceptable, the following steps should be taken. First, add the card type name to the local FØRTRAN array ECTT (Executive Control Type Table) within module XCSA, and increase the table length parameter (LECTT) by three for each new entry. The three word entry consists of two BCD words (4 characters/word) for the card type mnemonic and a one word integer flag indicating whether the card type is to be optional (=0) or required (=1) within the NASTRAN Executive Control Deck. Second, add a statement number to the computed-go-to branch vector. This branch vector transfers the XCSA logic to the correct card processing section within the module. Third, create the processing code, and add it to the module. If additional processing must be performed in XGPI, the Control Table format should be modified and the necessary logic added to the XGPI module.

6.3.1 The NASTRAN Card

A facility is provided whereby the default values in /SYSTEM/, which are initialized by the system Block Data subprogram, SEMDBD, or subroutine BTSTRP, can be altered at execution time. The contents of /SYSTEM/ are described in Section 2.4.1.8. Other locally used values may also be redefined.

The card which provides this capability is called the NASTRAN card. If this card is used, it must be the first card of the data deck (i.e., the card must precede the Executive Control

MODIFICATIONS AND ADDITIONS TO NASTRAN

Deck). The NASTRAN card is a free field card (similar to cards in the Executive Control and Case Control Decks). Its format is as follows:

NASTRAN keyword₁=value, keyword₂=value, ...

where the list of allowable keywords is as follows:

1. BUFFSIZE - Defines the number of words in a GINØ buffer. Note: fixed length records written by GINØ are of length BUFFSIZE - 3. This keyword changes the first word of /SYSTEM/.
2. MAXØPEN - Defines the maximum number of files that may be open at any one point in the program. This keyword changes the 30th word of /SYSTEM/.
3. CØNFIG - Defines the computer configuration for use in the timing equations in the matrix decomposition subroutines SDCØMP, DECØMP and CDCØMP. This keyword changes the 28th word of /SYSTEM/.
4. MAXFILES - Defines the maximum number of files to be placed in /XFIAT/ by GNFIAT. This keyword changes the 29th word of /SYSTEM/.
5. SYSTEM(I) - I refers to the Ith word of /SYSTEM/. This is a general form of altering any word in /SYSTEM/. Note that BUFFSIZE and SYSTEM(1) are equivalent, and MAXFILES and SYSTEM(29) are equivalent.
6. NLINES - Defines the 9th word of /SYSTEM/. This value sets the number of data lines per printed page. For 11" paper, an appropriate value is 50. For 8-1/2" paper, an appropriate value is 35.
7. TITLEØPT - Defines a local variable within SEMINT which is passed to TTLPGØ to control the printing of the NASTRAN title page. See Section 3.3.13 for a description of subroutine TTLPGØ.
8. MØDCØM(i) - Defines a nine-word array for module communications. Currently, only MØDCØM(1) is supported. If MØDCØM(1) = 1, diagnostic statistics from subroutine SDCØMP are printed.
9. HICØRE - Defines the amount of open core available to the user on the UNIVAC 1100 series machines. The user area default is nominally 65K decimal words. The ability to increase this value may be installation limited.

THE EXECUTIVE CONTROL DECK

10. FILES - Establishes NASTRAN permanent files as being disk files rather than tape files. The files are P00L, 0PTP, NPTP, UMF, NUMF, PLT1, PLT2, INPT, INP1, INP9. Multiple file names must be enclosed with parentheses such as FILES = (UMF,NPTP). The FILES parameter(s) must be last on the NASTRAN card. Note the plot files, PLT1 and PLT2, are not supported on the UNIVAC, therefore a physical tape must be assigned prior to job execution.
11. TRACKS - Defines track size of Calcomp plot tape if plotting is first to be done to disk. TRACKS = 7 implies a 7-track tape and TRACKS = 9 implies a 9-track tape.

Examples of use of the NASTRAN card follow.

```
NASTRAN    BUFFSIZE=878, SYSTEM(2)=3, MAXOPEN=10
```

The above card changes the 1st, 2nd and 30th words of /SYSTEM/. SYSTEM(2)=3 changes the system output unit from 6 to 3.

```
NASTRAN    SYSTEM(4)=4, MAXFILES=21
```

The above card changes the 4th and 29th words of /SYSTEM/. SYSTEM(4)=4 changes the system input unit from 5 to 4 (which means that all subsequent data must be present on unit 4).

THE CASE CONTROL DECK

6.4 THE CASE CONTROL DECK

The Case Control Deck is processed by the IFP1 module, whose Module Functional Description can be found in Section 4.3. A card can be added to the Case Control Deck by implementing the following steps.

1. Assignment of a Word in the CASECC Data Block (see Section 2.3.1.1).

If the card datum is to be passed on, a word must be assigned in CASECC. Several words are currently empty in CASECC. If more space in the fixed portion of CASECC is needed, modules which use CASECC to prepare data blocks for the Output File Processor (OFP) module (e.g., SDR2, VDR, PLA3), will need to be updated since they are sensitive to the length of CASECC. Otherwise, just change the value of LENCC in /IFP1A/. Some Case Control Cards only change cells in /SYSTEM/ and do not need space in CASECC.

2. Addition (or change) of a Key Word.

To add another key word simply lengthen /IFP1A/ by the number of new words, changing the IFFABD Block Data subprogram. The same procedure will allow you to change the spelling of a current keyword.

3. Identification of Card Type.

In IFP1 there are approximately 70 logical IF statements in a row. Simply add another one modeled after the existing statements with the new key word.

4. Addition of Card Dependent Code.

Add a small internal subroutine to process the new card. The simplest form of such an internal routine is to extract one integer from a card of the form SPC = 1. In this case set "IK" to the word assigned to the card in CASECC and transfer to the common code for this purpose. There are many examples of more complex cards under each card type. Changing these card dependent areas of code allows easy modification of existing card types.

5. Restart Implications

See the subroutine description for IFP1B in the Module Function Description for IFP1 for a description of the restart functions of IFP1.

6.5 THE BULK DATA DECK

The module which processes the Bulk Data Deck is the Input File Processor (IFP), whose Module Functional Description can be found in section 4.5. There are two primary reasons for adding or modifying a bulk data card. First, a new structural element or some other item is to be added to the NASTRAN system. This will require additional code in several other modules besides the Input File Processor (IFP); however, only IFP changes will be discussed here. Second, an alternate form of user input is desired for an already existing item. For example, the SPC1 card is an alternate form of the SPC card. In some cases, an alternate form can be accommodated on the same card. An example of this technique is found on the SPØINT card. The advantage gained in this case is that changes to the NASTRAN system are isolated to the Input File Processor.

There are three major references for the programmer who desires to make modifications or additions to the Input File Processor. Section 2 of the User's Manual gives a functional description of each bulk data card. Section 2.3.2 of the Programmer's Manual describes the format of the output data blocks generated by the Input File Processor, and section 4.5 of the Programmer's Manual contains a description of the processing flow which occurs within IFP. Any programmer responsible for making changes to the Input File Processor would be well advised to select a card which is similar to the one he is changing or implementing and "follow it through" the code, using the three references described above to guide him.

In most cases, the work required to add a new card will amount to adding entries to already existing tables in the IFF Block Data subprograms. The detailed steps for adding a new card to the Input File Processor are listed below.

1. Add the card name and corresponding table data to the IFP data tables contained in Block Data Subprograms IFX1BD, IFX2BD, IFX3BD, IFX4BD, IFX5BD, IFX6BD, and if needed, IFX7BD. The meaning of these entries is discussed in section 4.5.7.
2. Add an entry in the IFP code to call one of the IFP secondary routines IFS1P, IFS2P, IFS3P, or IFS4P, wherein the card dependent processing takes place, and add the necessary card dependent code to the appropriate secondary routine.

MODIFICATIONS AND ADDITIONS TO NASTRAN

3. If the new card is to be used in conjunction with a Rigid Format, appropriate entries must be made in the restart bit tables in /IFX0/ and in the Rigid Format tables. See section 1.10 for details.

RIGID FORMATS

6.6 RIGID FORMATS

The following steps will allow the addition of a new NASTRAN Rigid Format.

1. Compile and test the DMAP sequence thoroughly by running problems on it using APP DMAP or by using the ALTER feature with an existing Rigid Format.
2. All Rigid Formats must be classified FORCE or DISPLACEMENT. A call to a subroutine which stores the new Rigid Format must be added to subroutine XRGDFM of module XCSA (see Section 4.2). The subroutines which contain current Displacement Rigid Formats are called LDi, $i = 01, 02, \dots, 15$. The subroutines which contain current Heat Transfer Rigid Formats are called LD21, LD22, and LD23. The subroutines which contain current Aeroelastic Analysis Rigid Formats are called LD45 and LD46. Dummy calls are already setup for Displacement Rigid Formats LD47-LD50. They correspond to solutions 47 through 50. By choosing to use solutions 47 through 50 and creating the appropriate LDi routine, this step can be skipped.
3. An LDi routine must be written. The LDi routine must write the DMAP sequence, the Decision Tables, the File Name Table, and the Card Name Table on the NPTP. The format is as follows:

Record 1:

This record contains the coded DMAP sequence, 4 characters per word. Note that every DMAP instruction must end with the character: \$. In the existing LDi programs the data for this record are stored in the RD array.

Record 2:

<u>Number of Words</u>	<u>Contents</u>
1	Number of DMAP instructions (NDMAP).
1	Number of words in the decision table for each DMAP instruction (NBIT), $1 \leq \text{NBIT} \leq 5$.
NBIT*DMAP	Decision table for each instruction with the entries for instructions not in this subset set to zero. The words cannot be all zero. The zeroing of the subset entries can be accomplished by a call to XSBSET. (See XCSA MFD write-up). In the existing LDi programs, this table is stored in the IS1 array.
1	Number of entries in the File Name Table (NFILE). This number may be zero if no File Name Table is desired.

MODIFICATIONS AND ADDITIONS TO NASTRAN

<u>Number of Words</u>	<u>Contents</u>
3*NFILE	File Name Table - Each entry consists of 2 BCD words giving the data block name of any data block in the Rigid Format and one integer giving its bit position in the Decision Table. In the existing LDi programs, the File Name Table is stored in the JNM array.
1	Number of entries in the Card Name Table (NCARD). This number may be zero if no Card Name Table is desired.
3*NCARD	Card Name Table - Each entry consists of 2 BCD words giving the card Name of any NASTRAN Data Card and one integer giving its bit position in the Decision Table. In the existing LDi programs, the Card Name Table is stored in the INM array.

To modify an existing rigid format the appropriate tables (Decision, Card Name, File Name or DMAP) should be changed. Their locations in the existing LDi routines are detailed above.

6.7 FUNCTIONAL MODULES

A functional module communicates with the NASTRAN Executive System, and hence indirectly with other functional modules, only through its input data blocks, its output data blocks and its DMAP parameters. Hence a modification to a functional module which disturbs neither its output data blocks nor its DMAP parameters can be made without changing any other functional modules. If a modification to a functional module affects its output data blocks or its DMAP parameters, it must be determined (by referring to section 4, Module Functional Descriptions, of the Programmer's Manual and/or section 3, Rigid Formats, of the User's Manual) what modules use these output data blocks and DMAP parameters as input. If these modules are numerous, or if the changes to them are very extensive, it may be more profitable to write a new module(s) to accomplish the task at hand.

To add a functional module to the system three changes must be made: 1) update the Module Properties List (MPL), the Executive table which contains the (DMAP) name of the module, the number of input data blocks, the number of output data blocks, the number of scratch data blocks and the DMAP parameter list (see the description of the MPL in section 2.4 for more details); 2) update the Link Specification Table (LNKSPC), the Executive table which contains the (DMAP) name of the module, the module's entry point name and the link residence keys for the four machine types (see the description of the LNKSPC table in section 2.4 for more details); and 3) update one or more link driver routines, XSEMi, (see section 3.3.7) so that the module is called from one or more links. Steps 2 and 3 have been automated within the NASTRAN system. These automated procedures are described in sections 6.11.3.1 and 6.11.3.2 respectively. 4) Add the modules to the overlay structure for the appropriate link(s). The means of accomplishing this task is machine dependent, but a basic picture of the module and its subroutines and their relationships to other NASTRAN subroutines should be drawn. An updated Link Map such as those in section 5 should be made. 5) To actually add the module section 5 must be consulted for the particular computer which NASTRAN is operating on.

In addition considerable number of documentation changes must be made. These are described briefly for each manual.

1) Programmer's Manual:

A new subsection for section 4.0 must be written to describe the module. The various table of contents and indexes must be updated including sections 4.1.2 and 4.1.3. The overlay maps in

FUNCTIONAL MODULES

section 5 must be updated.

2) User's Manual:

If the module is to be a "user module" it should also be documented in section 5 of the User's Manual. Any new error messages must be documented in section 6. The module name should be added to the dictionary in section 7.

3) Theoretical Manual:

No additions are required here unless the module has some significant analytic developments which need to be documented.

6.8 ADDING A STRUCTURAL ELEMENT

6.8.1 Introduction to the Problem

This section defines the programming interfaces necessary to add a new structural element to NASTRAN. We assume the reader of this section has:

1. A good working knowledge of FØRTRAN IV
2. Experience in programming on a large-scale scientific computer, including use of overlays. (Preferably this experience has been on one of the computers on which NASTRAN is operational: IBM 360, ØS; UNIVAC 1108, Exec 8; or CDC 6600, SCOPE 3.0)
3. A working knowledge of matrix algebra, viz., addition, multiplication and inversion of matrices

We also assume the reader has had prior experience with neither structural analysis nor the NASTRAN program. Furthermore, we assume that a structural analyst has written mathematical specifications for a structural element, and that the reader must design, code, and checkout NASTRAN FØRTRAN subroutines that conform to these mathematical specifications.

6.8.1.1 Introduction to Structural Analysis

A scientific programmer must gain a minimal understanding of the physics and mathematical techniques involved in the application at hand. The following paragraphs, condensed from section 3 of the Theoretical Manual, give this minimal analytical background.

From a theoretical viewpoint, the matrix equation

$$[K]\{u\} = \{P\} \quad , \quad (1)$$

completely describes the formulation of a static (the most basic) structural problem. $[K]$ is called a stiffness matrix, $\{P\}$ is called a load vector and $\{u\}$, the unknown of the equation, is called a displacement vector. NASTRAN generates $[K]$ and $\{P\}$ from available information about the structure. Once Equation 1 has been formed, it is solved for each specified loading condition. Stresses in the structural elements and other desired results are then obtained from $\{u\}$ by a set of data recovery operations.

NASTRAN embodies a lumped element approach, i.e., the distributed physical properties of a structure are represented by a model consisting of a finite number of idealized substructures or elements that are interconnected at a finite number of points. All input and output data pertain to the idealized structural model.

The idealized structural model in NASTRAN consists of "grid points" (G) to which "loads" (P) are applied, and at which degrees-of-freedom are defined, and "elements" (E) that are connected between the points, as shown in Figure 1. Two general types of grid points are employed. They are:

1. Geometric grid point - a point in three-dimensional space at which an arbitrary number of components of displacement and rotation are defined. The location and orientation coordinates of each grid point are specified by the user. Components of displacement and rotation can be eliminated as degrees-of-freedom by means of "single-point constraints".
2. Scalar point - a point in vector space at which one degree-of-freedom is defined. A geometric grid point contains from one to six scalar points. Scalar points may exist that are not associated with grid points. Such points can be coupled to geometric grid points by means of scalar structural elements and by constraint relationships.

The structural element is a convenient localizing concept for specifying many of the properties of the structure, including material properties, mass distribution and some types of applied loads. Structural elements are defined on "connection" cards by referencing the grid points to which they are interconnected. In a few cases, all of the information required to generate structural matrices for the element is given on the connection card. In most cases, the connection card refers to a "property" card, on which the cross-sectional properties of the element are given. Adding a new structural element to NASTRAN necessitates designing and implementing a new connection card and, if defined, a new property card.

There are four general classes of structural elements:

1. Metric elements connected between geometric grid points. Examples include rod, plate, and shell elements.
2. Scalar (or zero-dimensional) elements connected between pairs of scalar points, or between one scalar point and "ground". Since each geometric grid point contains a number

ADDING A STRUCTURAL ELEMENT

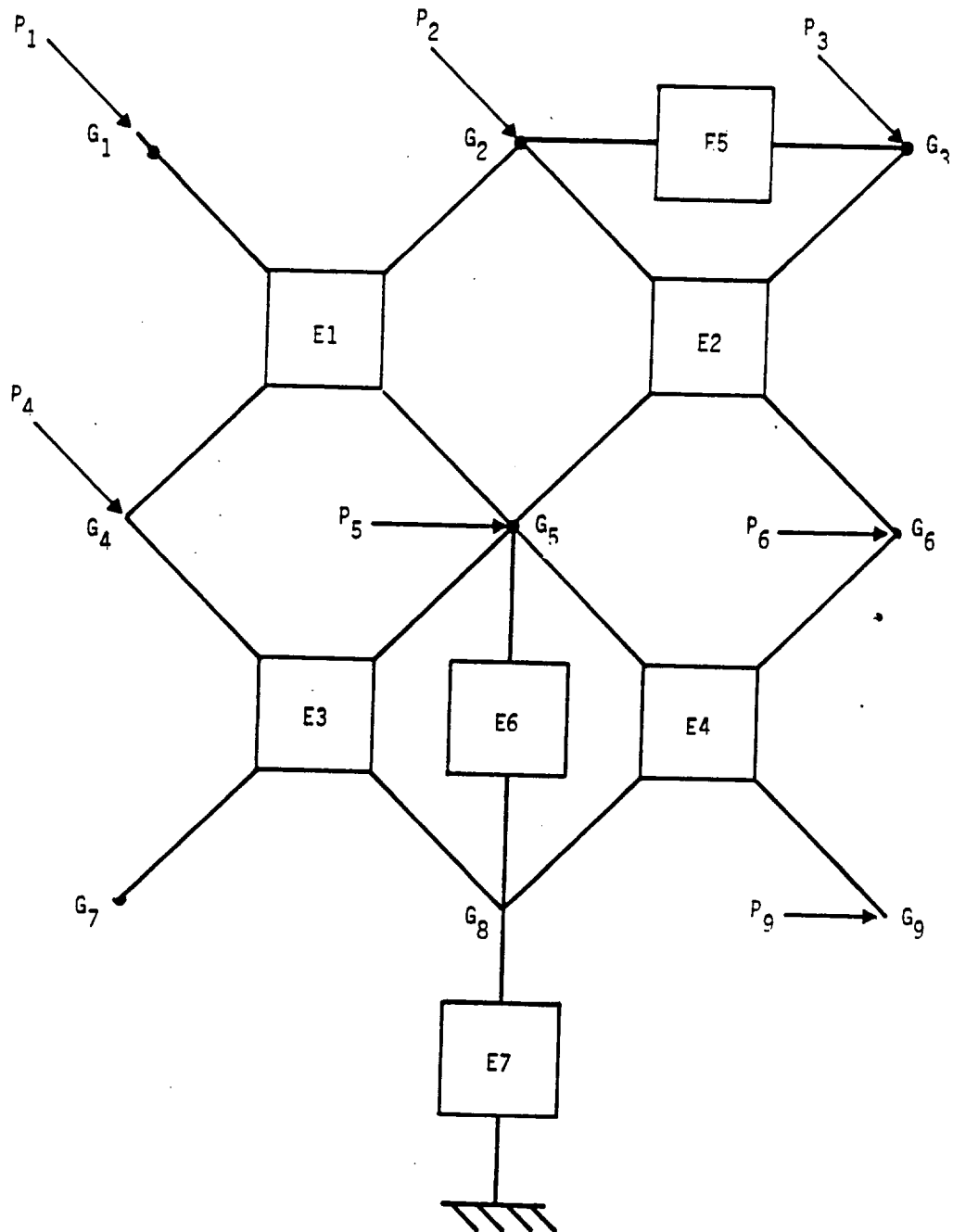


Figure 1. Topology of the idealized structural model.

of scalar points corresponding to specific components of motion, scalar elements can be connected between selected components of motion at geometric grid points.

3. General elements, whose properties are defined in terms of deflection influence coefficients (i.e., compliance matrices), and which may be interconnected between any number of geometric and scalar grid points. An important application of general elements is the representation of large components of structure by test data.

4. Constraint elements (or Constraints). The existence of a constraint element implies a linear relationship among the degrees-of-freedom to which it is attached. This relationship is of the form

$$\sum_g R_{cg} u_g = Y_c, \quad (2)$$

where u_g are degrees-of-freedom and Y_c is an enforced displacement. A linear relationship among the forces of constraint is also implied, since it is required that the forces of constraint do no work.

The remainder of this section will concentrate on class 1, metric elements, and, from this point on, the phrase "structural element" or simply "element" shall mean metric structural element as defined above.

6.8.1.2 General Problem Flow

NASTRAN consists of a number of subprograms, or modules, that are executed according to a sequence of macro-instructions. The NASTRAN Executive System (NES) controls the flow of this sequence. Twenty such sequences, called Rigid Formats, are permanently stored in NES and can be selected by means of control cards. Each rigid format corresponds to a particular type of structural analysis. Detailed explanations of each rigid format can be found in Section 3 of the User's Manual.

Since a stiffness matrix $[K]$ must be generated for all rigid formats, structural elements interact with all twenty rigid formats. However, if the reader has a minimal understanding of rigid format 1, Basic Static Analysis; rigid format 4, Static Analysis with Differential Stiffness; and rigid format 6, Piecewise Linear Analysis; then he will have enough background to add a new

element. Rigid formats 4 and 6 are minor variations of rigid format 1. Therefore, if all the module changes for rigid format 1 have been completed, only a few additional modules must be changed to add the new element to the differential stiffness and piecewise linear analysis element libraries. With these points in mind, the following subsection discusses the Basic Static Analysis rigid format in some detail.

6.8.1.2.1 Basic Static Analysis

Figure 2 shows a simplified flow diagram for Basic Static Analysis. Each block in the flow diagram represents a number of NASTRAN modules. Not every module has to be changed to add an element. Those that do have to be changed are called element-dependent; those that do not have to be changed are called element-independent.

In block 1 of Figure 2, the Input Data Processor, as the name implies, reads and analyzes the information on input data cards and reorganizes it into data blocks consisting of lists of similar quantities. NASTRAN input data cards reside in three separate decks: the Executive Control Deck, the Case Control Deck and the Bulk Data Deck. Section 2 of the User's Manual describes the contents of each of these decks. The reader need only be concerned with the Bulk Data Deck, for it is in this deck that the new element's connection and property information will be found. The Input File Processor (IFP) module analyzes each card of the Bulk Data Deck for correctness of format and distributes the data in the Bulk Data Deck to various data blocks. The primary function of the Executive Control Section Analysis (XCSA) module is to read and analyze the cards in the Executive Control Deck. XCSA also contains tables necessary for problem restarts, and it is in these tables that updates must be made. Detailed explanations of IFP and XCSA changes will be found in Sections 6.8.3.1 and 6.8.3.2, respectively.

In block 2 of Figure 2, the Geometry Processor generates coordinate system transformation matrices, tables of grid point locations, a table defining the structural elements connected to each grid point, and other miscellaneous tables such as those defining static loads and temperatures at grid points. The Geometry Processor consists of: Geometry Processor - Phase 1 (GP1); Geometry Processor - Phase 2 (GP2); Geometry Processor - Phase 3 (GP3); and the Table Assembler (TA1). Section 6.8.3.3 explains Geometry Processor interfaces which are minimal.

MODIFICATIONS AND ADDITIONS TO NASTRAN

The Structures Plotter generates tape output for an automatic plotter that will plot the structure (i.e., the location of grid points and the boundaries of elements) in one of several available three-dimensional projections. The Structures Plotter is particularly useful for the detection of errors in grid point coordinates and in the connection of elements to grid points. The Structures Plotter may also be used at the end of the program to superimpose images of the deformed and undeformed structure. The Structures Plotter consists of the Plot Set Definition Processor (PLTSET) module and the Structural Plotter (PLØT) module.

The Structural Matrix Assembler generates stiffness, mass, and damping matrices referred to the grid points from tabular information generated by the Input File Processor and the Geometry Processor. NASTRAN uses the mass matrix in static analysis for the generation of gravity loads and inertia loads on unsupported structures. The assembly of the structural matrices is performed in the Element Matrix Generator (EMG) which calculates the separate stiffness, mass, and damping matrix partitions for each element, the Element Matrix Assembler (EMA) which assembles the partitions, and the Structural Matrix Assembler - Phase 3 (SMA3), which generates stiffness matrix contributions from general elements. Since we are not concerned with general elements (see Section 6.8.1.1), programming interfaces in SMA3 are not discussed. Section 6.8.3.5 discusses the EMG/EMA interfaces.

In block 5 of Figure 2, the stiffness matrix is reduced to the form in which its matrix equation is finally solved through the imposition of single-point and multipoint constraints, and the optional use of matrix partitioning. No element-dependent code exists in block 5.

In block 6 of Figure 2, the Static Solution Generator - Phase 1 (SSG1) module generates load vectors $\{P_i\}$ from a variety of sources: concentrated loads at grid points; pressure loads on surfaces; gravity loads; temperature loads; and enforced deformations. The only types of loads that will concern the reader are thermal and enforced deformation loads, both of which are calculated using the stiffness properties of the structural elements. However, thermal and enforced deformation loads do not exist for all elements. The SSG1 interfaces are discussed in Section 6.8.3.7. Module SSG2, which is element-independent, reduces the load vectors $\{P_i\}$ to final form by the application of constraints and matrix partitioning.

ADDING A STRUCTURAL ELEMENT

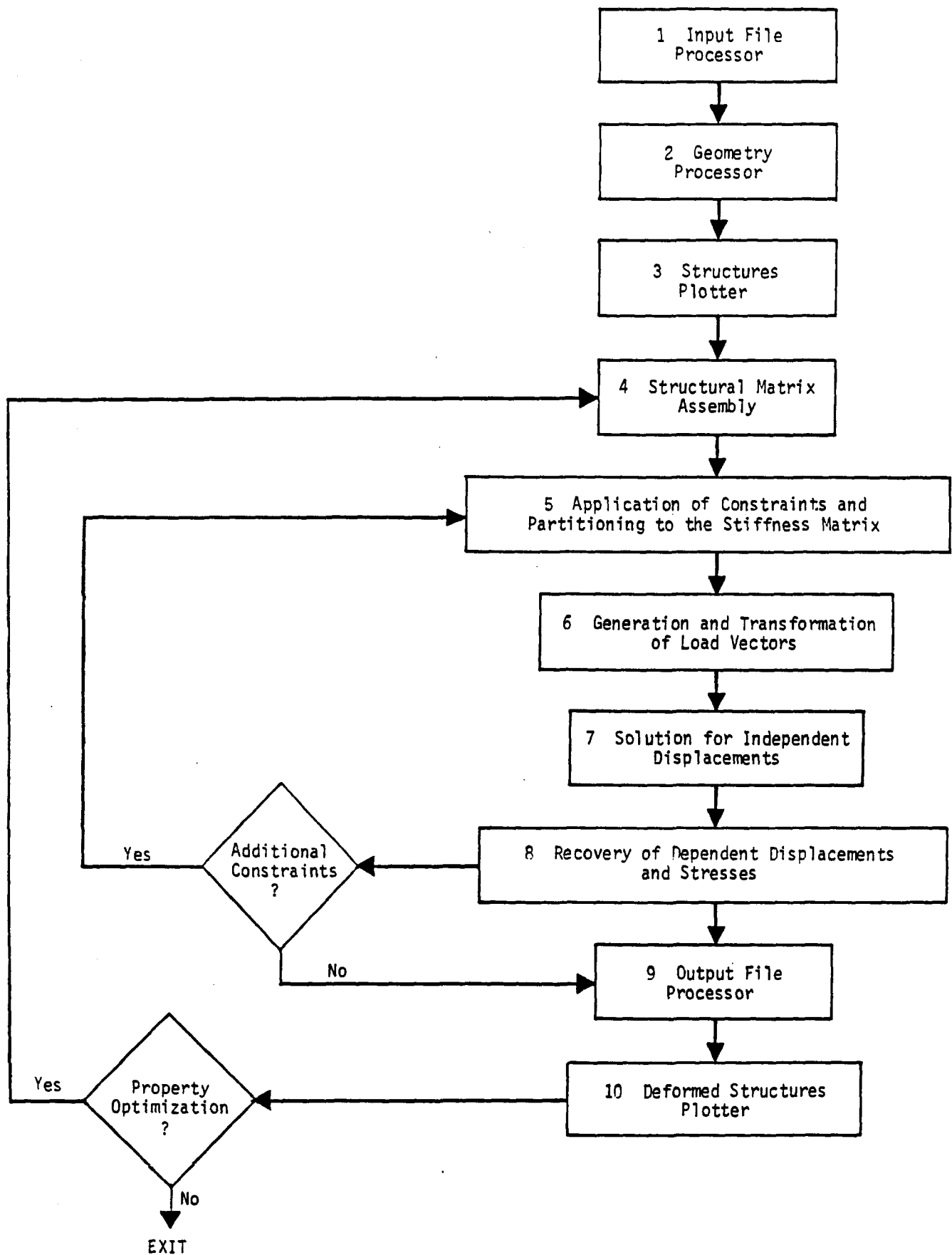


Figure 2. Simplified flow diagram for basic static analysis.

MODIFICATIONS AND ADDITIONS TO NASTRAN

In block 7 of Figure 2, the solution for the independent displacements $\{u_i\}$ is accomplished in two steps: decomposition of the stiffness matrix $[K]$ into upper and lower triangular factors; and solution for the $\{u_i\}$ corresponding to the specific load vectors, $\{P_i\}$, by means of successive substitutions into the equations represented by the triangular factors of $[K]$ (the so-called forward and backward passes). Modules RBMG2 and SSG3, both of which are element-independent, accomplish this solution.

In block 8 of Figure 2, module SDR1, which is element-independent, determines dependent displacements. The internal forces and stresses in each element are then computed in the Stress Data Recovery - Phase 2 (SDR2) module from knowledge of the displacement components at the grid points of the elements and the intrinsic structural equations of the element. SDR2 programming interfaces are discussed in Section 6.8.3.8.

Finally in block 9 of Figure 2, the Output File Processor (ØFP) module formats the element forces and stresses that were computed in SDR2 for printing on the system output file. ØFP interfaces are discussed in Section 6.8.3.9.

6.8.1.2.2 Static Analysis With Differential Stiffness

Figure 3 shows a simplified flow diagram for rigid format 4, Static Analysis with Differential Stiffness. A comparison between Figures 2 and 3 shows that the first eight blocks of Figure 2 and 3 are identical.

Contributions to the differential stiffness matrix are not defined for all elements currently in NASTRAN, and they may not be defined for a new element. The differential stiffness matrix, which is a first order approximation to large deformation effects, is a function of the most recently iterated displacement. Functional Module, DSCHK, (see Section 4.121) performs differential stiffness calculations based on user-supplied iteration parameters. The solution strategy basically involves a load adjustment (the "inner" loop) in order to satisfy iterated displacements within a specified convergence criteria.

ADDING A STRUCTURAL ELEMENT

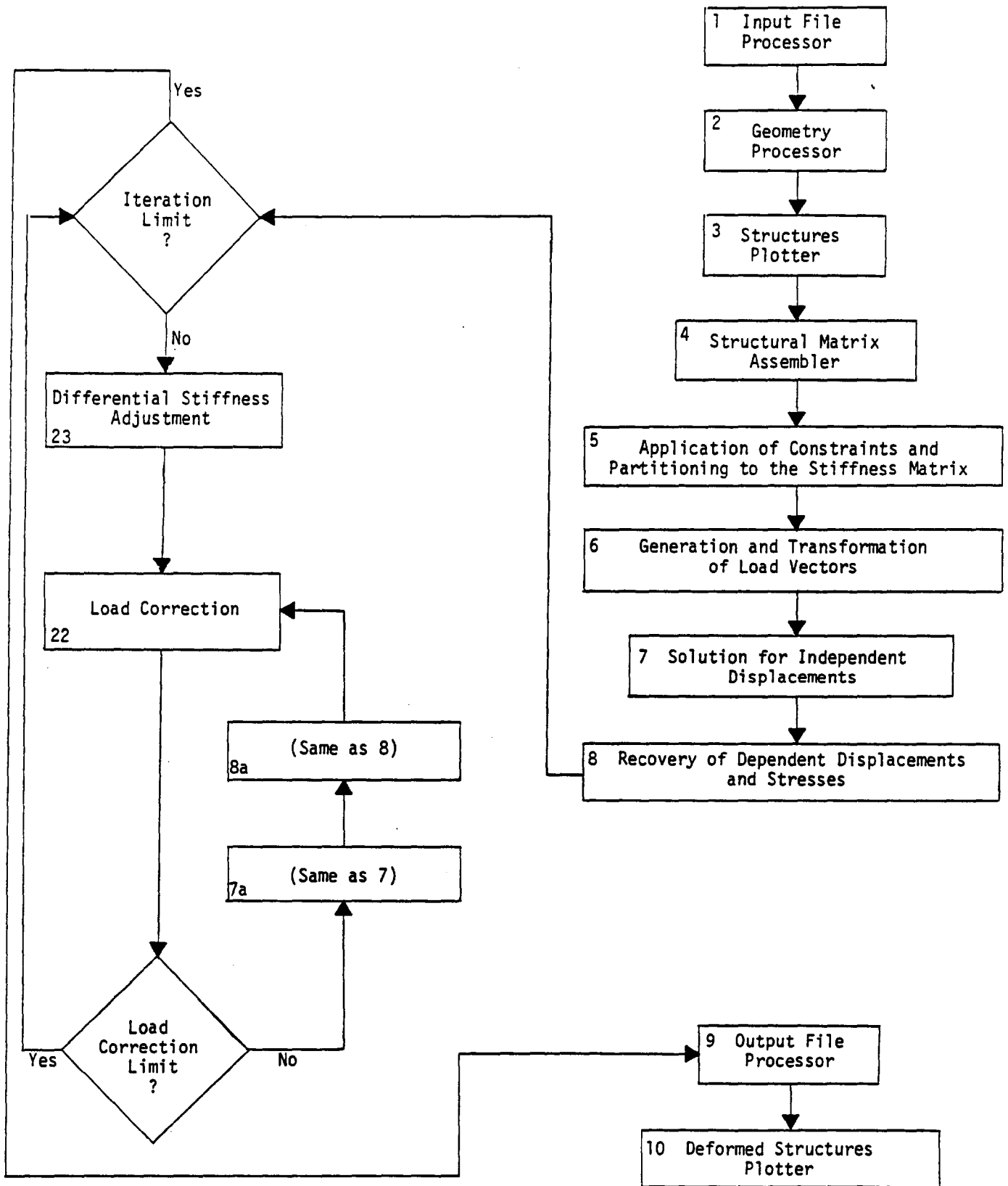


Figure 3. Simplified flow diagram for Static Analysis with Differential Stiffness.

6.8.1.2.3 Piecewise Linear Analysis

Figure 4 shows a simplified flow diagram for rigid format 6, Piecewise Linear Analysis. In piecewise linear analysis, solutions are obtained for structures with nonlinear, stress-dependent, material properties. The load level is increased to its full value by small increments, such that stiffness properties can be assumed to be constant over each increment. After each increment the combined strains in nonlinear elements due to all load increments are used, in conjunction with stress-strain tabular functions, to determine the appropriate stiffnesses for the next load increment. Piecewise linear analysis is not defined for all elements currently in NASTRAN and it may not be defined for a new element.

Blocks 1 through 4 of Figure 4 are identical to blocks 1 through 4 of Figures 2 and 3. Blocks 4A and 4B are performed by the Piecewise Linear Analysis - Phase 1 (PLA1) module. PLA1 classifies all elements as linear or nonlinear. An element is said to be linear if its modulus of elasticity E, defined on a MAT1 bulk data card, is not defined to be stress-dependent on a TABLES1 bulk data card. PLA1 generates the linear stiffness matrix using the element routines of the SMA1 module, and the nonlinear elements comprise the ESTNL and ECPTNL data blocks. The ESTNL and ECPTNL data blocks, used subsequently in modules PLA3 and PLA4 respectively, have the same general formats as the EST and ECPT data blocks from which they are derived. PLA1 reads the EST and ECPT, and, for each element entry, appends stress information about the element. Modules PLA3 and PLA4 update the appended stress information each time these modules are executed in the loop that extends from block 5 to block 21 in Figure 4. Section 6.8.3.11 discusses PLA1 interfaces.

Block 5 in Figure 4 is identical to block 5 in Figures 2 and 3 and is element-independent. Block 6 contains module SSG1, which is no different in this rigid format from the two previously discussed. Block 17 is element-independent, and block 7, identical to block 7 in Figures 2 and 3, is also element-independent.

Block 8A denotes module SDR1 which is, as indicated above, element-independent. Note that the SDR2 module, block 8B, is outside the loop.

ADDING A STRUCTURAL ELEMENT

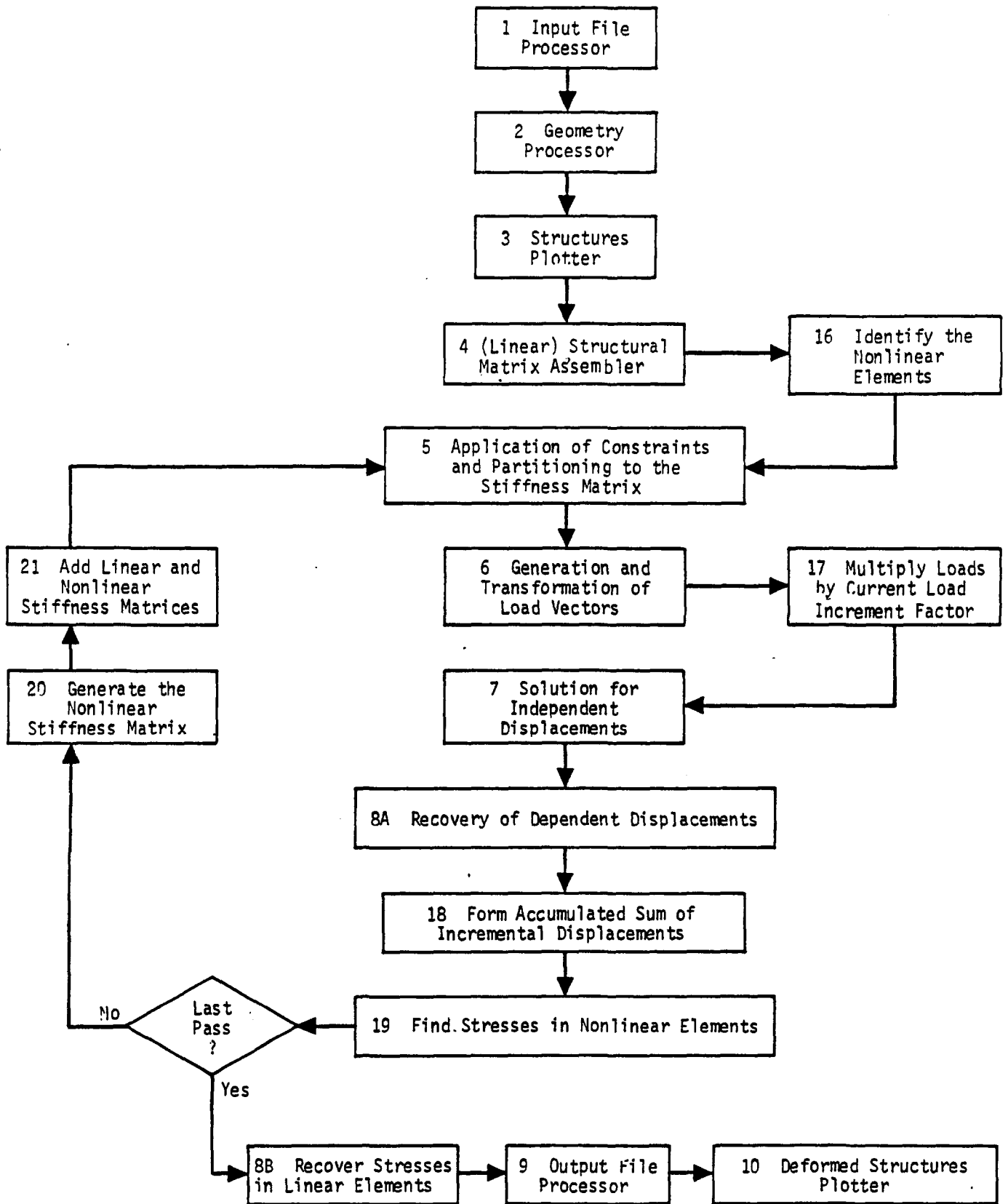


Figure 4. Simplified flow diagram for piecewise linear analysis.

2000

MODIFICATIONS AND ADDITIONS TO NASTRAN

Block 18 denotes module PLA2 which is element-independent, and in block 19 the stresses in the nonlinear elements are computed in the Piecewise Linear Analysis - Phase 3 (PLA3) module. This module is similar to the element stress data recovery portions of the SDR2 module. PLA3 interfaces are discussed in Section 6.8.3.12.

In block 20 of Figure 4, the Piecewise Linear Analysis - Phase 4 (PLA4) module generates the stiffness matrix associated with the nonlinear elements. PLA4 is very similar in structure to SMA1, and PLA4 interfaces are discussed in Section 6.8.3.13.

Block 21 completes the Piecewise Linear Analysis rigid format loop. The module here adds the linear stiffness matrix, which is constant throughout the loop, to the nonlinear stiffness matrix, which varies each time through the loop.

6.8.1.3 Summary

Summarizing Section 6.8.1, we have seen that:

1. NASTRAN embodies a lumped element approach wherein the distributed physical properties of a structure are represented by a model consisting of a finite number of idealized sub-structures or elements that are interconnected at a finite number of points.
2. The reader, who is an experienced FORTRAN scientific programmer, who knows the basics of matrix algebra but knows little or nothing about structural analysis, must design, code and checkout subroutines that will add the capability of a new element to the NASTRAN element library.
3. A major component of NASTRAN is an Executive System which controls the sequence of module executions according to options specified by the user.
4. Certain of the modules in NASTRAN are element-dependent and hence must be changed.
5. Some element-dependent modules have to be changed substantially; other element-dependent modules have to be changed only either to update tables, whose definitions are isolated to Block Data programs, or to skip element-dependent data which will not be used in certain modules (e.g., if differential stiffness and piecewise linear analysis are not defined for the element, no substantial changes need be made to DSMG1 and the piecewise linear analysis modules). The modules that have to be changed substantially are: IFP, SMA1, SMA2, SDR2 and

ADDING A STRUCTURAL ELEMENT

ØFP. Those modules which are in the second category are: XCSA, the Geometry Processor modules, PLTSET, SSG1, DSMG1, PLA1, PLA3 and PLA4.

6.8.1.3.1 Modules Which Must be Changed

The following summary lists the names of the modules that must be changed, the section of the Programmer's Manual that describes the module, the purpose of the module, and the reason for the change.

1. Name and Reference: Input File Processor (IFP); Section 4.5

Purpose: To read and analyze the information on input data cards that define the mathematical model of the structure; and then to distribute these data items to data blocks consisting of lists of similar quantities.

Reason for Change: When adding a new element, the user must define a connection card and, in most cases, a property card. He defines structural elements on connection cards by referencing the grid points that define the boundary of the element, (e.g., see the CRØD card in section 2.4 of the User's Manual). In most cases, the connection card refers to a property card, on which the cross-sectional properties of the element are given (see the PRØD card in section 2.4 of the User's Manual). In a few cases, the connection card gives all the information required to generate the structural matrices for the element.

2. Name and Reference: Executive Control Section Analysis (XCSA); Section 4.2

Purpose: To read and analyze the Executive Control Deck; also it contains tables for problem restarts.

Reason for Change: The names of the new connection and property cards must be added to the Card Name Tables, which are a subset of the restart tables.

3. Name and Reference: Geometry Processor, consisting of modules GP1, GP2, GP3, and TA1; sections 4.21, 4.22, 4.25, and 4.26, respectively.

MODIFICATIONS AND ADDITIONS TO NASTRAN

Purpose: For GP1, to generate the coordinate system transformation matrices; for GP2, to convert external grid point numbers on connection cards to internal numbers; for GP3, to process static loads and temperature data; for TA1, to process and collect element connection data, element property data, element geometry data, and, if applicable, element temperature data into two different data blocks for later processing. One data block, the Element Connection and Properties Table (ECPT), is used in matrix assembler modules SMA1, SMA2 and DSMG1; the other, the Element Summary Table (EST), is used in load generation and element data recovery modules SSG1 and SDR2.

Reason for Change: Common block GPTA1 must be updated. Descriptive information in this common block completely describes element interfaces in these four modules.

4. Name and Reference: Element Matrix Generator (EMG); Section 4.124

Purpose: To generate the element stiffness, mass, and damping partitions exclusive of general elements.

Reason for Change: A subroutine, called an "element routine" which generates the element stiffness matrix for the new element must be coded.

5. Name and Reference: Static Solution Generator - Phase 1 (SSG1); Section 4.41

Purpose: To compute the static loads selected by the user.

Reason for Change: Element-dependent code which generates load vector contributions due to thermal or enforced deformation loads must be added to SSG1.

6. Name and Reference: Stress Data Recovery - Phase 2 (SDR2); Section 4.46

Purpose: To recover internal forces and stresses in each element using the EST data block.

ADDING A STRUCTURAL ELEMENT

Reason for Change: Two element routines which recover element stresses and forces for the new element must be coded.

7. Name and Reference: Output File Processor (ØFP); Section 4.70

Purpose: To format and print data prepared for output by other functional modules.

Reason for Change: To incorporate formats for the element stresses and forces computed in SDR2.

8. Name and Reference: Differential Stiffness Matrix Generator - Phase 1 (DSMG1); Section 4.49

Purpose: To generate the differential stiffness matrix.

Reason for Change: If the added element is to have contributions to the differential stiffness matrix, a new element routine must be coded.

9. Name and Reference: Piecewise Linear Analysis - Phase 1 (PLA1); Section 4.52

Purpose: To partition all elements into two classes, linear and nonlinear; and to build the data blocks ESTNL and ECPTNL, which are similar in form to the EST and ECPT, and which are used in modules PLA3 and PLA4 respectively.

10. Name and Reference: Static Solution Generator - Heat (SSGHT); Section 4.105

Purpose: To generate nonlinear corrective heat loads for elements involved in heat transfer analysis.

MODIFICATIONS AND ADDITIONS TO NASTRAN

Reason for Change: If the new element is to be admissible to the class of elements for which piecewise linear analysis is defined, data that will be appended to the element's ECPT and/or EST entry to form its ECPTNL and/or ESTNL entry must be initialized.

11. Name and Reference: Piecewise Linear Analysis - Phase 3 (PLA3); Section 4.54

Purpose: To compute element stresses for nonlinear elements; to update the ESTNL data block with accumulated element stress information.

Reason for Change: To code a new element routine which will compute stresses and update accumulated stress information for the element.

12. Name and Reference: Piecewise Linear Analysis - Phase 4 (PLA4); Section 4.55

Purpose: To generate the stiffness matrix for nonlinear elements; to update ECPTNL data block with accumulated element stress information.

Reason for Change: To code a new element routine which will compute stiffness matrix contributions and update accumulated stress information for the element.

6.8.2 General Guidelines

Before proceeding with the details (given in section 6.8.3) of coding in each of the modules listed in section 6.8.1.3.1, this section gives general guidelines, some of which will be applicable to all the modules to be changed, while some will be applicable to only a certain class of modules.

6.8.2.1 FØRTRAN Rules

As indicated in section 6.2 NASTRAN is written almost entirely in FØRTRAN IV. Since the program operates on three machines (IBM 360, UNIVAC 1108 and CDC 6600), the NASTRAN design team chose a subset of FØRTRAN IV to be the "language" for NASTRAN coding. To plan for the possibility that an element added locally will be incorporated into the global NASTRAN system

ADDING A STRUCTURAL ELEMENT

Table 1 gives the classification of the modules listed in section 6.8.1.3.1.

Table 1. Classification of Modules to be Changed

A. Data Processing Modules	
<u>Name</u>	<u>Function</u>
IFP	Processes the Bulk Data Deck
XCSA	Describes the Card Name Table for problem restarts
GPTABD ⁽¹⁾	Describes connection/property characteristics of each element used by modules GP1, GP2, GP3, TA1, etc.
PLA1	Preprocessor for the Piecewise Linear Analysis rigid format
ØFP	Formats and prints answers
B. Structural Modules	
<u>Name</u>	<u>Function</u>
EMG	Generates the element, mass, and damping stiffness matrices
SSG1	Generates load vectors
SDR2	Computes element stresses and forces
DSMG1	Generates the differential stiffness matrix
PLA3	Computes element stresses for nonlinear elements
PLA4	Generates the stiffness matrix for nonlinear elements
SSGHT	Generates heat loads for nonlinear heat transfer
⁽¹⁾ GPTABD is a Block Data subprogram	

MODIFICATIONS AND ADDITIONS TO NASTRAN

(i.e., become operational on all NASTRAN computers) without unnecessary conversion problems, it is suggested that the programmer follow the NASTRAN FORTRAN rules given in Section 6.2 for all module changes.

6.8.2.2 Classification of Modules

The modules in NASTRAN can be classified in many different ways. For the purposes of this section we classify the modules that must be changed to add a new element into two categories: data processing modules and structural modules. The data processing modules are those whose output data blocks are used either as input data blocks to other data processing modules or as input data blocks to the structural modules. ØFP is also classified as a data processing module. The structural modules are those whose output data blocks are the matrices and vectors needed for the solution of the structural problem.

6.8.2.3 Data Processing Modules

We discuss the data processing modules with respect to the data blocks output from them. A data block is a set of data, a matrix or a table, occupying a file, which can be thought of as a logical FORTRAN unit. Section 2 gives detailed descriptions of the formats of the data blocks that are used in the twelve NASTRAN rigid formats. The formats are independent of rigid format. Table 2 gives the data blocks needed for element generation along with their use as input to the structural modules.

6.8.2.4 Structural Modules

The structural modules perform the actual floating-point arithmetic operations to generate matrices and load vectors and to recover element stress and force data. The structural modules contain element routines that: a) receive their inputs from the module driver; b) perform matrix operations to generate element-dependent matrices or vectors; and c) transfer their outputs to the driver or a module utility routine so they can be incorporated into a data block.

The structural modules are further classified into two classes: the matrix generation modules, consisting of EMG, DSMG1, and PLA4; and the load vector generation and data recovery modules, consisting of SSG1, SDR2, PLA3, and SSGHT.

ADDING A STRUCTURAL ELEMENT

Table 2. Data Blocks Needed for Element Generation

Data Block Name	Output From Module	Input to Modules
MPT	IFP	EMG, SSG1, SDR2, DSMG1, PLA1, PLA3, PLA4, SSGHT
DIT	IFP	EMG, SSG1, SDR2, DSMG1, PLA1, PLA3, PLA4, SSGHT
EDT	IFP	SSG1, SDR2, DSMG1
CSTM	GP1	EMG, SSG1, SDR2, DSMG1, PLA1, PLA3, PLA4
SIL	GP1	SSG1, SDR2, DSMG1, SSGHT
GPTT	GP3	SSG1, SDR2, DSMG1, SSGHT
ECPT	TA1	DSMG1, PLA1
GPCT	TA1	EMG, DSMG1, PLA1, PLA4
EST	TA1	EMG, SSG1, SDR2, PLA1, SSGHT
ESTNL	PLA1	PLA3
ECPTNL	PLA1	PLA4

MODIFICATIONS AND ADDITIONS TO NASTRAN

The matrix generation modules have the following common characteristics:

1. They use double precision arithmetic. (Note: Single precision is allowed in EMG.)
2. They use the following data blocks: MPT, DIT, and CSTM.
3. They use the EST or ECPT data block (or a variation, the ECPTNL, in the case of PLA4).
4. They use the utility routines GMMATD, INVERD, PREMAT and PRETRD.
5. They generate the matrices using the partitions associated with each element.

The load vector generation and data recovery modules have the following common characteristics:

1. They use single precision arithmetic.
2. They use the following data blocks: MPT, DIT, EDT (except PLA3), CSTM, SIL, GPTT (except PLA3).
3. They use the EST data block (or a variation, the ESTNL, in the case of PLA3) rather than the ECPT data block.
4. They use the utility routines GMMATS, INVERS, PREMAT and PRETRS.

6.8.2.4.1 The EST versus the ECPT

The Table Assembler module (TAl), the last of the data processing modules to be executed, processes element connection data, element property data, element geometry data and, if applicable, an element temperature datum. TAl merges these data into two different sorts for efficiency in subsequent processing. The Element Summary Table (EST) contains one logical record for each element type. For each element (record) in the EST, connection, property, geometry, and temperature data are grouped. The Element Connection and Properties Table is essentially the EST in a different sort. The ECPT contains one logical record for each grid or scalar point of the model. Each logical record contains EST data for each element connection to the grid or scalar point associated with the record. When the EMG/EMA sequence is used, the Grid Point-Element Connection Table (GPECT) is generated instead of the ECPT data block. The GPECT is identical to the ECPT data except that no property or geometry data is included.

ADDING A STRUCTURAL ELEMENT

The element matrix generator, load vector generation, and data recovery modules use the EST. The other matrix generation modules use the ECPT. Section 6.8.2.4.2.1 more fully describes the use of the ECPT. Table 3 shows the EST/ECPT data for a rod element. The last two columns give respectively the data block that contained the data and the bulk data card type where the user originally placed the data.

6.8.2.4.2 Matrix Generation Modules

The matrix $[K]$ in Equation 1 of Section 6.8.1.1 is a global or system stiffness matrix. It is called global because it contains contributions from all structural elements of the mathematical model. On the other hand, associated with each element is an element stiffness matrix. The paragraphs below explain the relationship between element stiffness matrices and the global stiffness matrix. Although the remarks are directed towards stiffness matrices, they apply equally as well to mass and differential stiffness matrices.

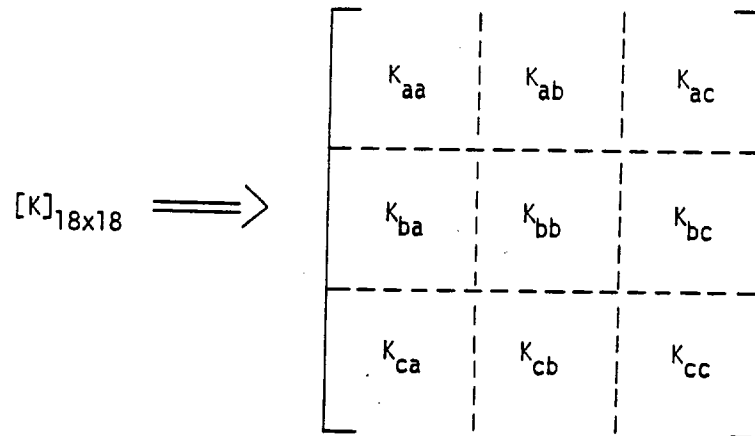
The stiffness matrix $[K]$ for a structural element consists of a six-by-six matrix partition for each combination of the connected grid points. Each six-by-six partition relates the six degrees-of-freedom of two connecting grid points. For example, a ROD element connects two grid points "a" and "b". The element stiffness matrix partitions are $[K_{aa}]$, $[K_{ab}]$, $[K_{ba}]$ and $[K_{bb}]$. A triangular element (e.g., TRMEM) connects three grid points, and the element stiffness matrix consists of nine six-by-six partitions: $[K_{aa}]$, $[K_{ab}]$, $[K_{ac}]$, $[K_{ba}]$, $[K_{bb}]$, $[K_{bc}]$, $[K_{ca}]$, $[K_{cb}]$ and $[K_{cc}]$. Figure 5 shows the position of these partitions in the overall element stiffness matrix for triangular membrane and rod. Figure 6 shows the way in which six-by-six element stiffness matrix partitions are related to a global stiffness matrix. It shows a structure consisting of four grid points and two elements, a triangular membrane and a rod. The six-by-six partition $[K_{33}]$ for the triangular membrane element is added to the six-by-six partition $[K_{33}]$ for the rod element. The partitions $[K_{14}]$, $[K_{41}]$, $[K_{24}]$ and $[K_{42}]$ are zero since no element connects either points 1 and 4 or points 2 and 4. Two methods are used to transmit the element matrixes: (1) in EMG, the partitions are assembled into an N-by-N matrix, as shown in Figure 5; (2) in DSMG1 and PLA4, the individual six-by-six partitions are inserted separately. In either case, an insertion routine performs the task of assembling the partition into the structure matrices.

MODIFICATIONS AND ADDITIONS TO NASTRAN

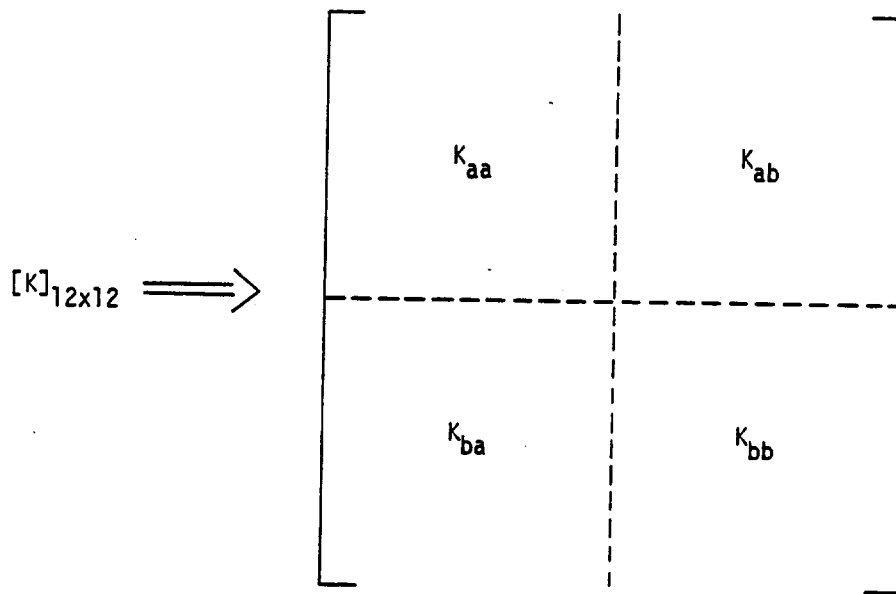
Table 3. EST/ECPT Data for a Rod Element

Word	Item	Type	Data Block	Card Type
1	Element ID	Integer	ECT	CR0D
2	Scalar Index for Grid Point A	Integer	ECT	CR0D
3	Scalar Index for Grid Point B	Integer	ECT	CR0D
4	Material ID	Integer	EPT	PR0D
5	Area (A)	Real	EPT	PR0D
6	Polar Moment of Inertia (J)	Real	EPT	PR0D
7	Torsional Stress Coefficient (C)	Real	EPT	PR0D
8	Nonstructural Mass (MU)	Real	EPT	PR0D
9	Coordinate System ID for Grid Point A	Integer	BGPDT	GRID
10	X-Coordinate of Grid Point A	Real	BGPDT	GRID
11	Y-Coordinate of Grid Point A	Real	BGPDT	GRID
12	Z-Coordinate of Grid Point A	Real	BGPDT	GRID
13	Coordinate System ID for Grid Point B	Integer	BGPDT	GRID
14	X-Coordinate of Grid Point B	Real	BGPDT	GRID
15	Y-Coordinate of Grid Point B	Real	BGPDT	GRID
16	Z-Coordinate of Grid Point B	Real	BGPDT	GRID
17	Element Temperature	Real	GPTT	TEMP

ADDING A STRUCTURAL ELEMENT



Triangular membrane element (3 grid points)



Rod element (2 grid points)

Figure 5. Element stiffness matrix partitions for a triangular membrane and a rod.

MODIFICATIONS AND ADDITIONS TO NASTRAN

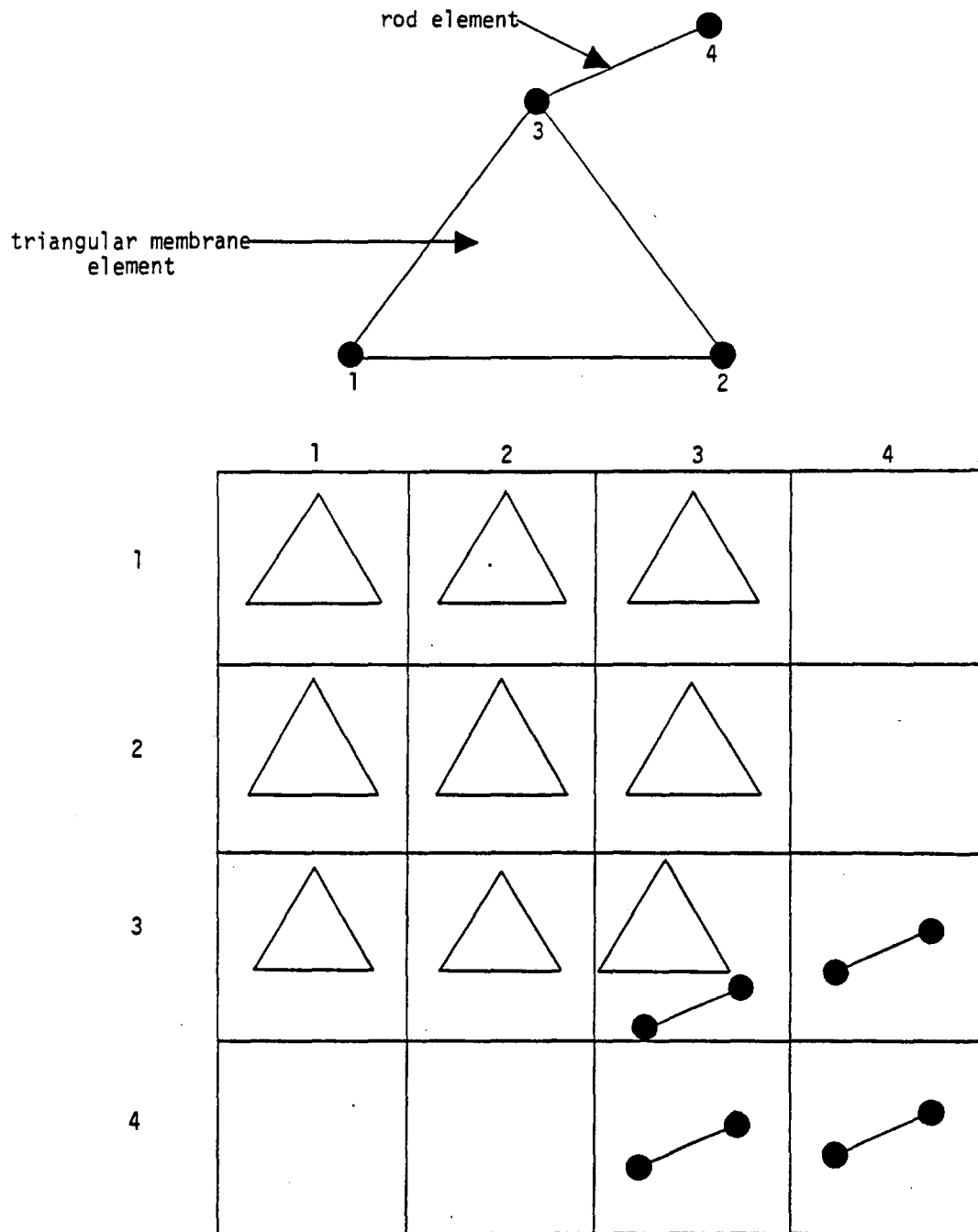


Figure 6. A simple structure and the associated global stiffness matrix.

6.8.2.4.2.1 Generation of Matrices

Since a structural element will affect terms in the global matrices only in rows and columns related to its interconnected grid points, each column i , say, (or row i since the global matrices are symmetric) may be formed using only elements connected to the grid point associated with column i . The DSMG1, PLA4, and the obsolete SMA routines form the global matrices six columns at a time. The data block that enables this to be done is the Element Connection and Properties Table (ECPT), output from the Table Assembler (TA1) module. Each record of the ECPT corresponds to a grid point (or scalar point) of the model, and, conversely every grid point (and scalar point) of the model corresponds to a record of the ECPT. The point to which a record of the ECPT corresponds is called the pivot point of the record. Each record contains the connection, property, geometry, and temperature data for all elements connected to the pivot point. Hence, data for an element will appear n times in the ECPT, where n is the number of points defining the element.

To generate a particular six-by-six element stiffness matrix partition $[K_{ij}]$, it is often necessary to generate the entire element stiffness matrix $[K]$. However, only those partitions $[K_{ij}]$, where i is the pivot point and $j = 1, 2, \dots, n$ (n being the number of grid points defining the element), are useful for the current ECPT record being processed, i.e., are useful for the current columns (or column if the pivot point is a scalar point) being generated. The unused partitions are recalculated and used when $j \neq i$ appears as a pivot point in a subsequent ECPT record. In the latest versions of NASTRAN, the entire matrix for each element is generated using the EST data block. The EMG module stores these matrices and an associated dictionary (KDICTION, MDICTION, etc.) on random access file storage. The EMA module retrieves the matrix partitions as they are needed and assembles them into a stiffness, mass, or damping matrix.

Although the matrices generated by EMG, EMA, DSMG1, and PLA4 are symmetric, NASTRAN generates complete columns and retains them for efficiency in succeeding matrix operations. This is necessary because all matrix operations are performed one column at a time (see Section 2 of the Theoretical Manual). Moreover, the availability of symmetric matrices by rows or columns is advantageous in some of the matrix operations.

6.8.2.4.2.2 Element Stiffness Matrix Partitions

Although the actual equations for the element stiffness matrices are different for each element, they follow a definite pattern. The six-by-six element stiffness matrix partition, $[K_{ij}]$, for the six columns related to point j and the six rows related to point i is given by

$$[K_{ij}] = [T_i]^T [K_e] [T_j] , \quad (3)$$

where $[T_i]$ and $[T_j]$ are the global coordinate system orientation matrices associated with grid points i and j , and $[K_e]$ is an element stiffness matrix in a coordinate system that is element-dependent. The matrices $[T_i]$ and $[T_j]$ are calculated from data in the CSTM data block, and $[K_e]$ is calculated from: a) connection, property, geometry, and temperature data from the EST or ECPT data block; and b) material property data from the Material Property Table (MPT) and the Direct Input Table (DIT) data blocks.

6.8.2.5 Utility Routines

A number of utility routines are available to all the structural modules. The matrix generation modules use double precision versions of these routines; the load vector generation and data recovery modules use corresponding single precision versions. These utility routines are:

1. GMMATD - General Matrix Multiply and Transpose (Double Precision). Section 3.4.32 describes GMMATD. A particular restriction is that all matrices in the calling sequence are stored by rows.
2. INVERD - In-core Matrix Inverse (Double Precision). Section 3.4.34 describes this standard matrix inversion routine.
3. PREMAT (with secondary entry point MAT) - Material Property Utility (Single Precision). PREMAT stores portions of the MPT and DIT data blocks in core, and MAT retrieves them when called by an element routine. See Section 3.4.36.
4. PRETRD (with secondary entry point TRANSD) - Utility for Modules that Use the CSTM Data Block (Double Precision). PRETRD reads the CSTM data block into core, and element routines call TRANSD to generate the matrices $[T]$ in Equation 3. See Section 3.4.37.

ADDING A STRUCTURAL ELEMENT

Single precision versions of these routines exist for use in modules SSG1, SDR2, and PLA3, all of which use single precision arithmetic. These single precision routines are: GMMATS, INVERS, and PRETRS (with secondary entry point TRANSS), and they are documented in Section 3.4.33, 3.4.35, and 3.4.38 respectively. The outputs of the MAT routine are single precision, and the element routines of the double precision matrix generation modules store them in double precision cells or convert them to double precision via the DBLE function prior to arithmetic calculations.

6.8.3 Specific Checklists

This subsection contains specific checklists for the modules that must be changed to add a new element to NASTRAN. These checklists should be used in conjunction with the program source code and the NASTRAN documentation, particularly the Module Functional Descriptions in Section 4 and the Structural Element Descriptions in Section 4.87.

6.8.3.1 Input File Processor (IFP)

An element connection (e.g., CNEWEL) bulk data card must be designed and added to the set of admissible bulk data cards. An element property card (e.g., PNEWEL) may have to be added. IFP processes both of these bulk data cards.

There are four major references for the IFP programmer: Section 2.4 of the User's Manual gives a functional description of each bulk data card; Section 2.3.2 of the Programmer's Manual describes the format of the output data blocks generated by IFP; Section 4.5 of the Programmer's Manual contains a description of the processing that occurs within IFP; and the source listings contain implemented element-dependent code.

6.8.3.1.1 Card Design

Before proceeding with the actual coding interfaces in IFP, it is necessary to discuss some aspects of the designs of the cards.

The element mnemonic, minus the "C" for "connection", must be no more than six characters, so the entire mnemonic is less than or equal to seven characters. This restriction is necessary because the card may be used as a double-field card (see Section 2.4 of the User's Manual). A

MODIFICATIONS AND ADDITIONS TO NASTRAN

similar restriction of course holds for the property card. For consistency and ease of use, follow the existing convention that the connection and property cards have the same mnemonic, except for the leading "C" and "P".

On the connection card, field 2 must be the element identification number (a positive integer), and field 3 must be, if a property card is defined, the property identification number (a positive integer). The next fields are reserved for the grid point identification numbers (positive integers) for the connecting grid points. The order of the grid points on the connection card defines the positive orientation of the element. Other connection information follows. If no property card is defined, the material identification number (again a positive integer) follows. Before card design is initiated, the reader should study existing connection cards documented in Section 2.4 of the User's Manual.

Field 2 of the property card must be the property identification number (a positive integer). Field 3 is the material identification number (a positive integer) which will be used, during program execution, to reference a material identification number on a MAT1, MAT2 or MAT3 material property card. The remaining fields are used for the element properties, which for the most part are real numbers. Required element properties must be listed first. This is done so that if a continuation card(s) is necessary to define all the allowed properties, it may be possible to define all the required element properties for a particular application with only the first card (e.g., see the PBAR card). Additionally, those properties that may lead to ill-conditioned element stiffness matrices or to operating system interrupts (division by zero, for example) should be restricted. Typical restrictions might be areas and thicknesses must be positive.

6.8.3.1.2 Coding Changes

IFP is, for the main part, table-driven. Entries in common blocks control the card data processing. These entries are initialized in seven Block Data subprograms, IFX1BD, $i = 1, 2, \dots, 7$.

After the reader updates the Block Data subprograms, he updates a computed-go-to statement which performs a branch on internal card identification number (one of the entries in the tables).

ADDING A STRUCTURAL ELEMENT

The statement number to which the branch is made is a call to one of the subroutines IFS1P, IFS2P, IFS3P or IFS4P, wherein card-dependent code must be added. Specific checklists follow.

6.8.3.1.2.1 Block Data Updates

1. The names (mnemonics) of the connection and property cards must be added to /IFPX1/, which is initialized in Block Data subprogram IFX1BD. The position of the names in this table defines the internal card identification number. Currently, over two-hundred (200) cards are defined. For the purposes of this discussion, let us assume that exactly two-hundred (200) cards are defined. The new connection and property cards will then have internal card numbers 201 and 202 respectively. The names must be defined as the first words of the I array in IFX1BD.
2. /IFPX2/, initialized in Block Data subprogram IFX2BD, contains two words per card type in the order of ascending internal card identification numbers. The first word of each pair gives the GINØ output file number, and the second gives the approach acceptability flag (see Tables 1 and 2 in section 4.5). All connection cards are output on the GEØM2 data block, which for IFP purposes is designated file number 8; and property cards are output on the EPT data block, which for IFP purposes is designated file number 2. We make the assumption for our present purposes that the element is acceptable for any problem approach: force, displacement or DMAP. Hence, the four words to be updated are: 8, 0, 2, 0. These four words should be added to words 41 through 44 of the array I3 in IFX2BD.
3. /IFPX3/, initialized in Block Data subprogram IFX3BD, contains two words per card type in the order of ascending internal card identification numbers. IFP uses the first word of each pair as the Conical Shell Problem flag; the second word contains the number of words to be output to the GINØ output file. No element connection or property cards other than CCØNEAX and PCØNEAX are allowed for a Conical Shell Problem; all second words of the pairs in /IFPX3/ are set to zero. Hence, the four words to be updated are: -1, 0, -1, 0. These four words should be added to words 41 through 44 of the array I3 in IFX3BD.
4. /IFPX4/, initialized in Block Data subprogram IFX4BD, contains two words per card type in the order of ascending internal card identification numbers. The first word of each pair is the smallest multiple of 4 greater than or equal to the number of required data items on the card, and the second word is 4 more than the smallest multiple of 4 greater than or equal

to the number of allowed data items on the card. Mathematically, let r be the required number of data items; let a be the allowed number of data items ($r \leq a$); let F be the value of the first word of the pair; and let G be the value of second word of the pair. Then:

$$F = 4 \left\lceil \frac{r + 4}{4} \right\rceil \text{ and } G = 4 + 4 \left\lceil \frac{a + 2}{4} \right\rceil$$

where $[x]$ is the greatest integer $\leq x$. For example, on the CBAR card (see Section 2.4.2 of the User's Manual), the data items on the continuation card are not required so that $r = 8$. Clearly, $a = 16$. Hence, $F = 8$ and $G = 20$. The four words must be added to words 41 through 44 of the array I3 in IFX4BD.

5. /IFPX5/, initialized in Block Data subprogram IFX5BD, contains two words per card type in the order of ascending internal card identification numbers. The first word of each pair is an index into /IFPX7/, and the second word is the field-2-uniqueness-check flag. For every bulk data card, each field except the first (the card mnemonic which is always BCD) is defined as either blank, integer, real, BCD, or double precision. An internal code has been established (0 = blank, 1 = integer, 2 = real, 3 = BCD, 4 = double precision, 5 = anything is permitted). Hence, a string of integers each between 0 and 5 describes a card's format. The length of the string is the number of fields allowed, including interior blank fields (e.g., fields 2, 4 and 5 on a BARØR card) but not including trailing blank fields. This string is contained in /IFPX7/, and the first word of the pair in /IFPX5/ is an index to the first word of this string. Before adding a new string, the reader should search /IFPX7/ to see if an existing string corresponds to the strings needed for the new "C" and "P" cards. We can make a check for duplicates in field 2 on both the "C" and "P" cards. We can do this because: (a) no duplicates are permitted in the set of all element identification numbers (similarly for the set of property identification numbers associated with an element type); (b) the Bulk Data Deck is sorted prior to IFP processing; and (c) IFP "looks at" two successive cards at a time.

6. /IFPX6/, initialized in Block Data subprogram IFX6BD, contains two words per card type in the order of ascending internal card identification numbers. IFP uses the two words as header information in the logical record associated with the card type (all the cards of one type are written in one logical record). The first word defines a card-type identification code, and the second word defines a bit position in a 96-bit "trailer." Modules that read,

ADDING A STRUCTURAL ELEMENT

via utility subroutine LOCATE, data blocks output by IFP use these two items. For the connection card, all the assigned card-type identification codes and trailer bit positions for GEOM2, which are listed in Section 2.3.2.2 must be searched and new unique ones chosen. The identification code can be any positive integer, but the trailer bit position must be between 1 and 96. A similar search and choice must be made for the two similar numbers for the property card that will reside on the EPT data block, documented in Section 2.3.2.5.

7. /IFPX7/, initialized in Block Data subprogram IFX7BD, contains format code strings. See paragraph 5 above for details.

6.8.3.1.2.2 Main Program and Card-Dependent Code Changes

The Block Data subprograms having been updated, make the following changes:

1. Update, in subroutine IFP, the computed-go-to statement which branches on internal card identification number. This statement follows the comment statement

C CALL SECONDARY ROUTINE TO EXAMINE EACH TYPE OF CARD

2. Add card-dependent code to one of the four card-dependent subroutines, IFSiP, $i = 1, 2, 3, 4, 5$. IFSiP processes most of the connection and property cards. Therefore, for consistency, the programmer will probably choose it. Observe that the same code is used for card types which are similar in format. The Bulk Data Card Descriptions in section 2 of the User's Manual in effect lay down the coding specifications. Observe that:

- a. The reader must update one of the three computed-go-to statements at the beginning of IFSiP (note that K is the internal card number, $KX = K - 100$, and $KY = K - 200$).
- b. The integer array M does not contain the card mnemonic, so that M(1) contains field 2, M(2) contains field 3, etc.
- c. N is the number of words to be written on the output file. It must be set in the card-dependent code.
- d. M is EQUIVALENCed to the real array RM which must be used for floating-point comparisons or operations.
- e. If the new card(s) has a format identical to an existing one, the reader might choose to use the existing code.

6.8.3.2 Executive Control Section Analysis (XCSA)

XCSA is responsible for the transmission to the remainder of the program of the restart tables associated with the rigid format selected by the user. However, only a single portion, the Card Name Tables, of the restart tables must be updated. Each rigid format and its associated restart tables are stored in a subroutine. For rigid format i ($i = 01, 02, \dots, 09, 10, 11, 12$), the routine name is LDi.

Two entries must be updated in LDi, one for the new "C" card and one for the new "P" card. Assume the new "C" card is CNEWEL and the new "P" card is PNEWEL. Then the entries are:

4HCNEW, 4HEL $\wedge\wedge$, 2

and

4HPNEW, 4HEL $\wedge\wedge$, 3

where \wedge denotes a BCD blank. The numbers 2 and 3 refer to bit positions of entries in a master execution mask (see Sections 3.1 + 1.3.1, ($i = 1, 2, \dots, 12$) of the User's Manual). These updates must be made after the last entry in the DATA statement for the array INM. All 12 LDi subroutine programs must be so updated.

The last three statements in each LDi subroutine are:

CALL WRITE (NPTP, INM, m, 1)

RETURN

END

Change the argument m in the CALL WRITE statement. The CALL WRITE statement writes n words of the INM array (onto the New Problem Tape, NPTP, with an end-of-record mark). The number m , which varies with the LDi routine, must be incremented by 6.

6.8.3.3 Geometry Processor and Table Assembler Modules (GP1, GP2, GP3, and TA1)

Data in common block /GPTA1/ entirely controls processing of element data in GP1, GP2, GP3 and TA1. Block Data program GPTABD initializes these data. Section 2.5.2.1 contains a description of /GPTA1/. When adding an element, the reader must change /GPTA1/ in the following ways:

ADDING A STRUCTURAL ELEMENT

Words for Table Header

Change

- | | |
|---|--|
| 1 | Increase this word, the number of entries, by 1. Call this new number of element entries n. |
| 2 | Increase this word, the pointer to the first word of the last element entry, by the number of words per entry, currently 24. |

Words for New Element Entry

Change

- | | |
|-----------|--|
| 1,2 | Include the new element's connection card mnemonic, e.g., CNEWEL. |
| 3 | Assign a new element-type identification number (this number should be n for consistency). |
| 4 thru 24 | Complete these items in accordance with definitions in Section 2.5.2.1. |

NOTE WELL: Structural modules that read the ECPT, EST, ECPTNL and ESTNL data blocks will use the new element-type identification number, n. Hence, this number must be determined and communicated early in the development process to all involved.

6.8.3.4 Plot Processor (PLØT)

If the element has special plotting requirements, subroutine LINEL should be updated. This routine defines and reads the element connection table (ELSETS) and creates the sequence of lines for drawing structural shapes. This routine is called because the number of grids is greater than four. If the element is not defined, no shape will be plotted. Additional tests on element type may also be made (e.g., the tetrahedron element) to call LINEL. Note that if the number of grids is 3 or 4 the first and last point are drawn as connected.

Element labeling for elements with more than four grid points, if desired, will need changes to subroutine ELELBL. Currently no labels are drawn since the plot would be too cluttered.

If more than 20 or less than 1 grid points exist for the element, module PLTSET subroutine SETINP must be changed. This restriction is to reject point elements and protect arrays dimensioned at 20 throughout the PLØT module.

MODIFICATIONS AND ADDITIONS TO NASTRAN

6.8.3.5 Element Matrix Generator (EMG)

A principal interface in adding a new element is to occur in the EMG module where the element stiffness, mass, and damping matrix partitions are generated. The changes are of two kinds: the element driver routine (EMGPRØ) must be modified to call the new element routine, and the new element subroutine itself must be added. Both single and double precision element routines may be included to be selected on user option.

ADDING A STRUCTURAL ELEMENT

6.8.3.5.1 Block Data and Module Driver Changes

At least one change, and possibly a second, must be made to include a new element. These changes are:

1. Update the array IG in block data TALABD, which defines the overlay segment in which the new element routine is to reside. (The EMG overlay structure is illustrated in Section 5.)
2. If the number of overlay segments for element routines must be increased, then the subroutine EMGSØC must be modified.

The following changes must be made to module driver, subroutine EMGPRØ.

1. Update the computed-go-to statement at FØRTRAN statement which performs a branch on internal element-type identification number. For existing elements, that do not contribute to the stiffness matrix, e.g., CØNMI, insert a transfer to read the next element type. Most elements do contribute to the stiffness matrix, and for these elements a transfer is made to an element routine call statement.
2. Insert a call to the new element stiffness matrix routine. This call must be followed by a transfer (GØ TØ) statement to read the next element type. This call statement has, in general, no arguments. The element's EST entry is passed to the element routine via /EMGEST/. The name of the element routine should use the element name (i.e., BAR) with a "D" or "S" appended to denote double or single precision arithmetic.

6.8.3.5.2 Coding the Element Subroutine

When coding the new element subroutine, follow this check list:

1. Document the element's EST entry at the beginning of the routine via comments.
2. Maintain the order to FØRTRAN specification statements given in Section 6.2.
3. Restrict the common block interfaces to the following:
 - a. /EMGPRM/ Use the following test variables:

<u>Word Number(s)</u>	<u>Symbol</u>	<u>Type</u>	<u>Description</u>
2	JCØRE	Integer	Location of first available word of open core
3	NCØRE	Integer	Location of last available word of open core

MODIFICATIONS AND ADDITIONS TO NASTRAN

<u>Word Number(s)</u>	<u>Symbol</u>	<u>Type</u>	<u>Description</u>
16-18	STIFF MASS DAMP	Integer	Indicates which type of matrices are desired. 0 = not.
19	MØDPREC	Integer	Indicates requested precision of output matrices
20	ERRØR	Logical	To be set = .TRUE. if an error is encountered
21	HEAT	Logical	.TRUE. if heat transfer matrices are desired
22	CØUPMASS	Integer	= 1 if coupled mass desired

Note: Only JCØRE and ERRØR may be modified by the element routine. If the sub-routine uses open core, JCØRE should be incremented a corresponding amount.

b. /EMGDIC/ Fill in the following integer data:

<u>Word Number(s)</u>	<u>Symbol</u>	<u>Description</u>
2	LDICT	Number of words in element dictionary = 5 + NGRIDS
3	NGRIDS	Number of connected grid points (SIL ≠ 0)

4. Use the utility routines GMMATD, TRANSD, INVERD, and MAT.

a. /MATIN/ and /MATØUT/. These are input and output data common blocks for the material property utility subroutine MAT (see Section 3.4.36). The outputs from MAT in /MATØUT/ are single precision, and they must be stored in double precision locations prior to arithmetic computations.

b. /SMA1DP/. This common block contains double precision variables which, for most programs, would be subroutine local variables. It is used so that open core (see Section 1.5) for SMA1 can be as long as possible. The use of this common block implies that element subroutines are not reentrant.

5. Generate M columns of the N-by-N full element matrix where N = number of grids times degrees of freedom. The output of matrix data is accomplished by the following subroutine call:

CALL EMGØUT(SDATA,DDATA.LDATA.EØE.DICT.FILE.INPREC), where: .

SDATA,DDATA = Single or double precision array of the matrix data. Contains a set of matrix columns corresponding to one or more grid points (DDATA may be same array as SDATA).

ADDING A STRUCTURAL ELEMENT

LDATA = Number of matrix terms in SDATA or DDATA.

EØE = End of element matrix data flag. If EØE > 0, the particular element matrix is finished.

DICT(NDICT) = Element dictionary array, where NDICT = 5 + number of connected points. Set the following words:

DICT(1) = ESTID = Internal element identification

DICT(2) = Form of element matrix

1 = square, full

2 = diagonal

DICT(3) = Order of element matrix (length of a column)

DICT(4) = Component code word. Integer representation of encoded bits which indicate connected degrees of freedom. For instance, if components 1, 4, and 6 are the connected degrees of freedom for each grid point, the code word will be:

$$\begin{aligned}\text{DICT}(4) &= \dots 000101001_2 \\ &= 2^0 + 2^3 + 2^5 = 41\end{aligned}$$

DICT(5) = Damping constant, g_e . (Only defined for stiffness matrices.)

FILE = Integer with value 1, 2 or 3, indicating stiffness, mass or damping terms, respectively.

IPREC = Precision of matrix data (1 = single, 2 = double).

The element data may be output as partitions corresponding to one or more grid points. (The entire matrix may be output with one call to EMGØUT). The matrix is stored by columns, each corresponding to one component of a grid point, or length equal to all degrees of freedom connected to the element. The order of the rows and columns corresponds to the grid points, sorted on SIL values, and the components given in the value DICT(4). All grid points must have the same connected components.

If the element subroutine coder has any questions or doubts, he should consult the listings of RØDD, SCALED, and EMGØUT as general examples.

6. For heat transfer analysis element matrices:

MODIFICATIONS AND ADDITIONS TO NASTRAN

- a. Test the logical variable HEAT in /EMGPRM/. IF .TRUE. perform the following.
- b. Use subroutine HMAT (see Section 3.4.73) to obtain conductivity and heat capacitance material data.
- c. Generate K (conductivity) and B (capacitance) matrices with one degree of freedom per connected point.
- d. Call EMGOUT for each matrix with:
 DICT(3) = NGRIDS
 DICT(4) = 1
 FILE = 1(K) or 3(B)

6.8.3.6 Static Solution Generator - Phase 1 (SSG1)

Subroutine EDTL (which has a secondary entry point, TEMPL) processes the EST data block one element at a time to compute temperature and enforced deformation loads. When an element is added to NASTRAN, update EDTL whether or not temperature or enforced deformation loads are defined for the new element. If the element has thermal or enforced deformation loading, the reader must code an element routine. If the element is used in heat transfer analysis, a heat input load (QVØL) may be applied to the element if code is added to the QVØL subroutine.

Currently subroutine QVØL is called by SSGSLT. The total heat input ($Q \cdot \text{volume}$) is divided equally between the connected points. The result is element independent. If additional elements follow the same rules, and have 8 points or less, modifications are simple.

ADDING A STRUCTURAL ELEMENT

6.8.3.6.1 EDTL Changes

Make the following changes to subroutine EDL:

1. Add an entry to the local array NECPT. NECPT(I) is the number of words in the EST entry for the element whose internal element-type identification number is I (recall I is set in GPTABD, see Section 6.8.3.3).
2. Change the computed-go-to element type to reflect the new element. If the new element does not have thermal or deformation loads defined, insert a transfer (to FORTRAN statement number 610) to skip the entire EST record; if it does, change the computed-go-to statement so that it points to element-dependent code in EDTL.
3. Add element-dependent code to EDTL which will:
 - a. Read the EST entry into /TRIMEX/.
 - b. Look-up the temperature at each grid point associated with the element. This look-up is accomplished via a call to subroutine FGPTT (see Section 4.41.11.27).
 - c. Call the element routine, passing the temperatures at the grid points and the beginning of the load vector array (at CORE(1)) through the calling sequence.

6.8.3.6.2 Coding the Element Routine

When coding the element routine follow this checklist:

1. Same as No. 1 in Section 6.8.3.5.2.
2. Same as No. 2 in Section 6.8.3.5.2.
3. Restrict the common block interfaces to the following:
 - a. /TRIMEX/. This block is 100 words in length and contains the element's EST entry.
 - b. /MATIN/ and /MATOUT/. These are used as input and output blocks for subroutine MAT.
4. Perform all arithmetic operations in single precision.
5. Use subroutine MAT to fetch material properties; MPYL and MPYLT are module utilities available for in-core matrix multiplication (subroutine GMMATS may alternatively be used); BASGLB and GLBBAS are module utilities that compute coordinate system transformation

MODIFICATIONS AND ADDITIONS TO NASTRAN

matrices (TRANSS may alternatively be used). See Section 4.41.11 for information on MPYL, MPYLT, BASGLB and GLBAS.

6. The scalar index numbers (internal degree-of-freedom numbers) in the element's EST entry are direct pointers into the load vector where the loading contributions should be added, i.e., the element routine does its own "insertion".

7. "Lift" as much code as possible from the element stiffness routine KNEWEL of module SMA1.

6.8.3.7 Stress Data Recovery - Phase 2 (SDR2)

The SDR2 module is divided into five stages. Two of the stages, stage III and stage V, deal with recovery of stress and force data for elements. In the following paragraphs, we will call these two stages phase 1 and phase 2, respectively.

Phase 1 computes and saves on a scratch file for phase 2 processing element stress matrices along with element properties, which are dependent upon the element. Phase 2 uses the outputs of phase 1 in conjunction with displacement vectors $\{u_i\}$ to compute final stress and force data in the elements.

Make changes to a module Block Data program and the driver routines for phase 1 and phase 2; also code two element routines, Sxxxx1 and Sxxxx2, where "xxxx" are four letters of the element's mnemonic, e.g., xxxx might be NEWL.

6.8.3.7.1 Driver Routine Changes

In subroutine SDR2B, the phase 1 driver, update the computed-go-to statement that performs a branch on element type, and add a call to Sxxxx1.

In subroutine SDR2E, the phase 2 driver, update the computed-go-to statement, and add a call to Sxxxx2. To output complex stresses and forces, increase the dimension of the CØMPLX array, and add a pointer string for stresses and forces if one does not currently exist. If complex stresses and forces are not permitted, set the two CØMPLX pointers in Block Data subprogram containing /GPTA1/ to 0.

ADDING A STRUCTURAL ELEMENT

6.8.3.7.2 The Element Routines

The following set of rules apply to both element routines:

1. Same as No. 1 in Section 6.8.3.5.2.
2. Same as No. 2 in Section 6.8.3.5.2.
3. Perform all arithmetic operations in single precision.
4. Use the utility routines TRANSS and GMMATS.

6.8.3.7.2.1 The Phase 1 Element Routine Sxxxx1

The following two items apply to the phase 1 element routine:

1. Restrict the common block interfaces to the following:
 - a. /SDR2X5/. The EST entry is contained in /SDR2X5/. At the conclusion of Sxxxx1, /SDR2X5/ contains the outputs of the routine: stress matrices and miscellaneous element properties. Output the scalar index numbers of the EST entry for use in phase 2.
 - b. /SDR2X6/. Use for scratch storage.
 - c. /MATIN/ and /MATOUT/. Used by MAT.
2. MAT is available to Sxxxx1, but not to Sxxxx2.

6.8.3.7.2.2 The Phase 2 Element Routine Sxxxx2

The following remarks apply to Sxxxx2:

1. The common block interfaces are:
 - a. /SDR2XX/. This block defines open core where the phase 2 driver stores the displacement vector.
 - b. /SDR2X4/. Three words are used, the 36th, 38th, and 39th. Word 36 is an index into /SDR2XX/. It points to the first word of the displacement vector. Word 38 is an element loading temperature, and word 39 is an element deformation. If thermal and deformation loading were not defined in SSG1, do not use these latter two words.

MODIFICATION AND ADDITIONS TO NASTRAN

c. /SDR2X7/. This block contains the computed element stresses. The element identification number, the first word of the EST entry, must always be the first word of /SDR2X7/.

d. /SDR2X8/. This area is work space and varies in content for each element.

2. The scalar index numbers are indexes to those components of the displacement vector

6.8.3.7.3 Heat Transfer "Stress Recovery"

Data recovery for the elements in a heat transfer problem is done with three subroutines. If heat transfer calculations are added to other elements they may be processed with new routines (Phase 1 and Phase 2) or the element calculations may be added to SDHTFF, SDHTF1, SDHTF2 subroutines.

SDHTF1 moves the EST data into a fixed data array /SDRX6/, calls the material routine HMAT, and calls SDHTFF to calculate the heat flow-temperature matrices. A new element would require an addition to the "P0INTR" array and minor code changes.

SDHTFF calculates the Phase 1 stress recovery matrices. All elements use a fixed output format (words 101-146 of /SDR2X5/. If more than 8 points are used, it must be changed.

SDHTF2 is the general Phase 2 element output calculation routine. All elements at present have the same input and output data arrays, the only difference is the number of points (NSIL) and the dimension of the flow (1, 2 or 3). The basic equation for the gradients and flow vectors are

$$\{\nabla T\} = \sum \{C_i\} T_i$$

$$\{q\} = [K]\{\nabla T\}$$

where the dimensions of K are NQ x NQ and the dimensions of $\{C_i\}$ are NSIL x 1.

The temperatures T_i are extracted from core $T(I) = ZZ(IVEC - 1 + SIL(I))$. The output is placed in /SDR2X7/.

ADDING A STRUCTURAL ELEMENT

6.8.3.8 Output File Processor (ØFP)

ØFP prints, with headings, element stress and/or force output. Each new element needs a unique heading and entries to describe the output format desired. This heading is in addition to the standard title, subtitle, and label that will be printed from information in the Case Control Deck. Page ejection is element-independent and automatic, with headings reprinted on each page of output until all data records have been printed.

6.8.3.8.1 ØFP Design

ØFP is, like IFP, essentially table-driven. This technique allows ØFP to avoid many lengthy format statements. In ØFP standard format statements contain the headings, and the formats for the items data record are built from tables.

Before modifying ØFP, the programmer should be familiar with the documentation references for ØFP, particularly those describing element stress and force output. Section 4.70 and the source listings for the subroutines described in that section are where the general descriptions for ØFP can be found. Sections 2.3.51 and 2.3.52 describe the specific makeup for element stress and force data blocks. ØFP will expect any new element type identification record to have a similar format as the other element types (see the description of record 1 in Section 2.3.28.15). ØFP also assumes that the number, order, and type of the new element's data record(s) have been determined (see description for record 2 of Section 2.3.28.15). Normally, SDR2 has set these data records for ØFP depending on the required output. The coding changes for ØFP will depend on a) the headings desired and b) the number, order, and type of items in the data record.

6.8.3.8.2 Adding the Headings

New elements may require many different headings, such as forces for the real case, forces for the complex case, stresses for the real case, and stresses for the complex case. And then all of these cases can be in SORT I or SORT II format. Changes have to be made for each case.

Each heading in \emptyset FP consists of five lines of output. Each line is printed by a separate write statement, so if any two lines are the same, then the same statement can be used. Thus, when the analyst and the programmer are deciding on wording and spacing of the headings, they should lay them out in a consistent manner. They may be able to use existing formats for some lines.

The basic heading pointer is contained in the doubly subscripted B array (see Section 4.70.9). The actual line pointers are in the C array, and format statements in subroutines \emptyset FP1 and \emptyset FP1A contain the output headings. The pointer to the B array is set in word 2 of each input identification record (record 1) (SDR2 has set this word correctly for the new element). The reader then computes CP \emptyset INT, where CP \emptyset INT is an index into the C array. To do this, word 2 of record 1 and the element-type identification number (see Section 6.8.3.3) are needed for each desired type of output. The value of C(CP \emptyset INT) is a pointer into the D array, which will define the format string for the data record. The values of the next five entries in the C array, (C(CP \emptyset INT + 1) to C(CP \emptyset INT + 5)) are used as numbers in a computed-go-to statement in \emptyset FP1. Each of these five numbers refers to a write statement, which prints one line of the heading. \emptyset FP1 is so large that the computed-go-to statement has been continued in \emptyset FP1A. Thus, new format statements and write statements should be added to \emptyset FP1A. As new write statements are added, the computed-go-to near the beginning of \emptyset FP1A and the test for the maximum number of write statements must be updated.

6.8.3.8.3 Adding the Data Record Code

C(CPØINT) is equal to DPØINT which is an index to the next available space in the D array. The D array is contained in the Block Data program ØFP1BD. The value of D(DPØINT) is a packed four-digit number. The right two digits give the number of output lines the data record will produce, and the left two digits give the number of data records used per line. If the left two digits are null, then only one data record is used per line.

Values for entries D(DPØINT + 1) to a D(DPØINT + N), whose value is 0, define pointers to the E array or ESINGL array. Positive values point to the E array by the formula $(5 * D(DPØINT + I) - 5)$, where I varies from 1 to N. Negative values in the D array are used as pointers into the ESINGL array by the formula $|D(DPØINT + 1)|$. Both the E array and the ESINGL array are contained in the Block Data program ØFP5BD. Both of these arrays contain pieces of a format statement, and the programmer strings the pieces together to space and place the data items correctly under the heading. The programmer can use the pieces that are listed or add more. The only rule is to exclude the ending comma from any piece of a format statement. In general, all the necessary format pieces to describe a line of a data record are included in the E and ESINGL arrays.

6.8.3.9 Differential Stiffness Matrix Generator - Phase 1 (DSMG1)

The element interfaces with DSMG1 are in two phases. Phase 1 reads the ECPT data block and, for each element, appends displacement vector components of the associated grid points, an average element loading temperature, and an element deformation. Phase 1 writes this appended ECPT entry onto a scratch file which has the same general format as the ECPT. Phase 2 processes the scratch file in a fashion similar to ECPT processing in SMA1, so the differential stiffness element routines reside in the phase 2 portion of the module.

6.8.3.9.1 Phase 1 Changes

When appending the components of the displacement vector to the ECPT element entry in sub-routine DS1, either append the three translational components of displacement at each grid point or append all six components of displacement at each grid point. The test for this determination is made immediately after FØRTRAN statement number 300.

If an element type is not in the differential stiffness set (DSARY(1)=0), phase 1 does not write its ECPT entry on the scratch file.

6.8.3.9.2 Phase 2 Changes

Change the computed-go-to statement in subroutine DS1A that performs a branch on element-type identification number, and insert a call to the new element routine. Name the new element routine DNEWEL. The computed-go-to statement in DS1A need not be changed if the element is not in the differential stiffness set.

6.8.3.9.3 Coding the DNEWEL Subroutine

When coding the new element subroutine, follow this checklist:

1. Document the element's appended ECPT entry at the beginning of the routine via comments.
2. Same as No. 2 in Section 6.8.3.5.2.
3. Restrict the common block interfaces in DNEWEL to the following:
 - a. /DS1AAA/. The first word is the pivot point, NPVT.
 - b. /DS1AET/. This block is 100 words in length and is the means of communicating the element data from the ECPT data block to the element subroutines. Since the data in /SMA1ET/ are mixed (real and integer), an EQUIVALENCE must be used.
 - c. /MATIN/ and /MATOUT/. These are input and output data common blocks for the material property utility subroutine MAT (see Section 3.4.36). The outputs from MAT in /MATOUT/ are single precision, and they must be stored in double precision locations prior to arithmetic computations.
 - d. /DS1ADP/. This common block contains double precision variables which, for most programs, would be subroutine local variables. It is used so that open core (see Section 1.5) for SMA1 can be as long as possible. The use of this common block implies that element subroutines are not reentrant.
4. Make a test to ensure that the scalar index number (internal number for a grid point) for one of the connecting grid points matches the pivot point, NPVT.
5. Perform all arithmetic operations in double precision.
6. Use the utility routines GMMATD, TRANSD, INVERD and MAT.
7. Call DS1B (Section 4.27.8.3) to "insert" the six-by-six matrix partitions.

6.3.10 Piecewise Linear Analysis - Phase 1 (PLA1)

Make the following changes to module PLA1 if the added element is to be admissible to the set of elements for which Piecewise Linear Analysis is defined.

1. Change the computed-go-to statement (on element-type identification number), which reflects whether an element is in the set of elements for which Piecewise Linear Analysis is defined.
2. Add element-dependent code which initializes stress information appended to the ECPT and EST data blocks. The commented code in PLA1 along with the descriptions in Sections 2.3.34.3 and 2.3.34.4 for the ESTNL and ECPTNL data blocks respectively serve as models for the addition of this code.
3. Update the local array, PLAARY. $PLAARY(I) = 1$ if the I^{th} element type is an element for which Piecewise Linear Analysis is defined, and $PLAARY(I) = 0$ otherwise.

PLA1 uses the SMA1 module environment so the SMA1 element routines may be called to make contributions to the linear stiffness matrix. Hence all of the common blocks of SMA1 must be available to PLA1. Also the majority of the subroutines, including Block Data program SMA1BD, of module SMA1 must be available to PLA1. The only subroutines of SMA1 not needed are SMA1, SMA1A and DETCK (see Figure 13 in Section 5.2).

6.8.3.11 Piecewise Linear Analysis - Phase 3 (PLA3)

The element interfaces with PLA3 are in two phases. Phase 1, accomplished in subroutine PLA31, reads the ESTNL data block and, for each element, appends displacement vector components corresponding to the grid points of the elements. The output of phase 1 is a scratch file of appended ESTNL data to be processed in phase 2. Subroutine PLA32, the phase 2 driver, reads the scratch file and calls element routines which compute stresses and which update the accumulated stress data in the ESTNL entry.

6.8.3.11.1 PLA31 Changes

Make the following changes to subroutine PLA31:

1. Update the local arrays ESTWDS, which defines the number of words to be read from the ESTNL data block for each element, and NGPTS, which defines the number of grid points for each element. Both arrays are ordered by the internal element-type identification number defined in /GPTABD/ (see Section 6.8.3.3).

2. When appending the components of the displacement vector to the ESTNL element entry, either append the three translational components of displacement at each grid point or append all six components of displacement at each grid point. The default value is three. If six components per grid point are desired, update the logical IF statement between statement numbers 20 and 30.

6.8.3.11.2 PLA32 Changes

Make the following changes to subroutine PLA32:

1. Update the array ESTWDS, as defined in Section 6.8.3.12.1; the array NSTWDS, which defines the number of words per element written on the \emptyset NLES data block; and the array NWDSP2, which defines the number of words per entry in the scratch file. These arrays are ordered by element-type identification number.
2. Change the computed-go-to statement which performs a branch on element-type identification number, and insert a call to the new element routine, PSNEWL.

6.8.3.11.3 Coding the PSNEWL Subroutine

When coding PSNEWL, follow this checklist:

1. Document the element's appended ESTNL entry at the beginning of the routine via comments.
2. Same as No. 2 in Section 6.8.3.5.2.
3. The common block interfaces are:
 - a. /PLA32E/. Contains the element's appended ESTNL entry.
 - b. /PLA32S/. Scratch block for variables local to PLA3 element routines.
 - c. /PLA32C/. Contains two words GAMMA and GAMMAS, corresponding to γ and γ^* defined in Equations 2 and 3 of Section 4.54.
 - d. /S \emptyset UT/. Contains the computed element stresses in the order expected by \emptyset FP. The first word of /S \emptyset UT/ must be the element identification number, which is always the first word in the element's appended ESTNL entry.
 - e. /MATIN/ and /MAT \emptyset UT/. Input and output blocks for subroutine MAT.

ADDING A STRUCTURAL ELEMENT

4. Perform all arithmetic operations in single precision.
5. Use utility routines MAT, GMMATS, TRANSS, and INVERS.
6. After completion of the computations and prior to returning to PLA32, update the element routine, in the cells assigned to them in /PLA32E/, the new updated stress information. These updated data are used the next time PLA3 is executed.
7. Some of the code in PSNEWEL might be taken from subroutine KNEWEL of SMA1 and subroutines SNEWL1 and SNEWL2 of SDR2.

6.8.3.12 Piecewise Linear Analysis - Phase 4 (PLA4)

PLA4 has two phases. Phase 1, incorporated in subroutine PLA41, reads the ECPTNL data block, and, for each element, appends displacement vector components of the associated grid points. The appended element ECPTNL data are written on a scratch file in a format that is the same as that of the ECPTNL. The phase 2 driver, PLA42, processes the scratch file in a way similar to ECPT processing in SMA1, calling element routines which in turn generate stiffness matrix partitions of the nonlinear stiffness matrix.

6.8.3.12.1 Phase 1 Changes

Make the following changes to subroutine PLA41.

1. Update the local arrays NWØRDS and NGPTS. NWØRDS(1) is the number of words read from the ECPTNL data block for the I^{th} element type, and NGPTS(I) is the number of grid points associated with the I^{th} element type.
2. When appending the components of the displacement vector to the ECPTNL element entry, either the three translation components of displacement at each grid point or all six components of displacement at each grid point can be appended. The default value is three. If six are to be appended, then expand the logical IF statement:

IF (ELTYPE. EQ. 34) NWDS = 6

located between statement numbers 20 and 30.

6.8.3.12.2 Phase 2 Changes

Make the following changes to Block Data subprogram PLA4BD and subroutine PLA42.

1. Change the /PLA42C/ variables IØVRLY, NWØRDS, and possibly NLINKS in PLA4BD. The explanations in Section 6.8.3.5.1 of the variables of the same names in /SMA1CL/ apply here as well.
2. In PLA42, update the local array NWDSP2, which defines the number of words per element entry to be written on the ECPTNL output data block.
3. Change the computed-go-to statement in PLA42 which performs a branch on element-type identification number, and insert a call to the new element routine, PKNEWL, following this computed-go-to statement.

6.8.3.12.3 Coding the PKNEWL Subroutine

When coding PKNEWL, follow this checklist:

1. Document the element's appended ECPTNL entry at the beginning of the routine via comments.
2. Same as No. 2 in Section 6.8.3.5.2.
3. PKNEWL common block interfaces are:
 - a. /PLA42C/. The first word, NPVT, is the pivot point. The next two words are the same as GAMMA and GAMMAS in /PLA32C/ (see Section 6.8.3.11.2, paragraph 3(c)).
 - b. /PLA42E/. Contains the element's appended ECPTNL entry.
 - c. /PLA42D/. Scratch block for variables local to PLA4 element routines.
 - d. /MATIN/ and /MATØUT. Input and output blocks for MAT.
4. Same as No. 4 in Section 6.8.3.9.2.
5. Same as No. 5 in Section 6.8.3.9.2.
6. Same as No. 6 in Section 6.8.3.9.2.
7. Subroutine PLA4B performs the insertions (see Section 4.55.8.3). Insert only those six-by-six partitions corresponding to the pivot point.

ADDING A STRUCTURAL ELEMENT

8. After completion of the computations and prior to returning to PLA42, update, in the cells assigned to them in /PLA42E/, the new updated stress information. These updated data are used the next time PLA4 is executed.

9. Some of the code in PKNEWL might be taken from subroutine KNEWEL of SMA1 and subroutines SNEWL1 and SNEWL2 of SDR2.

6.8.3.13 Static Solution Generator - Heat Transfer (SSGHT)

If an element is used to generate a heat transfer conductivity matrix, it must be added to the SSGHT module for calculation of corrective heat flows for temperature-dependent conductivity. In the present code all elements are processed with two routines, each performing a separate calculation step.

The nonlinear calculations involve two phases. Subroutine SSGHT1 reads the EST data and sets up the call to HMAT, the material routine. If the material is nonlinear, the basic element data is converted to a common format in /ESTØUT/ and written for subsequent use. The contents of this table may vary for different elements since only the SSGHT2 routine will use the data. The contents of /ESTØUT/ for existing elements are:

- Element ID
- Subroutine type (1-dimensional, 2-dimensional, asymmetric)
- Name of element (2 words)
- Scalar indices (≤ 8 words)
- Material ID
- Area factor
- Orientation angle
- Grid point locations (3 x 8 matrix)
- Reference material data (6 words)

The Element Summary Table (EST) is read and the type of element is tested. If the element is accepted, the EST data is simply moved into the common format in /ESTØUT/, the reference material data is extracted by calling the MAT routine and added to /ESTØUT/, and the block (45 words) is written on a scratch file (FILE) provided by the main routine.

Subroutine SSGHT2 uses the element data created by SSGHT1 and the current solution temperature vector in core ($UNI = u_n^i$) to create a nonlinear correction load $\{\delta P\}$ by the equation:

ADDING A STRUCTURAL ELEMENT

$$\{\delta P\} = [K(T_i) - K(T_0)]\{T_i\} \quad (1)$$

where $[K(T_0)]$ is the conductivity matrix evaluated as in the EMG module using the original material, and $[K(T_i)]$ is evaluated at the correct temperature. For most of the elements

$$\{\delta P\} = [C]^T[G(T_i) - G(T_0)][C]\{T_i\} \quad (2)$$

where $G(T_0)$ is given in the /ESTOUT/ data, $G(T_i)$ is evaluated at the current temperature, and $[C]$ is a constant matrix.

The resultant loads δP_i on each point i are added to core in the position

$$\delta P(IG) = \delta P(IG) + \delta P(I) \quad (3)$$

where $IG = SIL(I)$.

To add a new element, the basic tasks are to add code to SSGHT1 to write the necessary element data on the scratch file, and add code to SSGHT2 to generate $\{\delta P\}$ from Equation (1). It is suggested that additional elements be coded as separate subroutines, called by SSGHT1 and SSGHT2, and overlayed to save critical core space in this module.

6.8.4 Updating the NASTRAN Manuals

Concurrent with coding and check out, update the NASTRAN manuals. Updating will apply to the three principal manuals, Theoretical, User's and Programmer's. If demonstration problems are formally devised, updates to the Report on the Demonstration Problems may be desired. However, only the interfaces with the three principal manuals will be discussed here.

The NASTRAN manuals have been designed to accommodate future additions and modifications. Each major section (e.g., Section 6 of the Programmer's Manual) stands alone with its own page numbers, equation numbers, figure numbers and table numbers, so that new sections can be added without significant disruption. In the following paragraphs, each major section of each of the three manuals will be examined to determine whether or not an update will be necessary.

6.8.4.1 The Theoretical Manual

The major sections in the NASTRAN Theoretical Manual are numbered up to 17. However, Section 6 has been reserved for future development. Although the structural analyst and not the programmer will probably update the Theoretical Manual, the following paragraphs, preceded by major section numbers, identify material that might be updated in that section.

2. No updates. The matrix operations discussed in this section are the system matrix operations routines and not the in-core matrix subroutines used by element routines.
3. Only Section 3.8.4, Element Algorithms for Piecewise Linear Analysis, is an update candidate. The new element may or may not (the membrane elements currently incorporated in NASTRAN are not admissible to piecewise linear analysis) be analyzed in the piecewise linear analysis rigid format.
5. This section is the primary candidate for updating. For existing elements, the discussion in this section falls short of a complete presentation of all the equations implemented in the program. (Section 8 of the Programmer's Manual contains the complete

MODIFICATIONS AND ADDITIONS TO NASTRAN

equations for each element.) Note that neither differential stiffness nor piecewise linear analysis material is presented in this section; the former is discussed in Section 7, and the latter in Section 3.8.

7. If the new element contributes to the differential stiffness matrix, a new major subsection must be written.

14. The new element may involve special modeling techniques. If this is the case and the analyst decides it is worthy of note, a new major subsection might be added to Section 14 rather than incorporating the special technique in Section 3.8, 5, or 7.

15. Significant error analyses performed during program development might be incorporated here.

6.8.4.2 The User's Manual

The NASTRAN User's Manual contains seven major sections. The following paragraphs, preceded by major section numbers, identifies material that might be updated in that section.

1. For each element in NASTRAN, Section 1.3 gives the mnemonics for the connection and (if defined) the property card; defines the basic structural and inertia properties of the element; describes the element coordinate system; lists the stresses and forces in the element; and gives diagrams for the element coordinate system and direction of element forces and/or stresses. Update this section.

2. Section 2.4.2 documents each NASTRAN bulk data card with a description. The connection card and, if defined, the property card must be documented. The design of these cards should be one of the initial tasks that the analyst and programmer perform.

3. Section 3 has 21 subsections. Section 3.1 is an introduction, and Sections 3.2 through 3.21 document each of the NASTRAN rigid formats. Section 3.1 does not have to be updated. Subsections 3.i.1 and 3.i.2 ($i = 2, 3, \dots, 21$) do not have to be updated. If the new element has both lumped and coupled mass matrix options in module SMA2, then update the explanation for the C0UPMASS parameter in Sections 3.i.3 ($i = 2, 3, 7, 9, 10, 15, 21$) and 3.j.4 ($j = 4, 5, 6, 8, 11, 12, 13, 14, 16, 20$).

ADDING A STRUCTURAL ELEMENT

4. If the new element can be plotted, then update the list of element types that can be specified on a SET definition card in the Structural Plotter request packet to reflect the new element's name. Additionally update the list of labels for element types that are for element-type identification. These lists are in Section 4.2.2.1. Update the tables of element-stress item codes and element-force item codes which are keys to what can be plotted with the XYPLØT module. These tables are at the end of Section 4.3.3.
6. If the programmer and analyst find that a general user message related to elements, such as message 2026 in Section 6.2.3, does not fulfill their needs, they may choose to add a new message to subroutine MSGWRT or USRMSG. If this is the case, Section 6.2.3 must be updated.
7. Assume the new element has the mnemonic NEWEL. Then add the following entries to the NASTRAN Dictionary:

CNEWEL	IB	New element connection definition card
NEWEL	IC	Requests structure plot for all NEWEL elements
PNEWEL	IB	New element property definition card

6.8.4.3 The Programmer's Manual

The NASTRAN Programmer's Manual contains nine major sections. The following paragraphs, preceded by major section numbers, identify material that might be updated in that section.

2. Make the following changes to Section 2:
 - a. Add the data on the new element connection card, CNEWEL, to the data block description for GEØM2, Section 2.3.2.2.
 - b. Add the data on the new element property card, PNEWEL, to the data block description for EPT, Section 2.3.2.5.
 - c. Update the data block description for EST, Section 2.3.8.1.
 - d. If the new element is defined for piecewise linear analysis, update the data block description for ESTNL, Section 2.3.34.3.

MODIFICATIONS AND ADDITIONS TO NASTRAN

- e. If the new element is defined for piecewise linear analysis, update the data block description for ECPTNL, Section 2.3.34.4.
 - f. Update the element stress and force output data descriptions in Sections 2.3.51 and 2.3.52 respectively.
3. If the material property options currently implemented in subroutine PREMAT, Section 3.4.36, are inadequate for the new element and PREMAT is consequently modified, then update this section.
 4. Section 4 contains numerous subsections that must be updated, some extensively. They are:
 - a. Update the index in Section 4.1.3 to include all the new entry points for element routines coded for EMG, SSG1, SDR2, DSMG1, PLA3 and PLA4.
 - b. Update Tables 1 and 2 at the end of Section 4.5.
 - c. Add a subroutine description for the new element matrix routine, e.g., NEWELD, to Section 4.124. (Of course add two descriptions if both a lumped mass matrix and a coupled mass matrix routine are added.)
 - d. For some elements (e.g., a triangular ring), SSG1 incorporates thermal loading in an element subroutine; for others, entry point TEMPL of subroutine EDTL is used. If a new element routine is coded, add a subroutine description to Section 4.41.11.
 - e. Add two subroutine descriptions for the new phase 1 and phase 2 stress data recovery element routines, e.g., SNEWL1 and SNEWL2, to Section 4.46.8.
 - f. For module DSMG1 (Section 4.49):
 - (1) Update the first paragraph of Section 4.49.7.
 - (2) If necessary, update indented paragraph 6 in Section 4.49.7.
 - (3) Add a subroutine description for the new differential stiffness matrix element routine, e.g., DNEWEL, to Section 4.49.8.

ADDING A STRUCTURAL ELEMENT

- g. For module PLA3 (Section 4.54):
 - (1) If necessary, update the second paragraph of Section 4.54.7.
 - (2) Add a subroutine description for the new piecewise linear analysis stress data recovery element routine, e.g., PSNEWL, to Section 4.54.8.
 - h. For module PLA4 (Section 4.55):
 - (1) If necessary, update the second paragraph of Section 4.55.7.
 - (2) Add a description for the new nonlinear stiffness matrix generation element routine, e.g., PKNEWL, to Section 4.55.8.
 - i. In Section 8, Structural Element Descriptions:
 - (1) Update Table 1.
 - (2) Add a new subsection to Section 8 to describe the equations used in the element routines for all structural modules. This section's introduction stated that it was assumed that a structural analyst has written mathematical specifications for the structural element. The analyst should write these specifications in a format similar to those in Section 8, that is, list the ECPT/EST input data; list coordinate system data obtainable from the CSTM data block; define the material property assumptions and data; list the equations common to all element routines; list the equations specific to stiffness and mass matrix generation, thermal and enforced deformation loading, stress and force data recovery, differential stiffness matrix generation, and stress data recovery and stiffness matrix generation for piecewise linear analysis.
5. Update the overlay diagrams in Section 5.2 as follows:
- a. Change link 8 to reflect the new element routines in EMG.
 - b. Change link 5 if a new subroutine was added to SSG1.
 - c. Change link 13 to reflect the new element routines added to SDR2, DSMG1, PLA3 and PLA4.

6.8.5 Dummy User Elements (DUM1 thru DUM9)

A capability exists within NASTRAN which permits a user to enter his own element subroutines for purposes of generating stiffness and mass matrix contributions, thermal load contributions and for computation of various stress, force, etc. outputs. Through the use of ADUMi, CDUMi, and PDUMi (i = 1 to 9) bulk data cards, he may enter geometry, property and connection data as is done for any other NASTRAN structural element. The difference is that this input data is of a dynamic nature based on the parameters he enters via the ADUMi card.

Thus a user who may have a structural element formulation not found within NASTRAN may, through the dummy element capability, implement it into NASTRAN with a lot less difficulty than would be the case of adding an entirely new element.

The procedure for utilizing a dummy element is:

1. Create an element stiffness routine KDUMi modeled after an existing routine (e.g., subroutine KRØD) which will compute and output via insertion routines one 6x6 matrix for each connecting grid point with respect to the connected pivot point. (Each connected grid point becomes the pivot point independently, see Section 1.8).
2. If desired, generate a similar routine MDUMi to compute the MASS matrix based on an existing routine (e.g., subroutine MRØD).
3. If desired, generate a routine DUMi similar to an existing routine (e.g., RØD) to compute a thermal load or element deformation load. At present, the user must update subroutine EDTL by adding a call to SSGETD; this may change the parameters in the call to DUMi.
4. If stress and/or force outputs are desired, generate two routines SDUMi1 and SDUMi2 similar to SRØD1 and SRØD2 to compute the stress and/or force outputs.
5. For any of the above routines prepared (KDUMi, MDUMi, DUMi, SDUMi1, and SDUMi2), a "Linkedit" run will be necessary to load these into the NASTRAN executable.

In the design of these element routines the "user programmer" needs to understand the format of the EST table (see Sections 2.3.8.1 and 2.3.8.3).

Note also common block table /GPTA1/. (See Section 2.5.2).

Note also the above sections (see Section 6.8.1 through 6.8.4).

ADDING A STRUCTURAL ELEMENT

All module code for the basic dummy elements has been provided in modules SMA1, SMA2, DS1, SSG1, and SDR2.

As element routines do not use NASTRAN executive functions, the user programmer should, before attempting a linkedit of his element routines into the NASTRAN executable, do the following:

1. Prepare a "load and go" environment which, e.g., simulates SMA1. This environment would contain a driver routine which would set up interface common blocks to the element routine KDUMi. KDUMi would call a dummy SMA1B insertion routine which might only print the matrices.
2. Prepare similar environments for the thermal and stress data recovery functions.

When the user-programmer feels he has element routines which are modeled correctly with respect to NASTRAN, a linkedit might be performed. Linkedits are in general non-trivial systems-programmer's tasks with large overlay programs, and thus all work possible should be accomplished before this is attempted.

Dummy output formats and headings are provided within NASTRAN. These encompass both real and complex (see Sections 2.3.51 and 2.3.52).

Refer to subroutine descriptions in Section 3 for the following subroutines:

1. GMMATD
2. GMMATS
3. INVERD
4. INVERS
5. PREMAT
6. PRETRS

PRINTED OUTPUT

6.9 PRINTED OUTPUT

The majority of the NASTRAN data scheduled for output are input to the Output File Processor (ØFP) module. This module performs the actual formatting and outputting of the data.

To implement additional output capability in the ØFP, the following ground rules should be observed in designing a data block intended to be input to ØFP for output to the system printer or punch unit.

1. The data block's name, for consistency, should begin with an "Ø".
2. There should be one or more repeating record pairs, where the record pair is of the following form:

<u>WORD</u>	<u>MODE</u>	<u>DESCRIPTION</u>	
1	Available for Data		Identification Record (An odd-numbered Record)
2	I	Major ID	
3	I	Minor ID	
4	I	Subcase ID	
5			
.	Available for Data		
.			
9			
10	I	Number of words per entry	
11			
.	Available for Data		Data Record (An even-numbered Record)
.			
50			
51	BCD		
.	.	32 Words Title	
.	.	32 Words Subtitle	
.	.	32 Words Label	
146	BCD		

Repeating data entries of the length specified in the preceding record. This record may be null but should exist.

MODIFICATIONS AND ADDITIONS TO NASTRAN

3. The Major ID is formulated depending on the data type as follows.

a. For a new data type an integer should be selected which equals I, where I is a value one greater than the current number of accepted data types within ØFP.

b. The Major ID then depends on the data classification

$$\text{Major ID} \left\{ \begin{array}{l} I = SØRT1 - \text{Real} \\ 1000 + I = SØRT1 - \text{Complex} \\ 2000 + I = SØRT2 - \text{Real} \\ 3000 + I = SØRT2 - \text{Complex} \end{array} \right.$$

4. The Minor ID is optional to distinguish subclasses of the Major ID.

With the data block format outlined above, the NASTRAN systems programmer may implement the new data formats within the current ØFP procedure by modeling from any current data type now handled by ØFP. The current ØFP has dynamic formatting for the outputting of entries of the Data Record. However, if the programmer is not familiar with the five level pointer procedure used for the dynamic formatting, it is recommended that he implement logic to explicitly output the Data Record.

ØFP will, without modification, output new data blocks via the table print routine, TABPT.

6.10 PLOTTER OUTPUT

Plotted output in NASTRAN is restricted to three modules: 1) the Structural Plotter (PLØT - see section 4.24); 2) the XY Plotter (XYPLØT - see section 4.69); and 3) the Matrix Plotter (SEEMAT - see section 4.74). Each of these modules uses the NASTRAN plotter software package of utility routines, each of which is individually documented in section 3.4.

6.10.1 Changes to the Plotter Software

In order to add a new plotter to the NASTRAN plotter software, the following must be done: 1) up to four (4) new subroutines (AXISi, LINEi, TYPEi, WPLTi) must be written, where "i" is the internal index of the new plotter (see section 3.1 for the current values of i); and, 2) changes and/or additions must be made to ten (10) existing subroutines (AXIS, FNDPLT, LINE, PLØTBD, SELCAM, SKPFRM, SYMBOL, TIPE, TYPFLT, TYPINT).

Initially, it should be decided which of the four possible new subroutines will be needed for the new plotter. For the most part, three will be the minimum: LINEi, TYPEi, WPLTi. If the new plotter has no typing capability, an existing subroutine, DRWCHR, will be used in place of TYPEi. It is strongly recommended that the calling sequences of any new subroutines be the same as those of their existing counterparts. None of the four possible new subroutines should present much difficulty to the programmer if he uses their existing counterparts as models.

Subroutine AXISi need only be written if the following two conditions are true on the new plotter: 1) lines must be drawn generally as a series of short lines; and 2) there does exist a special plotter instruction permitting the drawing of a horizontal or vertical line without representing it as a series of short lines. Both of these conditions do exist for plotter 3 and may exist for plotter 10. If either one of these two conditions is not true for the new plotter, then LINE should be used in place of AXISi.

Subroutine LINEi is always required. If the new plotter must be put into the line drawing mode before any lines can be drawn, LINE1, LINE2, LINE9 and LINE10 should be used as models. If the new plotter is an incremental plotter, LINE4 should be used as the model. Otherwise, use LINE3 as the model. In addition, LINE1, LINE2 and LINE3 represent lines as series of short lines; LINE9 and LINE10 use only one plot command to draw a line of any length.

MODIFICATIONS AND ADDITIONS TO NASTRAN

Subroutine TYPEi should be written if there is a typing capability on the new plotter. If the new plotter must be put into the typing mode before any lines can be drawn, TYPE1, TYPE2, TYPE9 and TYPE10 should be used as models. Otherwise, use TYPE3 as the model. All characters should be vertically oriented and miniature in size. Each of the model subroutines has three important symbols initialized in a DATA statement: LSTCHR, NCHR and CHAR. LSTCHR is the index of the last legitimate character in the CHAR94 table (see section 2.5) available on the new plotter. NCHR is the number of characters for which the character code on the new plotter differs from the corresponding character code in the CHAR94 table. CHAR is a list of NCHR pairs of indices into the CHAR94 table: $CHAR_{1,j}$ is the index value of the character code which will produce the wrong typed character on the plotter, and $CHAR_{2,j}$ is the index value of the character code which will produce the correct typed character. One further note: some plotters have both a "typewriter" and "single-character typing" mode. The existing TYPEi subroutines use only the "single-character typing" mode. It is strongly recommended that this policy be continued for all new plotters.

Subroutine WPLTi is almost entirely dependent upon the structure of the plot commands for the new plotter. The existing WPLTi subroutines will help the programmer only insofar as the overall logic is concerned. Once the plot command is constructed, it is written using subroutine SWRITE. Some plotters also require special information at the beginning and/or end of each record on the plot tape. If this is the case with the new plotter, WPLT3, WPLT4, WPLT9 and WPLT10 contain the logic necessary to recognize the beginning and/or end of a plot tape record. In addition, all the existing WPLTi subroutines, except for WPLT4, generate fixed-length plot commands. This results in a simpler subroutine and also adapts easier to the computer-independent characteristic of NASTRAN. It is therefore recommended that this policy be continued, if at all possible, for all new plotters.

Of the ten (10) subroutines to which changes and/or additions must be made, changes in six of the ten subroutines are dependent upon a) the internal index of the new plotter; and b) which of the four possible new subroutines have been written. These six (6) subroutines are: AXIS, LINE, SYMBOL, TIPE, TYPFLT and TYPINT. In each there are one or two computed-go-to statements based upon the internal index of the new plotter which will have to be enlarged. Then corresponding statements calling the new subroutines (AXISi, LINEi, or TYPEi) will have to be added. If an AXISi subroutine was not written for the new plotter, subroutine LINE should be called

PLOTTER OUTPUT

from subroutine AXIS instead of calling AXISi. If a TYPEi subroutine was not written for the new plotter, then subroutine DRWCHR should be called from subroutines SYMBØL, TIPE, TYPFLT and TYPINT instead of calling TYPEi.

Subroutine FNDPLT relates the external plotter and model names with its internal plotter and model indices. There is a table within this subroutine which reflects this relationship. The programmer need only append the external name of the new plotter and/or an additional model name to an existing plotter, together with the corresponding internal plotter and model indices. In addition, the value of the variable NPLTRS will have to be incremented by one whenever a new external plotter name is added.

Subroutine PLØTBD is a Block Data subprogram. Both the PLTDAT and SYMBLS tables (see section 2.5) must be expanded to reflect the addition of the new plotter. A new 20-word section must be appended to the PLTDAT table reflecting the physical characteristics of the new plotter. The values in this new section must correspond on a one-to-one basis with the values defined for the existing plotters. In addition, the buffer size defined for the new plotter must be a multiple of 60 characters if computer independency is to be preserved. A new 20-word section must be appended to the SYMBLS table. The first "NSYM" values in this new section are indices into the CHAR94 or CHRDRW tables (see section 2.5). These indices represent the special symbols used by the SYMBØL subroutine for the new plotter.

A new section must be added to subroutine SELCAM. The function of the new section varies depending upon the requirements of the new plotter. If the new plotter is a table plotter, a plot command should be generated to stop the plotter so that the operator can prepare for the next plot. If the new plotter is a microfilm plotter, the requested camera must be selected. Some plotters require a special "header" record at the beginning of each plot. If so, it must be generated in this subroutine. Again, the function of this subroutine varies depending upon the requirements of the new plotter, and as a result, the added section is very likely to be entirely plotter-dependent. In addition, there are two computed-go-to statements (based upon the internal index of the new plotter) which must be enlarged to reflect the new plotter.

The last subroutine to which changes and/or additions must be made is SKPFRM. Its function is to skip a variable number of frames on a microfilm plotter, or to skip over the current plot on a drum plotter. If the new plotter is neither a microfilm nor a drum plotter, no new section

is needed. In either case, there is a computed-go-to statement (based upon the internal index of the new plotter) which must be enlarged to reflect the new plotter. If the new plotter is a microfilm plotter, the commands necessary to skip the frame(s) must be added to the subroutine (e.g., plotters 3 and 9). If the new plotter is a drum plotter, then the commands necessary to skip over the current plot must be added to the subroutine (e.g., plotter 4). As with subroutine SELCAM, any added section is very likely going to be plotter-dependent.

Finally, an important word of caution to the programmer must be stated: if the computer and installation independence of the NASTRAN plotter software package is to be maintained, no existing software provided for his computer or installation ought to be used to accomplish the task of adding a new plotter to the NASTRAN plotter software package. This restriction prevents the direct usage of on-line plotters. NASTRAN does provide however for the indirect usage of on-line plotters as described in section 6.10.6.

6.10.2 Changes to the PLØT Module, the Structural Plotter

In order to add a new plotter to the NASTRAN structural plotter module, PLØT, only one subroutine need be altered: PLØPR. This subroutine generates messages to the plotter operator. Probably the only message which should be added is one to identify the plotter for which the structural plots are being generated. There is a repertoire of other messages in this subroutine which should be general enough to apply to any plotter type. If additional messages are needed, the programmer must remember that the plotter operator is not normally a "programmer type". Hence he will be more apt to respond to messages written in his language than to messages written in programming terminology. In addition, there are several computed-go-to statements based upon the internal index of the new plotter. These will have to be enlarged to reflect the new plotter.

6.10.3 Changes to the XYPLØT Module, the XY Plotter

No changes are required within the XYPLØT module when adding a new plotter. However, a minor change is required in subroutine IFPIXY of the IFP module if and only if the new plotter uses the NASTRAN BCD plot tape, PLT1. If this is the case, the new internal plotter number which is generated by subroutine FNDPLT of the NASTRAN plotter software should be added to the FØRTRAN logical IF statement directly after the following comment in subroutine IFPIXY:

PLOTTER OUTPUT

C

C IF MORE BCD PLOTTERS ARE ADDED EXPAND THE FOLLOWING TEST

C

6.10.4 Changes to the SEEMAT Module, the Matrix Plotter

No changes to the SEEMAT module are necessary when a new plotter is to be added since the plotter and model names are communicated to the module through the DMAP calling sequence (see section 4.74).

6.10.5 Use of the NASTRAN Plotter Software in a New Module

There exists in NASTRAN a self-contained computer-independent environment for creating plots on a large number of plotters. Currently three modules use this environment: 1) PLØT, the Structural Plotter; 2) XYPLØT, the XY Plotter; and 3) SEEMAT, the Matrix Plotter. In order for another module to generate plots in this same environment, it is essential that the module writer understand this environment.

The NASTRAN plotting environment can be understood most easily if viewed as being composed of five parts: 1) parameter definitions; 2) parameter initialization; 3) plot initiation; 4) plot creation; and 5) plot termination.

There is one DMAP parameter (passed through blank common) and two named common blocks in the parameter definition section of the NASTRAN plotting environment. The DMAP parameter is PLTNUM, the plot number of the last plot created, and is both an input and output integer value. The two named common blocks, with their symbolic contents, are as follows:

COMMON/XXPARAM/PBFSIZ,MEDIUM,BFRAMS,SKIP(4),XPAPSZ,YPAPSZ

where:

PBFSIZ = size of the plot tape buffer (integer)

MEDIUM = medium output request (integer, default value = 2)

= { 1 = film only
2 = paper only
3 = both

BFRAMS = number of blank frames (on film only) between plots (integer, default value = 1)

XPAPSZ = width of the paper (inches, for table plotters only) to be used (real, default value = 8.5 inches)

YPAPSZ = height of the paper (inches, for table plotters only) to be used (real, default value = 11.0 inches)

COMMON/PLTDAT/MØDEL,PLØTER,REGION(4),AXMAX,AYMAX,XEDGE,YEDGE,SKIP(10),PCNTIN,CNTCHX,CNTCHY,
SKIP(4),PLTYPE,PLTAPE,SKIP(2),FCNTIN

where:

MØDEL = model index of the plotter to be used (integer, default value = 1)
 PLØTER = plotter index of the plotter to be used (integer, default value = 3 - the SC 4020 microfilm plotter)
 REGION = the region of the plotting surface (x_{min} , y_{min} , x_{max} , y_{max}) in which a plot is being created (real)
 AXMAX = width of the plotting surface less any borders (real)
 AYMAX = height of the plotting surface less any borders (real)
 XEDGE = width of the border on the left and right side of the plotting surface (real)
 YEDGE = height of the border on the top and bottom of the plotting surface (real)
 PCNTIN = number of counts (plotter units) per inch of paper (real)
 CNTCHX = the width (includes horizontal spacing between characters) of each printed or drawn character (real)
 CNTCHY = the height (includes vertical spacing between characters) of each printed or drawn character (real)
 PLTYPE = plotter type (integer)
 = $\begin{cases} \underline{+1} & \text{microfilm plotter} \\ \underline{+2} & \text{table plotter} \\ \underline{+3} & \text{drum plotter} \end{cases}$
 = $\begin{cases} <0 & \text{if no typing capability exists on the plotter.} \\ >0 & \text{if typing capability does exist on the plotter.} \end{cases}$
 PLTAPE = GINØ file name of the plot tape (BCD)
 = $\begin{cases} \text{PLT1} & \text{for an even parity plot tape} \\ \text{PLT2} & \text{for an odd parity plot tape} \end{cases}$
 FCNTIN = number of counts (plotter units) per inch of film (real, FCNTIN = PCNTIN if the plotter is not a microfilm plotter)

The usage and initialization of each of the variables listed above are detailed in the following descriptions of the four remaining sections of the NASTRAN plotting environment (see section 2.5 for a more detailed description of /XXPARAM/ and /PLTDAT/).

The parameter initialization section of the NASTRAN plotting environment involves initializing all the variables listed above for the /XXPARM/ and /PLTDAT/ named common blocks, with the exception of MEDIUM and BFRAMS. First, the plotter and model indices (PLØTER, MØDEL) of the plotter to be used must be set by using subroutine FNDPLT:

```
CALL FNDPLT (PLTTER,MØDELN,PLTID,MØDID)
```

where:

PLTTER = plotter index of the plotter specified in PLTID and MØDID (integer, output)
 MØDELN = model index of the plotter specified in PLTID and MØDID (integer, output)
 PLTID(2) = 8 character name (4 characters per word, left-adjusted, with all remaining characters blank) of the plotter of interest (BCD, input).
 MØDID(2) = model identification of the plotter of interest. MØDID_i may either be numeric or BCD, depending upon the way in which various models are identified for the plotter of interest. If MØDID_i is BCD, it must be 4 characters, left-adjusted, with all remaining characters blank. (Integer and/or BCD, input and output).

See section 3.1 for a list of current plotter and model names.

Subroutine FNDPLT will attempt to match PLTID with an internal list of plotters available in the NASTRAN plotting environment. If no match is found, FNDPLT sets both PLTTER and MØDELN = 0 and immediately returns. Otherwise, FNDPLT will attempt to match MØDID with an internal list of models for the plotter of interest. If no match is found, a default model for the plotter is stored in MØDID. Then the corresponding plotter and model indices are stored in PLTTER and MØDELN. Generally, PLTID and MØDID will be provided by the NASTRAN user. It is suggested that the module writer set MØDID(1) and MØDID(2) equal to zero before inserting the user supplied values for MØDID. This provides the user with the capability of defaulting to a standard default model for his specified plotter by supplying no values for MØDID or by supplying a value only for MØDID(1).

Having called subroutine FNDPLT to determine the plotter and model indices, the module writer should then set PLØTER and MØDEL equal to PLTTER and MØDELN, respectively:

```
PLØTER = PLTTER
```

```
MODEL = MØDELN
```

PLOTTER OUTPUT

Next, the remainder of the /PLTDAT/ named common block must be initialized. Subroutine PLTSET must be used to do this. PLTSET sets all the variables in /PLTDAT/, except MØDEL and PLØTER, to values dependent upon the plotter index (PLØTER). In addition, if the plotter is a table plotter, AXMAX and AYMAX are also functions of the paper size (XPAPSZ, YPAPSZ). For this reason, if the current or default paper size is to be changed, it must be done before calling PLTSET. Finally REGION is initialized in PLTSET as follows:

```
REGION(1) = 0.  
REGION(2) = 0.  
REGION(3) = AXMAX  
REGION(4) = AYMAX
```

and the plot tape buffer size (PBFSIZ) in the /XXPARM/ named common block is initialized.

This completes the parameter initialization section of the NASTRAN plotting environment. The next step is plot initiation. Once this step is initiated, none of the parameters in either the /XXPARM/ or /PLTDAT/ named common blocks may be changed, except for MEDIUM, BFRAMS and REGION, without repeating the parameter initialization step.

In the plot initiation step, the module writer must first ensure that the plot tape (PLTAPE) is indeed a physical tape. This can be done by using logical function TAPBIT. If the function result is .FALSE., the plot tape is not a physical tape and hence no plotting should be attempted by the module:

```
IF (.NOT.TAPBIT(PLTAPE)) "no physical tape setup"
```

Having verified the existence of a physical plot tape, an array of PBFSIZ words must be provided for the NASTRAN plotting software as follows:

```
CALL SØPEN ($n,PLTAPE,BUFFER,PBFSIZ)
```

where:

n = FØRTRAN statement number to which SØPEN is to return if PLTAPE has not been correctly initialized.

BUFFER = an array of PBFSIZ full words.

Next, the plotter must be started. However, before doing this, the plot number (PLTNUM) should be incremented by one. In addition, this is the module writer's last opportunity to change the medium request (MEDIUM) and the number of blank frames between plots (BFRAMS). To start the plotter, the module writer must call subroutine STPLØT:

CALL STPLØT (PLTNUM)

Subroutine STPLØT will select the proper medium (if appropriate), generate an identification plot (if necessary), and insert the specified number of blank frames (on film only).

Having initiated a plot, the module writer must then create the plot. In the plot creation section of the NASTRAN plotting environment, there are seven (7) subroutines provided for various tasks: AXIS, LINE, PRINT, SYMBOL, TIPE, TYPFLT, TYPINT. An explanation of the calling sequence and purpose of each of these subroutines exists in section 3. It is essential that the module writer familiarize himself with the calling sequences and purposes of these subroutines. In addition, it is just as essential that he also understand a certain amount of the philosophy which exists in these subroutines as a class. What follows is an attempt to explain this philosophy, together with pertinent suggestions.

There are three operating modes defined in these seven subroutines: 1) axis mode; 2) straight line mode; and 3) typing mode. These three modes are totally independent of each other and are mutually self-exclusive. What this actually implies is that only one mode can be active at any one point in time. Subroutine AXIS operates in the axis mode; subroutine LINE operates in the straight line mode; and subroutines PRINT, SYMBOL, TIPE, TYPFLT and TYPINT all operate in the typing mode.

Each of these subroutines has a common argument, which is always the last argument in each of the calling sequences. This argument (ØPT) is used to initiate a mode (ØPT=-1), operate within a mode (ØPT=0), or terminate a logical subset of plot commands within a mode (ØPT=+1). Only while operating within a mode (ØPT=0) do any of the other arguments in a calling sequence have any meaning. For this reason, it is usually a good practice to use zeros for the other arguments when initiating a mode or when terminating a logical subset of plot commands.

It is strongly recommended that the set of all commands used to create a plot be grouped into logical subsets of commands, with each subset operating in only one of the three possible modes.

PLOTTER OUTPUT

This does not mean that all axes or all lines or all typing be included in one subset of commands. In many cases it is more logical to create several subsets of axis commands or straight line commands. The module writer must call one of the subroutines which operates within the same mode, with $\emptyset PT = -1$, so that the mode will be properly initiated. It is then recommended that, following such a subset of commands, the module writer again call one of the subroutines which operates within the same mode, with $\emptyset PT = +1$, to terminate the subset of plot commands. An example of these recommendations is as follows:

```
CALL TIPE (0,0,0,0,0,-1)
```

```
CALL PRINT (.....,0)
```

```
⋮
```

```
CALL TIPE (.....,0)
```

```
⋮
```

```
CALL SYMBOL (.....,0)
```

```
⋮
```

```
CALL TYPFLT (.....,0)
```

```
⋮
```

```
CALL TYPINT (.....,0)
```

```
⋮
```

```
CALL TIPE (0,0,0,0,0,+1)
```

Generally, the module writer should avoid constantly changing plot modes. This is suggested for two reasons. First, some plotters operate very inefficiently in a mode switching environment. Second, should an error exist in either the software or hardware, it might be necessary to dump the generated plot tape. Interpreting this dump would generally be no easy task. However, if the idea of creating subsets of commands was used in generating the plot tape, the task of locating the command(s) causing the problem(s) would be eased considerably in most cases.

All the subroutines used in creating a plot require that at least one of the arguments be the location of a point on the plotting surface. In each case, the point(s) must be specified as real

numbers already scaled to the plotter units. There is no general recommendation as to when this scaling should take place. In some cases, it would be more logical to perform all the scaling at once. In other cases, it would be more logical to perform the scaling on each subset of plot commands. While in other cases, it would be more logical to perform scaling only on an as-needed basis. The choice is left entirely to the discretion of the module writer. Since AXMAX and AYMAX in the /PLTDAT/ named common block define the width and height of the plotting surface in plotter units, and since the plotter origin can always be assumed to be in the lower left corner of the plotting surface, the required scaling ought to be a relatively easy task.

The lower left corner of the plotting surface is always at (0.,0.), while the upper right corner of the plotting surface is always at (AXMAX,AYMAX). These are also the default values of REGION (see the /PLTDAT/ named common block) as set by the PLTSET subroutine. The purpose of the REGION parameters is to define a rectangular plotting area, outside of which no plotting is to be attempted. For this reason, each of the seven subroutines used in creating a plot always compares the point(s) specified in the calling sequence with the REGION values. No portion of any line or axis will be drawn outside the corresponding rectangular plotting area. Nor will any typing be attempted outside this same area. The usage of these REGION parameters is left to the discretion of the module writer and the requirements of the module design. In most cases, the default values of REGION will not be altered by the module. One situation in which the REGION parameters will be altered is when the user has the capability of specifying that a plot is to be drawn only within a specific portion of the total plotting surface and that any part of the plot which appears outside this area is not to be drawn. In any case, if the REGION parameters are altered within the module, this can be done at any time on an as-necessary basis.

Having created the desired plot in the plot creation step, the only remaining task for the module writer is terminating the plot. This is a very simple task, accomplished by calling STPLØT:

CALL STPLØT (-1)

This subroutine, which was also used to initiate the plot, upon sensing a negative argument, will terminate the current subset of plot commands, skip to a new frame (if appropriate), and write an end-of-file mark on the plot tape (if necessary).

PLOTTER OUTPUT

If additional plots are to be created within the same module, the entire process just described must be repeated, starting with step 3 (plot initiation). If, however, a new plotter is specified for the succeeding plots, step 2 (parameter initialization) must also be repeated. If a new paper size is specified, subroutine PLTSET must be re-executed prior to repeating step 3.

This concludes the description of the NASTRAN plotting environment. There are two other sections in the NASTRAN Programmer's Manual which deal with this environment. It is suggested that the module writer read these sections also so that he may acquire more of an understanding of the NASTRAN plotting environment than this section affords him. These other sections are: Changes to the Plotter Software, section 6.10.1, and NASTRAN General Purpose Plotter, section 6.10.6.

6.10.6 NASTRAN General Purpose Plotter

One feature which the NASTRAN plotting software lacks is the capability of direct usage of the plotting equipment attached on-line to a computer. This is due not to special purpose programming, but rather to one of the basic characteristics of NASTRAN: computer independence. To access on-line plotters would not only make NASTRAN computer-dependent, but probably installation-dependent also. This installation dependency would result from the necessity of using special subroutines provided by the computer installation to access the on-line plotter, with no guarantee that subroutines having the same name and calling sequences would be available at any other computer installation. Even so, there would almost certainly occur a subroutine naming conflict, due to the great number of subroutines in NASTRAN.

An effort is made in NASTRAN to partially overcome this deficiency. In general, NASTRAN will produce a plot tape which can be used directly by any one of several off-line plotters. In addition, NASTRAN can be directed by the user to produce a so-called "General Purpose Plotter" tape. Another program, completely external to NASTRAN, would then have to exist, its function being to translate this "plot" tape for the on-line plotter so that it will produce the plots intended by NASTRAN. This implies that in order to produce a NASTRAN plot, two programs must be run: first, NASTRAN itself; and then the external translator program.

The purpose of this section is to explain the characteristics and construction of the "NASTRAN General Purpose Plotter" tape, so that a programmer will be able to write a program to translate this "plot" tape for the on-line plotter. Understanding the overall logic used by the NASTRAN plotter software package in producing a plot tape will simplify the task of writing this translator program. It is therefore recommended that the programmer familiarize himself not only with this section of the Programmer's Manual, but also with sections 6.10.1 and 6.10.5, which describe the technique of adding a new plotter to the NASTRAN plotting software package.

The "NASTRAN General Purpose Plotter" tape is composed of a simple set of elementary plot operations, which can easily be deciphered by a FØRTRAN program on any digital computer. As each operation is deciphered, the translator program should direct the on-line plotter to appropriate action. This would normally be done by using the installation software to interface between the translate program and the on-line plotter. With the existence of this external translator program, NASTRAN would then have the capability of indirectly referencing the corresponding on-line

PLOTTER OUTPUT

plotter. A by-produce of this environment is the implied capability of indirectly accessing any plotter, whether on-line or off-line, assuming the appropriate external translator programs are written.

The "NASTRAN General Purpose Plotter" tape is a seven-track, odd parity, fixed-length record tape. An end-of-file mark follows the last plot only. Each record is composed of 3000 six-bit-unsigned integers (750 words on an IBM S/360, 500 words on Univac 1108, 300 words on a CDC 6600) and is composed of 100 plot commands, each being composed of 30 six-bit unsigned integers (15 half-words on an IBM S/360, 5 words on a Univac 1108, 3 words on a CDC 6600). Not all plot commands will have useful information in all 30 six-bit integers. Some commands use only two of the 30 six-bit integers, while others use 22. The general format of each command is as follows:

$$P C R_4 R_3 R_2 R_1 R_0 S_4 S_3 S_2 S_1 S_0 T_4 T_3 T_2 T_1 T_0 U_4 U_3 U_2 U_1 U_0 00000000 ,$$

where:

- P = plot command,
- C = control index,
- R_i = decimal digit of an integer called R,
- S_i = decimal digit of an integer called S,
- T_i = decimal digit of an integer called T,
- U_i = decimal digit of an integer called U,
- 0 = zero.

The plot command is a six-bit integer, any one of seven (7) possible plot commands, as follows:

- 0 = no operation,
- 1 = start new plot,
- 2 = select camera,
- 3 = skip to a new frame,
- 4 = type a character (may also = 14),
- 5 = draw a line (may also = 15),
- 6 = draw an axis (may also = 16).

The control index is also a six-bit integer. It may be a pen number, a line density, a camera number, or a pointer into a list of characters and symbols. The four integer values (R,S,T,U) specified in a command must be reconstructed by the external translator program. Each integer value is represented in the command as follows:

$$d_4 d_3 d_2 d_1 d_0$$

where the original integer value is given by:

$$d_4 10^4 + d_3 10^3 + d_2 10^2 + d_1 10^1 + d_0 10^0$$

The significance of each of the four integer values (R,S,T,U) may vary from one plot command to another.

The no-operation (0) command is simply a padding for plot records which may otherwise have been less than 300 characters long. All 30 characters of this command will be zero.

The start-new-plot (1) command will always be the first command introducing each new plot. The first integer (R) will be the plot number. The second and third integers (S and T) are the maximum x and y values specified in any other command for this plot. The minimum x and y values are always zero and are therefore not specified in the start new plot command. If necessary, the translator program can use these maximum x and y values to scale subsequent integer values so that the plot will not exceed the limits of the plotting surface. The plot number is included because some plotters require the plot number as part of the first command for each new plot. In addition, if the receiving plotter is a table plotter, the translator program should issue a command to the plotter which will stop it so that the plotter operator can change the paper. If the plotter is a drum plotter, the translator program must skip a sufficient amount of paper to insure that the previous plot will not be over-plotted. And if the receiving plotter is a micro-film plotter, nothing else need be done.

The select-camera (2) command uses only the control index (C). The remaining 28 characters are always zeros. This command is meaningful only on a microfilm plotter having both film and hardcopy output. The control index is the camera or medium request number: 1 = film only; 2 = hardcopy (paper) only; and 3 = both. Upon receiving this command, the translator program should issue a command to the receiving plotter selecting the requested camera or output medium,

then this command should be ignored.

The skip-to-a-new-frame (3) command also uses only the control index. The remaining 28 characters are always zeros. This command is meaningful only on a microfilm plotter. The control index is the camera or output medium request number: 1 = film only; 2 = hardcopy (paper) only; and 3 = both. The appropriate camera will have already been selected in a previous select-camera command. The only reason the camera number is included in this command is because some microfilm plotters require the camera or output medium to be specified in both a select camera and skip frame command. Upon receiving this command, the translator program should issue a command to the receiving plotter to skip to a new frame. If the receiving plotter is not a microfilm plotter, then this command should be ignored. Note: at least one skip-to-a-new-frame command will appear after each start-new-plot command and before the next start-new-plot command.

The type-character (4), draw-line (5), and draw-axis (6) commands will always occur in sets, i.e., a set of type-character commands, a set of draw-line commands, a set of draw-axis commands. There may be more than one set of each type of command, but within a set the commands will all be of an identical type. This is done because on some plotters it is very inefficient to frequently change modes (e.g., typing mode, line drawing mode) of operation. The plot command of the first command in a set will always = 10 + the basic plot command value, i.e., type-character = 14; draw-line = 15; and draw-axis = 16. In all subsequent plot commands in the set, the plot command value will always equal the basic plot command value.

For a type-character command, the control index is a pointer into a specific list of characters and special symbols. The list of characters to which the pointer applies is Section I of the CHAR94 table (see section 2.5), with the following exceptions: 48 = dot, 49 = circle, 50 = square, 51 = diamond, 52 = triangle (point up). The first two integer values (R and S) in the plot command represent the x and y coordinates of the point on the plotting surface at which the center of the character or symbol should be typed. The remaining 18 characters of the command are always zeros. Upon receipt of a type-character command, the translator program should issue a command to the receiving plotter to type the requested character or special symbol at the specified point. Of course, there is no guarantee that all the possible characters and special symbols can be typed by the receiving plotter. If any character or special symbol cannot be typed by the receiving plotter, the translator program will then have to make a substitution or not type the character at all.

For a draw-line command, the control index is either a pen number (for table and drum plotters) or a line density (for microfilm plotters). If the receiving plotter is a microfilm plotter, it is recommended that the translator program simply draw the line as many times as is indicated by the line density value, rather than using any special density settings available on the plotter hardware. The first two integer values (R and S) represent the x and y coordinates of the starting point of the line. The next two integer values (T and U) represent the x and y coordinates of the ending point of the line. The last 8 characters of the command are always zeros. Upon receipt of this command, the translator program should issue a command to the receiving plotter to draw the line. Note: some plotters require that a line be broken into a series of short lines. If this is the case on the receiving plotter, the translator program will have to accomplish this task unless the installation software makes provision for this automatically.

The draw-axis command is identical to the draw-line command. The only difference is in the orientation of the drawn line. The line drawn by a draw-axis command will always be either horizontal or vertical. For most plotters, the translator program will handle this command just like a draw-line command. However, some plotters which would ordinarily require that lines be broken into a series of short lines, may have a special command available to draw a horizontal or vertical line of any length. For these few plotters only will this command have any special significance in the translator program. If such is the situation, the translator program, upon receipt of this command, should issue a command to the receiving plotter to draw the axis. Otherwise, the translator program should simply issue a command to the receiving plotter to draw a line representing the axis.

ADDITION OF A NEW LINK

6.11 ADDITION OF A NEW LINK

Links can be added to the NASTRAN system but the Executive System changes to increase the link limit will be required as described in Section 6.11.5. Links are numbered consecutively, and this should be maintained.

It is assumed that the reader is familiar with the other alternatives for adding new material to the system (Sections 6.2 through 6.10), and that a new link is normally needed only if the addition of new modules to a present link makes that link non-executable because of an excess of overlay structure or decks. In this case, the programmer generates a new link through the following steps (assuming the link limit is not exceeded):

1. Decide what modules to include in the new link.
2. Add any new modules to the MPL Executive Table (see Section 2.4.2.2) in deck XMPLBD.
3. Generate a new Link Specification Table and a new link driver.
4. Subsys (create an absolute element of) the new link.

6.11.1 Modules to Include

The following entry points of Executive modules must be included in each new link: XCHK, XCEI, XSAVE, XPURGE, XEQUIV, XSFA and QPARAM. The following non-root segment subroutines must be included in each new link: MSGWRT, USRMSG, BTSTRP, ENDSYS and XEØT. Charts depicting the overlay structures for all these routines can be found in Section 5. The addition or use of other modules is at the discretion of the NASTRAN systems programmer. Any existing NASTRAN module can be included in the new link if it is so desired.

6.11.2 Addition of New Modules

All new modules must be added to the MPL Executive table (see Section 2.4.2.2). Once this is done, the new MPL needs to be added to the present NASTRAN system before proceeding to generate a new Link Specification Table and a new link driver. The new modules need not be present in the system, but their entry points in the MPL must be in the system.

MODIFICATIONS AND ADDITIONS TO NASTRAN

6.11.3 Generation of a New Link Specification Table and a New Link Driver

The addition of a new link requires that: a) the Link Specification Table, LNKSPC, (see Section 2.4.2.7) be updated; and b) a new XSEMI deck (see Section 3.3.7), which will be the main program for the new link, be generated.

The FORTRAN code, in punched card form, necessary to accomplish both of the above tasks can be produced automatically in a NASTRAN run using a LNKSPC update deck as indicated below.

6.11.3.1 Link Specification Table Update

To update the Link Specification Table the user must insert a LNKSPC update deck after the Bulk Data Deck in a NASTRAN run. The processing of a LNKSPC update deck is initiated by logical "sense switches" on a DIAG card in the Executive Control Deck (see explanation below); hence no special header card is required in the LNKSPC update deck. The format of each card of the LNKSPC update deck is as follows:

$$\alpha \beta L_1, \dots, L_n$$

where

α = Module DMAP name (1 to 8 characters)

β = Entry point name (1 to 6 characters)
"NONE" (6 characters) if there is no entry point name

L_1, \dots, L_n = zero or more integers, specifying all links the module resides in.

The α , β and L_i fields must be separated by at least one blank or one comma. An example of a card in a LNKSPC update deck is

```
CHKPNT    XCHK    1,2,3,4,5,6,7,8,9,10,11,12,13,14
```

If the Executive module CHKPNT with entry point XCHK is in links 1 to 13, the above card will add it to a link 14. Be sure to include all the other Executive modules in any new link. Cards in this format are repeated until all the modules in the new link have been named. The end of the LNKSPC update is specified by the following card:

```
ENDDATA
```

ADDITION OF A NEW LINK

The following logical sense switches, set by a DIAG card in the Executive Control Deck, must be used to process the LNKSPC update deck:

1. Logical sense switch 29 allows the LNKSPC update deck to be processed.
2. Logical sense switch 28 causes the new Link Specification Table to be punched.*
3. Logical sense switch 31 causes the new Link Specification Table to be printed.

The following DIAG card within the Executive Control Deck

DIAG 28,29,31

accomplishes this result.

*This code is for the Block Data subprogram XBSBD, and may be compiled under that name as punched, or altered into the present deck to retain the comments presently in that deck. The latter procedure is recommended.

6.11.3.2 Generation New Link Driver Decks

The FØRTRAN code necessary to make a new link driver deck can be produced by setting logical sense switch 30 and the logical sense switches corresponding to the desired link(s). For example the Executive Control Deck card:

DIAG 30,2,14

will punch the code for XSEM02 and XSEM14. The FØRTRAN code produced, when altered into the deck "XSEMX" (see Section 3.3.8), will produce the desired XSEMi routine. XSEMX is the model for the link drivers.

This driver code may be produced in the same run as the code for the Link Specification Table, or this code may be produced by adding the new XBSBD deck to the user's system and then setting the appropriate sense switches on a subsequent run.

If logical sense switch 30 is set, the program will automatically terminate after satisfying all sense switch requests to punch XSEMi's. In essence, the structural problem submitted is only a dummy needed to start NASTRAN. The user is cautioned to check his DIAG card(s) carefully when logical sense switch 30 is set to be sure only the sense switches corresponding to the desired links are set.

6.11.4 Subsys the New Link

The subsys deck necessary to add a new link will vary with the computing machine (see Section 5). The safest way for the NASTRAN systems programmer to develop the subsys deck for the new link is to copy the overlay structure of the necessary routines, which are listed in Section 6.11.1, and then add the new functional module(s) to this base. The mechanisms for switching from one link to another already exists in NASTRAN up to 15 links, so that, once the subsys of the new link has been accomplished, NASTRAN will be able to assimilate it.

6.11.5 Increasing the Link Limit

To increase the link limit beyond 15, the following Executive System changes must be made:

1. Increase the size of the NAME array in the /SEM/ common block to the desired link limit. Add the additional link names (e.g., NS16, NS17, etc.) to the NAME array via the DATA statement. The /SEM/ common block is defined in the System Block Data subprogram SEMDBD.
2. Items MAXLNK, DRVRNM, and LNKEDT, defined in subroutine XGPIBS of Executive Preface module XGPI (see Section 4.7), must be updated. Increase the value for MAXLNK in the /XLINK/ common block to the desired link limit. Add the additional XSEMI names (e.g., XSEM16, XSEM17, etc.) to the local DRVRNM array via the DATA statement. Add the additional link numbers (e.g., 16, 17, etc.) to the local LNKEDT array via the DATA statement.

6.11.6 Adding a New Link to the CDC Version of NASTRAN

The only change required for the CDC version of NASTRAN is to supply the subsys for the new link to the linkage editor and linkedit all links including the new link.

6.11.7 Adding a New Link to the IBM Version of NASTRAN

The only change required for the IBM Version of NASTRAN is to linkedit the new load module for the new link into the partitioned data set containing the other NASTRAN load modules. The subsys for a new link should be patterned after existing links and must include subroutines LINKNSXX and EJDUM2. A CHANGE card for LINKNSXX to reference the new link driver subroutine (e.g., CHANGE XSEMXX (XSEM16)) is required.

ADDITION OF A NEW LINK

6.11.8 Adding a New Link to the UNIVAC Version of NASTRAN

The following steps are required to add a new link to the UNIVAC version of NASTRAN:

1. Update machine dependent subroutine SEARCH to include checks for the added link.
2. Update machine dependent subroutine SETC to include the name of the new link in the array LINKNM.
3. Add an element to the file containing the NASTRAN absolute elements with the same name as the new link number. This element will contain only one card and that will be a @XQT for the new link.
4. Map the new link into the file containing the NASTRAN absolute elements.

6.12 WRITING A NEW MODULE

The purpose of this section is to draw together material presented throughout the manual from the point of view of the NASTRAN applications programmer (i.e., the programmer assigned to add a new capability to NASTRAN). This section is provided as a guide for the general module writer, and will conclude with a specific example of a simple module.

6.12.1 Summary of NASTRAN Coding Conventions and Terminology

The new NASTRAN applications programmer is typically overwhelmed by the vastness of the system into which he will inject his modifications and additions. In this section, a review of commonly used terminology and coding conventions will be given in an attempt to assist the new programmer and provide a review for the programmer who infrequently works with the system.

A module in NASTRAN is a collection of subprograms which performs a logical set of data processing tasks. A module is executed by the user by writing and executing a DMAP instruction. Modules communicate with other modules and with the NASTRAN Executive System (EXEC) only through:

1. Data Blocks (Tables or Matrices existing as collections of data on a physical storage device)
 - a. Input Data Blocks are referenced by the GINØ file reference numbers 101,102,..., corresponding to the position of the data block in the DMAP call statement.
 - b. Output Data Blocks are referenced by the GINØ file reference numbers 201,202,..., corresponding to the position of the data block in the DMAP call statement.
 - c. Scratch Data Blocks are referenced by the GINØ file reference numbers 301,302,..., corresponding to any arbitrary order the module writer desires.
 - d. Note that there are no data blocks which are used for both input and output except the scratch data blocks which exist only internally within the module. An exception is present when a output data block is APPENDED.
2. Parameters (single values)
 - a. Parameters may be input, output or both as desired by the module programmer.
 - b. Each parameter defined for the module has a type and a corresponding number of computer words as follows:

MODIFICATIONS AND ADDITIONS TO NASTRAN

	<u>Type</u>	<u>Example</u>	<u>No. Words</u>
(1)	Integer	-3	1
(2)	Real	2.9	1
(3)	BCD	BX2	2
(4)	Double Precision	-3.2D0	2
(5)	Complex	(1.3, -4.9)	2
(6)	Complex Double Precision	(6.2D0,0.0D0)	4

c. Parameters are stored in the /BLANK/ labelled COMMON block in the order defined by the DMAP call statement.

3. Executive System Common Blocks

Several System Common Blocks are accessible to the module programmer. These include parametric values used by various utility routines, FORTRAN unit definitions, the GINØ buffer length value, and numerous communication areas for various utility routines such as the matrix packing routines. A description of these common blocks may be found in Section 2.

4. Executive System Utility and Matrix Operation Routines

A large number of routines are directly callable by the module programmer. These vary from elementary bit manipulation routines to extremely involved matrix operation routines. Routines which are usable by module programmers are described in Section 3.

The characteristics or properties of all modules are prescribed by the Module Properties List (MPL), a table used by the DMAP compiler XGPI and defined by the Block Data program XMPLBD. The number of input data blocks, output data blocks, scratch data blocks, as well as the number and type of the parameters is defined by the MPL and must be adhered to by both the user when using the module and by the module writer. The properties of a module can be changed only by recompiling XMPLBD.

The subprograms of a module may consist of SUBROUTINE subprograms, FUNCTION subprograms and/or unnamed BLOCK DATA subprograms as desired by the programmer. The main subprogram for the module, however, must be a SUBROUTINE subprogram without arguments. The name of this subprogram is prescribed by the Link Specification Table which is defined by the BLOCK DATA program XBSBD.

WRITING A NEW MODULE

Labeled common blocks may be defined for the module as desired for intra-module communication so long as unique names are chosen. Similarly, the names of all new subprograms must be unique. It is extremely important to remember to close all open GINØ files before executing a RETURN to the Executive System.

The basic GINØ file element is the logical record, which contains an arbitrary number of words of data. A most important non-FØRTRAN feature in NASTRAN is the ability to read and write part of a logical record. An interesting and useful application is the "blast read/write" illustrated below. Let NW be the number of words of working core (open core less GINØ buffers and any other pre-assigned areas) available to the module starting at X(K). If we wish to copy the contents of GINØ file F1 onto GINØ file F2, the following represents the most efficient way since the data is handled in as large as pieces as possible. Assume that both files are open and rewound.

```
10  CALL READ($30,$20,F1,X(K),NW,0,M)
    CALL WRITE(F2,X(K),NW,0)
    GØ TØ 10
20  CALL WRITE(F2,X(K),M,1)
    GØ TØ 10
30  CALL CLØSE(F2,1)
    CALL CLØSE(F1,1)
```

The DMAP name of a data block is not known a priori to the programmer since it is defined at execution time by the user in his DMAP sequence. If the programmer desires to obtain the name of the data block he is processing, he may do so via

```
CALL FNAME(F,NAM)
```

where F is the GINØ file reference identification number (101,203, etc.), and NAM is a two word array into which the name will be stored.

6.12.2 Module Design

Before a module can be written, it must be designed. While the general process of design cannot be formalized, certain NASTRAN rules and conventions can be discussed and illustrated.

MODIFICATIONS AND ADDITIONS TO NASTRAN

A module can obtain input data in three ways: 1) through any of a fixed number of input data blocks, 2) via parameters maintained by the EXEC, and 3) from system common blocks. Thus, the first items to determine are the input data blocks required, their number, and the parameters. A module can only create printed output, output data blocks, or parameters. It may not update system tables. The second item to determine is the number of output data blocks. Only one matrix can be written per output data block, and the format is prescribed. Table data blocks, on the other hand, may be formatted as the module designer pleases. A module may require temporary storage (other than central memory), and hence require scratch files. (These are often required by NASTRAN utility subroutines, such as MPYAD, etc.)

A module can be completely described by the number of inputs, the number of outputs, the number of scratch files, and an ordered list of parameters and their types (integer, real, BCD, double precision, complex single precision or complex double precision). The total number of files required should be less than 26. To be used, a module must be scheduled into a DMAP sequence along with other modules. A primary module design consideration must be the DMAP sequence in which it will function. Items to be considered include: 1) Are all input data blocks available prior to the running of this module? 2) Are the parameter types consistent? 3) Do the formats of the output data blocks agree with the input formats for the modules which will process them?

While the flow of data within a module can be as desired, a few guidelines are given below:

1. Try to use existing utility routines. In other words, try to express operations to be performed as tasks which can be solved by existing subroutines. For this purpose the utility routines of Section 3 can be classified as follows:

Processing Tables

READ	OPEN
WRITE	CLOSE
FWDREC	FNAME
BCKREC	SORT
REWIND	BISCH
EOF	GOPEN
RDTRL	FREAD
WRTTRL	

WRITING A NEW MODULE

Printing Output

CØNMSG	MATDUM
SSWCH	TABPRT
PEXIT	DMPFIL
PAGE	BUG
MESSAGE	EJECT

In-Core Matrix Operations

GMMATD	INVERS
GMMATS	PRETRD
INVERD	PRETRS

Processing Data Cards

PRELØC	XRCARD
PRETAB	PREMAT

Processing Matrices (General)

ØPEN	GØPEN
CLØSE	FREAD
FWDREC	BLDPK
RDTRL	PACK
WRTTRL	INTPK
FNAME	UNPACK

Processing Matrices (Operations)

SSG2A	SSG3A
SDR1B	SØLVER
UPART	FACTØR
SSG2B	TRANP1
SSG2C	

MODIFICATIONS AND ADDITIONS TO NASTRAN

Plotting

AXIS	STPLØT
SKPFRM	SYMBOL
SELCAM	TIPE
IDPLØT	TYPFLT
LINE	DRWCHR
PRINT	PLTSET
SØPEN	FNDPLT
SCLØSE	

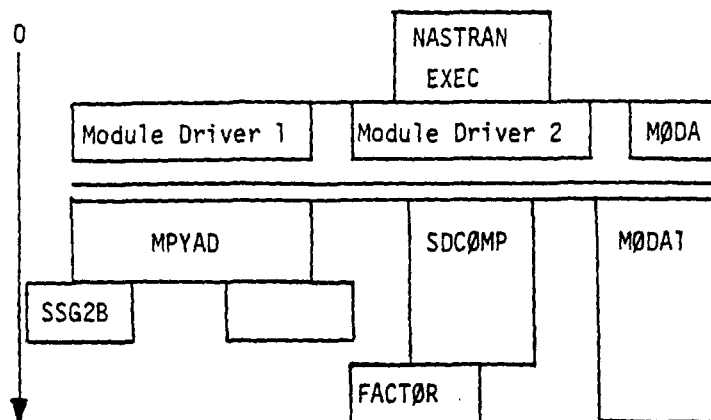
2. Try to limit material held in the central memory to as few items as possible. Thus, matrices should be processed a column at a time, if possible, and tables handled by logical records.
3. Stay within the existing overlay structure. In NASTRAN, certain types of subroutines are grouped together. The existing groupings can be determined by examining the overlay maps in Section 5. In particular, each major matrix operation is grouped by itself. This leads to the concept of a module driver and several subroutines which each call only one major matrix routine. For example, assume a module called MØDA wishes to read data cards, assemble a matrix, multiply this matrix by an input matrix, and decompose the result. It should be organized as follows:

Matrix Driver:

```
SUBROUTINE MØDA
CALL MØDA1(      ) --- Outputs new matrix.
CALL SSG2B(      ) --- Multiplies by input matrix.
CALL FACTØR(     ) --- Decomposes matrix.
RETURN          --- Back to EXEC.
END
```

WRITING A NEW MODULE

The overlay environment might be as follows:



Note that all matrix routines and all major areas of code are placed in core after all module drivers; therefore, each module driver should be as short as possible. Note that no data can be left in central memory between each part of the module; i.e., all data must be transferred to scratch files, thus freeing a maximum of central memory for each major matrix operation. Note also that the names of each user-generated subroutine should be related to the module name, as indicated in the example.

6.12.3 Implementing the New Module

Actual implementation of the new module can be done in two ways. The simplest way is to pick an existing dummy module (as documented in Section 5 of the User's Manual) which contains at least as many inputs, outputs, scratches, and parameters as needed for the new module. Failing this, the procedures described in Sections 6.11.2 and 6.11.3 must be followed. The new decks must be added to the existing overlay structure through the overlay control language of each particular machine.

The NASTRAN EXEC will (at the proper moment) call the specified entry point to the module. Note that no arguments are allowed. Thus, the new module must start as SUBROUTINE MØDA, if MØDA is its entry point. The module must conclude with a RETURN statement. Parameters will be placed in the /BLANK/ labelled COMMON block in the order specified in the DMAP calling sequence. Assume the module has three parameters, a BCD value, an integer, and a complex single precision number. They could be referenced as

```
COMMON /BLANK/ BCD(2),INTGR,CØMPLX(2)
```

MODIFICATIONS AND ADDITIONS TO NASTRAN

The parameter values may be changed by the module at will, and the /BLANK/ COMMON block used as the module writer pleases. (Note that /BLANK/ is usually in the root segment, and should be held to 300 words or less.)

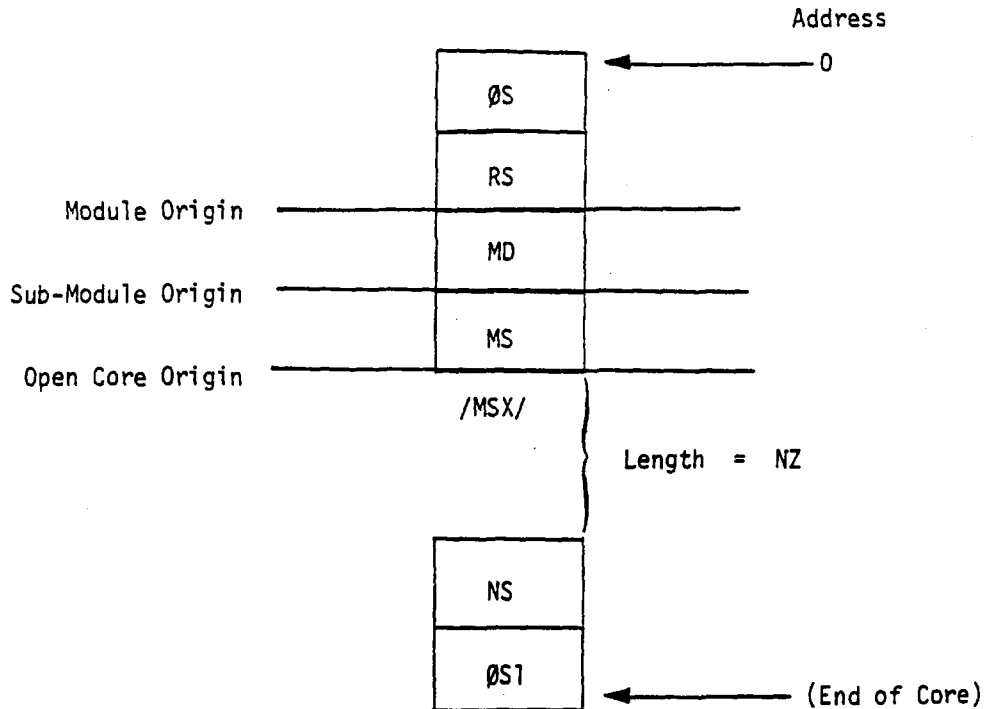
6.12.4 Coding a Module Subroutine

The remaining tasks should now be broken into coding individual module subroutines. (It may be possible to code a module which only calls existing subroutines. SCE1 is such a module.) Module subroutines are normal FORTRAN subprograms written in NASTRAN's limited FORTRAN subset. Note that bit operations have been added to NASTRAN FORTRAN via RSHIFT, LSHIFT, ANDF, ORF, etc. Two major differences are noted by experienced FORTRAN programmers, open core and I/O management. These are treated in subsequent sections.

6.12.4.1 Open Core Coding

A module subroutine can, of course, have normal FORTRAN arrays and COMMON blocks as it pleases, but many items in NASTRAN, such as the size of a matrix or the length of a table, are open ended or variable from run to run. These items must be held in a variable length storage space. Many FORTRAN programs simply DIMENSION an array as large as possible and work within this array. NASTRAN chooses to view this array as starting in a particular COMMON block, the name of which is assigned by each module. This COMMON block is positioned at the first available location within the existing overlay (usually immediately after the particular module subroutine). The module subroutine obtains the length of the array in this COMMON block by calling upon a NASTRAN EXECUTIVE function KORSZ. The module writer may organize this array as he pleases, as long as he stays within its specified length. FORTRAN programmers coding subroutines on the UNIVAC 1108 should review Section 5.4.4 for possible addressing difficulties. Consider the following diagram, which shows a picture of the main memory of a hypothetical computer.

WRITING A NEW MODULE



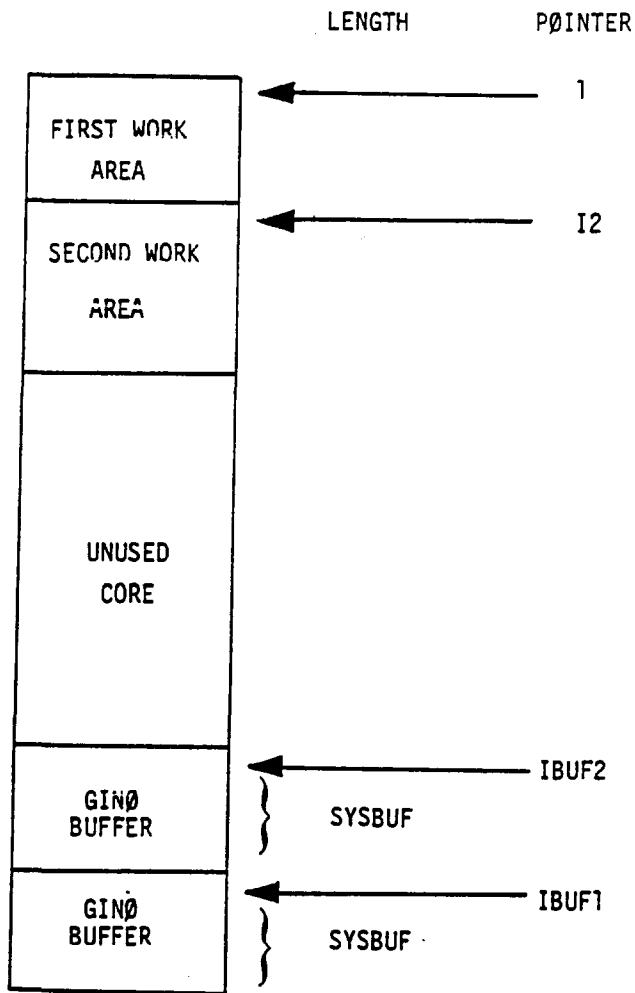
OS represents the area reserved for the resident Operating System, RS represents the area reserved for the Root Segment of NASTRAN, MD represents the area reserved for the Module Driver, and MS represents the area reserved for the Module Subroutine which is currently executing. /MSX/ is the labelled ~~COMMON~~ block used by MS to delimit the beginning of the variable array (open core). NS represents the area reserved for the NASTRAN EXEC, and OS1 represents the remainder of the core controlled by the operating system. The words of core from /MSX/ to the beginning of NS can be used by module subroutine MS. There are NZ of them.

A typical module subroutine determines the length of open core as follows:

```
COMMON /MSX/IZ(1)
NZ = KORSZ(IZ)
```

MODIFICATIONS AND ADDITIONS TO NASTRAN

This subroutine might organize the open core as shown below:



Such diagrams are indispensable to module programmers. The programmer would locate items such as the first GINØ buffer by $IZ(IBUF1)$. Such locations can be computed by

$$IBUF1 = NZ - SYSBUF + 1$$

$$IBUF2 = IBUF1 - SYSBUF$$

The proper length of open core should be checked before any use is made by a statement such as

$IF(2*NRØW+2*SYSBUF .LT. NZ) CALL MESSAGE (-8, 0, NAME)$

WRITING A NEW MODULE

6.12.4.2 I/O Management

Data blocks are selected via their GINØ file names. The input data blocks have positional numbers beginning with 101. Thus, data for the fourth input data block can be obtained by referencing GINØ file number 104. Outputs have positional numbers beginning with 201; scratches with 301. A module must close all of its open files before returning to EXEC.

The module writer must supply a buffer to GINØ for each file as he uses it. Supplying this buffer is referred to as opening the file. This buffer should exist in the module's open core. During the time the file is open, this area belongs to GINØ. Do not store in this area. The area may be reused after the GINØ file is closed. The length of this area is given by the value of the first word in the /SYSTEM/ COMMON block.

A typical module subroutine might proceed as follows:

```
SUBROUTINE MØDA1(INPUT)
  INTEGER SYSBUF
  COMMON / SYSTEM/SYSBUF,NØUT
  COMMON / MSX/IZ(1)
  NZ = KØRSZ(IZ)
  IBUF1 = NZ - SYSBUF+1
  IF(NZ .LT. SYSBUF) CALL MESSAGE(-8 . . .)
  CALL GØPEN(INPUT,IZ(IBUF1),0)
  .
  .
  .
  Process file
  .
  .
  WRITE(NØUT, ) -----
  .
  .
  CALL CLØSE(INPUT,1)
  RETURN
  END
```

MODIFICATIONS AND ADDITIONS TO NASTRAN

Printed output should be written onto the FORTRAN unit given by the value of the second word of /SYSTEM/.

All NASTRAN data blocks have a header record, which can be supplied by GOPEN or FNAME, and a trailer, which can be supplied by RDTRL or WRTTRL. Nonzero trailers must be written for all output data blocks and should be written on all scratch data blocks. These trailers are particularly important when processing matrices.

6.12.5 Sample Module Coding

This section contains the entire FORTRAN code for a new NASTRAN module. The module is to normalize a matrix to a maximum magnitude of 1.0 in each column. The module also is to output the number of rows and columns of the matrix as output parameters.

```
MODULE NAME = NØRM
Entry Point: NØRMM
Inputs:      1
Outputs:     1
Scratches:   0
Parameters:  2 - integer - output
DMAP Calling Sequence:
      NØRM  ANYMAT / NØRMMAT / V,N,NCØL / V,N,NRØW $
      SAVE  NCØL,NRØW $
```

The module is written as a single subroutine (rather than as a module driver/subroutine combination). Note the handling of multiple precision and complex values in open core and the liberal use of comments.

WRITING A NEW MODULE

SUBROUTINE NORMM

```

C
C   THIS IS THE DMAP MODULE NORM WHOSE CALL INSTRUCTION IS
C
C   NORM IN / OUT / V,N,NCOL / V,N,NROW $
C
C   NUMBER OF INPUT DATA BLOCKS      1
C   NUMBER OF OUTPUT DATA BLOCKS     1
C   NUMBER OF SCRATCH DATA BLOCKS    0
C   NUMBER OF INPUT PARAMETERS        0
C   NUMBER OF OUTPUT PARAMETERS       2--INTEGER
C
C   NORM WILL NORMALIZE EACH COLUMN OF AN INPUT MATRIX
C
C   INTEGER SYSBUF,MCH(7),NAME(2)
C   REAL Z(1)
C   DOUBLE PRECISION DZ(1),DMAX,CDZ(2,1),COMAX
C   COMPLEX CZ(1)
C   DIMENSION NW(4)
C   COMMON /NORMX / IZ(4)
C   COMMON /ELANK / NCOL,NROW
C   COMMON /SYSTEM/ SYSBUF,NDUT
C   COMMON /UNPAKX/ ITC,II,JJ,INCR
C   COMMON /PACKX / ITA,ITB,III,JJI,INCRI
C   EQUIVALENCE (IZ(1),Z(1),DZ(1),CZ(1),CDZ(1,1))
C   DATA INPUT/101/,IOUT/201/,NAME/4HNDP4,NM
C   DATA NW/1,2,2,4/
C
C   NORM WILL USE OPEN CORE AT IZ
C
C
C

```

MODIFICATIONS AND ADDITIONS TO NASTRAN

```

C      DETERMINE IF INPUT MATRIX IS PRESENT
C
      MCB(1) = INPUT
      CALL RDTRE(MCB)
      IF(MCB(1).LE.0) GO TO 500
C
C      OUTPUT PARAMETERS
C
      NCOL = MCB(2)
      NROW = MCB(3)
      WRITE(IOUT,10)NCOL,NROW
10  FORMAT(36H0*** USER INFORMATION MESSAGE ****, ,7HNCOL = ,I10,2H, ,
      *      7HNROW = ,I10)
C
C      ALLOCATE OPEN CORE
C
      NZ = KORSZ(IZ)
      IBUF1= NZ-SYSBUF+1
      IBUF2= IBUF1-SYSBUF
C
C      CHECK FOR SUFFICIENT CORE TO HOLD AN UNPACKED
C      COLUMN OF THE INPUT MATRIX AND TWO GINO BUFFERS
C
      KW = MCB(5)
      IF(NROW*NW(KW)+2*SYSBUF.GT.NZ)CALL MESSAGE(-8,0,NAME)
C
C      OPEN INPUT AND OUTPUT MATRICES
C
      CALL GOPEN(INPUT,IZ(IBUF1),0)
      CALL GOPEN(IOUT,IZ(IBUF2),1)
C
C      INITIALIZE MATRIX TRAILER

```

WRITING A NEW MODULE

```

C
      MCB(1) = IDUT
      MCB(2) = 0
      MCB(5) = 0
      MCB(7) = 0

C
C      SET UP FOR MATRIX PACK/UNPACK
C
      ITC = KW
      INCR = 1
      ITA = ITC
      ITB = ITC
      INCR1 = 1

C
C      LOOP ON EACH COLUMN
C
      DO 100 I = 1,NCOL

C
C      ONLY BRING IN BAND TERMS
C
      II = 0
      CALL UNPACK(INPUT,I2),RETURN(5(10))

C
C      GET READY TO OUTPUT
C
      III = II
      JJI = JJ
      NTERM = JJ-II+1
      GO TO (20,30,40,50),KW

C
C      FIND MAX -- REAL SINGLE PRECISION
C

```

MODIFICATIONS AND ADDITIONS TO NASTRAN

```

20 XMAX = 0.0
   DO 22 J = 1, NTERM
      XMAX = AMAX1(ABS(Z(J)), XMAX)
22 CONTINUE
   IF(XMAX.LE.0.0) GO TO 80
C
C   NORMALIZE
C
   DO 25 J = 1, NTERM
      Z(J) = Z(J)/XMAX
25 CONTINUE
   GO TO 90
C
C   FIND MAX -- REAL DOUBLE PRECISION
C
30 DMAX = 0.0D0
   DO 32 J = 1, NTERM
      DMAX = DMAX1(DABS(DZ(J)), DMAX)
32 CONTINUE
   IF(DMAX.LE.0.0D0) GO TO 80
C
C   NORMALIZE
C
   DO 35 J = 1, NTERM
      DZ(J) = DZ(J)/DMAX
35 CONTINUE
   GO TO 90
C
C   FIND MAX -- COMPLEX SINGLE PRECISION
C
40 CMAX = 0.0
   DO 42 J = 1, NTERM

```

WRITING A NEW MODULE

```

      CMAX = AMAX1(CABS(CZ(J)),CMAX)
42 CONTINUE
      IF(CMAX.LE.0.0) GO TO 80
C
C   NORMALIZE
C
      DO 45 J = 1,NTERM
      CZ(J) = CZ(J)/CMAX
45 CONTINUE
      GO TO 90
C
C   FIND MAX -- COMPLEX DOUBLE PRECISION
C
50 CDMAX = 0.000
      DO 52 J = 1,NTERM
      CDMAX = DMAX1(DSQRT(CDZ(1,J)**2+CDZ(2,J)**2),CDMAX)
52 CONTINUE
      IF(CDMAX.LE.0.000) GO TO 80
C
C   NORMALIZE
C
      DO 55 J = 1,NTERM
      CDZ(1,J) = CDZ(1,J)/CDMAX
      CDZ(2,J) = CDZ(2,J)/CDMAX
55 CONTINUE
      GO TO 90
C
C   NULL COLUMN --
C
60 I11=1
      JJ1=1
      DO 65 J=1,4

```

MODIFICATIONS AND ADDITIONS TO NASTRAN

```

      85 IZ(J)=0
C
C      OUTPUT NEW COLUMN
C
      90 CALL PACK(IZ,IOUT,MCB)
      100 CONTINUE
C
C      CLOSE OPEN FILES
C
      CALL CLOSE(IOUT,1)
      CALL CLOSE(INPUT,1)
C
C      CREATE TRAILER FOR THE OUTPUT DATA BLOCK
C
      CALL WRTTRL(MCB)
      GO TO 900
C
C      HANDLE CASE WHERE INPUT DATA BLOCK IS PURGED BY SETTING OUTPUT
C      PARAMETERS TO ZERO. OUTPUT DATA BLOCK WILL BE AUTOMATICALLY
C      PURGED BY THE EXECUTIVE SYSTEM SINCE NO TRAILER IS WRITTEN.
C
      500 NCOL = 0
      NRJW = 0
C
C      RETURN TO THE EXECUTIVE SYSTEM
C
      900 RETURN
C
      END

```


6.13 THE NASTPLT MODEL POST PROCESSOR

6.13.1 Introduction

A basic NASTRAN design concept is to make NASTRAN self-contained and independent of local installation software. This concept is implemented for the NASTRAN plot subsystems by incorporating code into NASTRAN to drive plotting devices directly. Thus a user CALCOMP request causes NASTRAN to generate a tape which is mounted directly on a CALCOMP plotter station, thus bypassing installation plot code. In recent years, the plotters employed by NASTRAN users have proliferated, and the concept of installation independence has become impossible to maintain. It is more productive to provide the NASTRAN user a model post processor which can be used with the NASTRAN General Purpose Plotter described in Section 6.10.6, and which can be readily adapted to the user's installation plot software.

The following paragraphs discuss the NASTPLT Model Post-Processor. Section 6.13.2 describes the post-processor and gives implementation information. Section 6.13.3 discusses input data requirements for the post-processor, and Section 6.13.4 presents examples of its usage.

6.13.2 The NASTPLT Post-Processor

6.13.2.1 Overview

The NASTPLT post-processor is designed to provide open-ended plot capabilities while isolating computer and plotter dependent code. The flow of NASTPLT is commented in the code and outlined below. Program MAIN calls PLOT to initialize plotting, inputs plot control directives in Namelist format, then calls TRNSIT to generate a series of plots. TRNSIT calls COMPUT to read and unpack a record of plot commands produced by the General Purpose Plotter. TRNSIT then calls INTG to decode the commands, one at a time. Line and axis commands are executed by calls to PLOT and character (typing) commands are handled by SYMSET. Upon completion of the plots specified by the Namelist directive control is returned to MAIN to process the next set of plot control directives.

6.13.2.2 Implementation

Figure 1 details the subprograms used by NASTPLT. Several of these are computer and plotter independent. Others may require modification: these are described below.

NASTRAN SUPPORT PROGRAMS

SUBROUTINE CØMPUT (KBUF, N, LUN) reads from logical unit LUN a binary record of 3000 characters or bytes, representing 100 plot commands. It then unpacks the characters into the KBUF array. If an end-of-file is encountered on read, N is set to -1.

SUBROUTINE PACK (STRING, IN, CHAR) inserts a single character, CHAR, into position IN in array STRING. Its purpose is to pack characters into a string for plotting.

SUBROUTINE PLØT (X, Y, IPEN) interfaces with the installation plot software. If

IPEN = 0 , initialize the plotter.

IPEN ≤ 0 , start a new frame, if applicable.

IPEN = 2 , draw a line from the current position of the pen to a point (X, Y) inches from the frame origin.

IPEN = 3 , move the pen to position (X, Y).

SUBROUTINE SYMBØL (X, Y, HI, STRING, ANG, N) interfaces with the installation plot software to plot a STRING of N characters. The first character starts at position (X, Y). The characters are HI inches high, and the string forms an angle of ANG degrees with the horizontal.

6.13.3 NASTPLT Input

Input to NASTPLT consists of a series of NASTRAN-generated plot commands on file PLT2, and plotter directives on file INPUT. The plotter directives are input via Namelist /INPUT/ and are defined below. If the user does not input a value for a directive, the indicated default value is used.

<u>Directive</u>	<u>Description</u>	<u>Default</u>
XMAX	Maximum paper length of plot in X-direction, inches	[10.0]
YMAX	Maximum paper length of plot in Y-direction, inches	[10.0]
SPACE	Distance between plots, inches	[3.0]
IPEN	Pen density factor. If greater than one, the coordinates of each line are moved 0.001 inches and redrawn, IPEN-1 times	[1]
HI	Character height, inches	[0.07*YMAX]
NPLØT	Number of plots to be drawn using the current directive set. (When completed, a new directive set is input)	[999]

THE NASTPLT MODEL POST PROCESSOR

<u>Directive</u>	<u>Description</u>	<u>Default</u>
IDEBUG	Number of lines of debug output, one plot command per line, to be printed	[0]
LUN	FORTTRAN logical unit number of the plot input file	[1]
LIN	FORTTRAN logical unit number of the card reader	[5]
LØUT	FORTTRAN logical unit number of the printer	[6]

6.13.4 Use of NASTPLT with NASTRAN

Use of the NASTPLT plot post-processor as an interface between NASTRAN and the installation plot software offers the user flexibility which is not available in other modes. Some examples are described below.

- (1) Character size. When using the plot-postprocessor, the recommended Case Control card is [PLØTTER NASTPLT T, 1], which specifies a table plotter operating in typing mode. These terms are meaningless to NASTPLT, but they allow the user flexibility in determining character size. For instance, the table plotter software in NASTRAN spaces characters appropriately for a 0.1 inch character height for any specified paper size. If it is desired to print smaller, more closely spaced characters on a 10 x 10 inch plot, the user could specify to NASTRAN a paper size of 15 x 15, and to NASTPLT XMAX = YMAX = 10. Conversely, if a larger character size were desired, a smaller paper size could be input to NASTRAN.

The user who is not concerned about character size can simplify implementation of the plot-postprocessor by specifying T, 0 on the PLØTTER card. This makes NASTRAN draw the characters as line segments. (Note that this produces about six 30 byte plot commands per character, which can generate a very large PLT2 file if many characters are to be drawn.)

- (2) Use of Different Plotters. NASTPLT does not rewind the NASTRAN plot tape PLT2. Therefore, the user can process the plot tape in several job steps, directing the output of each step to a separate plotter, or changing pens or paper size between plots.

NASTRAN SUPPORT PROGRAMS

Subprogram	Type	Function
MAIN	Machine Independent	Process input directives
TRNSIT	Machine Independent	Interpret and process NASTRAN plot commands
SYMSET	Machine Independent	Pack character strings for plotting
INTG	Machine Independent	Decode (X, Y) coordinates in plot commands
COMPUT	Machine Dependent	Read and unpack a record of 100 NASTRAN plot commands
PACK	Machine Dependent	Pack a Hollerith character string
PLØT	Plotter Dependent	Call installation plot routines
SYMBOL	Plotter Dependent	Call installation character string plot routines

Figure 2. NASTPLT Plot-Postprocessor - Summary of Subprograms

7.1 RESTART TABLES

Restart tables have been designed for the NASTRAN system to give a user the capability to restart a solution to a user's problem. The definition of a restart in the NASTRAN system is the capability to continue a solution to a user problem via follow on executions without having to rerun an entire rigid format sequence. NASTRAN contains several rigid format (DMAP) solutions. For an explanation of DMAP see Section 5.0 of the NASTRAN User's Manual. The user may select one of these rigid formats to solve or analyze a problem. To understand the relationship of restart to a rigid format involves an understanding of the three restart tables that comprise the Module Execution Discussion Table (see Section 1.10). The three restart tables that make up the MEDT in NASTRAN are:

- CARD NAME RESTART TABLES
- FILE NAME RESTART TABLES
- RIGID FORMAT CHANGE RESTART TABLES

Along with the three restart tables are their complementary bit position tables as follows:

- CARD NAME RESTART BIT POSITION TABLES
- FILE NAME RESTART BIT POSITION TABLES
- RIGID FORMAT CHANGE BIT POSITION TABLES

The bit position tables have been designed to associate a particular name with a selected bit position. Bulk data and case control card deck names are associated with bit positions defined in the Card Name Restart Bit Position Table. Mnemonics with \$ appended are always associated with changes in the Case Control deck. All DMAP file names are associated with bit positions in the File Name Restart Bit Position Table. Likewise rigid format solution numbers are associated with bit positions in the Rigid Format Change Bit Position Tables. Each rigid format has its own unique set of restart and bit position tables. For an example of each table for Rigid Format 1, see Sections 10.2.1 through 10.2.5.

7.1.1 Card Name Restarts

The Card Name Restart Table consists of DMAP module names (section 10.2.3). Associated with each module name there are bit positions which link the DMAP module with bulk data or case

RIGID FORMAT RESTART TABLES

control cards given in the Card Name Restart Bit Position Table (section 10.2.1). NASTRAN uses the bit position as a tag identifying which DMAP module to execute. The following example illustrates the use of these tables in determining the effects of changing a particular bulk data card on a modified restart. Suppose the FØRCE bulk data card is changed by the user when executing Rigid Format 1. The Card Name Restart Bit Position Table associates bit 60 with the FØRCE bulk data card, therefore and DMAP module name associated with bit 60 in the Card Name Restart Table will be executed.

7.1.2 Rigid Format Restarts

The Rigid Format Restart Bit Position Table is a two dimensional table consisting of bit positions vs. solution numbers (section 10.1). The solution numbers identify particular rigid formats. Bit positions 63 through 93 are associated with solution numbers (section 10.1). Bit positions 78 through 93 are reserved for additional rigid formats. Restarts involving a solution number (rigid format) change may require particular DMAP modules to be executed as opposed to executing the entire DMAP sequence. Required DMAP modules may have already been executed by the previous solution (rigid format). The previous solution number was placed on the old problem tape of the current NASTRAN run. The Rigid Format Restart Bit Position Table is scanned for the bit associated with the old problem tape solution number. The bit used is associated with particular DMAP instructions to be executed in the Rigid Format Restart Table.

7.1.3 File Name Restarts

The File Name Restart Table consists of DMAP module names (section 10.2.5). Associated with each module name are bit positions which link the DMAP module with file names given in the File Name Restart Bit Position Table (section 10.2.2). NASTRAN uses the bit positions as a tag identifying which DMAP module to execute generation of the required data blocks (files). When a restart run is executed, NASTRAN searches the DMAP instructions to be executed for files required for the run (after determination of DMAP modules to be executed due to changes in Case Control, Bulk Data, and Solution Number). These files are compared with files on the Old Problem Tape (ØPTP). If any files are required that are not available from the ØPTP, the modules which generate these files will be executed. The modules to be executed are identified through the File Name Restart Table. The following example illustrates the use of these tables in determining which modules to execute during a restart to satisfy file requirements. Suppose

RIGID FORMAT RESTART TABLES

a restart is made using rigid format 1 and the file BGPDT is required and not available from the ØPTP. The File Name Restart Bit Position Table associates bit 94 with file BGPDT, therefore any DMAP module name associated with bit 94 in the File Names Restart Table will be executed.

STATIC ANALYSIS

7.2 RESTART TABLES FOR STATIC ANALYSIS

7.2.1 Bit Positions for Card Name Restart Table

Card Name	Bit Pos.	Card Name	Bit Pos.	Card Name	Bit Pos.
AXIC	1	CQDPLT	2	PTRPLT1	3
AXIF	1	CQUAD1	2	PTRSHL	3
CELAS1	1	CQUAD2	2	PTUBE	3
CELAS2	1	CQUADTS	2	PTWIST	3
CELAS3	1	OROD	2	GENEL	4
CELAS4	1	CSHEAR	2	CONM1	5
CMASS1	1	CTETRA	2	CONM2	5
CMASS2	1	CTORDRG	2	PELAS	6
CMASS3	1	CTRAPAX	2	PMASS	7
CMASS4	1	CTRAPRG	2	MAT1	8
CORD1C	1	CTRBSC	2	MAT2	8
CORD1R	1	CTRIA1	2	MAT3	8
CORD1S	1	CTRIA2	2	MAT4	8
CORD2C	1	CTRIAAX	2	MAT5	8
CORD2R	1	CTRIARG	2	MATT1	8
CORD2S	1	CTRIATS	2	MATT2	8
GRDSET	1	CTRIM6	2	MATT3	8
GRID	1	CTRMEM	2	MATT4	8
GRIDB	1	CTRPLT	2	MATT5	8
POINTAX	1	CTRPLT1	2	TABLEM1	8
RINGAX	1	CTRSHL	2	TABLEM2	8
RINGFL	1	CTUBE	2	TABLEM3	8
SECTAX	1	CTWIST	2	TABLEM4	8
SEQGP	1	CWEDGE	2	TEMPMT\$	8
SPOINT	1	PBAR	3	TEMPMX\$	8
ADUM1	2	PCONEAX	3	AXISYM\$	9
ADUM2	2	PDUM1	3	CRIGD1	9
ADUM3	2	PDUM2	3	CRIGD2	9
ADUM4	2	PDUM3	3	CRIGD3	9
ADUM5	2	PDUM4	3	CRIGDR	9
ADUM6	2	PDUM5	3	MPC	9
ADUM7	2	PDUM6	3	MPCADD	9
ADUM8	2	PDUM7	3	MPCAX	9
ADUM9	2	PDUM8	3	MPC\$	9
BAROR	2	PDUM9	3	SPC	10
CBAR	2	PHBDY	3	SPC1	10
CCONEAX	2	PIHEX	3	SPCADD	10
CDUM1	2	PQDMEM	3	SPCAX	10
CDUM2	2	PQDMEM1	3	SPC\$	10
CDUM3	2	PQDMEM2	3	ASET	11
CDUM4	2	PQDMEM3	3	ASET1	11
CDUM5	2	PQDPLT	3	OMIT	11
CDUM6	2	PQUAD1	3	OMIT1	11
CDUM7	2	PQUAD2	3	OMITAX	11
CDUM8	2	PQUADTS	3	SUPAX	12
CDUM9	2	PROD	3	SUPPORT	12
CHBDY	2	PSHEAR	3	TEMP	13
CHEXA1	2	PTORDRG	3	TEMPAX	13
CHEXA2	2	PTRAPAX	3	TEMPO	13
CIHEX1	2	PTRBSC	3	TEMPP1	13
CIHEX2	2	PTRIA1	3	TEMPP2	13
CIHEX3	2	PTRIA2	3	TEMPP3	13
CONROD	2	PTRIAAX	3	TEMPRB	13
CQDMEM	2	PTRIAATS	3	WTHASS	14
CQDMEM1	2	PTRIM6	3	GRDPNT	15
CQDMEM2	2	PTRMEM	3	PLOTEL	16
CQDMEM3	2	PTRPLT	3	IRES	17

RIGID FORMAT RESTART TABLES

<u>Card Name</u>	<u>Bit Pos.</u>
PLOTS	18
POUTS	19
LOOPS	22
LOOP1\$	23
COUPMASS	24
CPBAR	24
CPQOPLT	24
CPQUAD1	24
CPQUAD2	24
CPRD	24
CPTRBSC	24
CPTRIA1	24
CPTRIA2	24
CPTRPLT	24
CPTUBE	24
NOLOOPS	31
DEFORM	59
DEFORM\$	59
LOAD\$	59
RFORCE\$	59
SPCD	59
FORCE	60
FORCE1	60
FORCE2	60
FORCEAX	60
LOAD	60
MOMAX	60
MOMENT	60
MOMENT1	60
MOMENT2	60
PLOAD	60
PLOAD1	60
PLOAD2	60
PLOAD3	60
PRESAX	60
SLOAD	60
GRAV	61
RFORCE	61
TEMPLD\$	62

STATIC ANALYSIS

7.2.2 Bit Positions for File Name Restart Table

<u>File Name</u>	<u>Bit Pos.</u>	<u>File Name</u>	<u>Bit Pos.</u>
BGPD	94	LLL	108
CSTM	94	DM	109
EQEXIN	94	PG	110
GPD	94	PL	111
GPL	94	PO	111
SIL	94	PS	111
ECT	95	QR	111
GPTT	96	RULV	112
SLT	96	RUOV	112
GPECT	97	ULV	112
EST	97	UOOV	112
GEI	97	PGG	113
GPST	98	QG	113
KGGX	98	UGV	113
MGG	99	DEF1	114
KGG	100	OES1	114
ASET	101	OPG1	114
RG	101	OQG1	114
USET	101	OUGV1	114
YS	101	PUGV1	114
OGPST	102	ELSETS	115
GM	103	GPSETS	115
KNN	104	PLTPAR	115
KFF	105	PLTSETX	115
KFS	105	KDICT	116
KSS	105	KELM	116
GO	106	MDICT	116
KAA	106	MELM	116
KOO	106	OPTPT1	117
LOO	106	OPTPT2	118
KLL	107	EST1	118
KLR	107	OGPF81	119
KRR	107	ONRGY1	119

RIGID FORMAT RESTART TABLES

7.2.3 Card Name Restart Table

DMAP Inst.	<u>Bit Position</u>						
	1	10	20	30	40	50	60
BEGIN	1234567890123456789	234					9012
FILE	1234567890123456789	234					9012
\$\$\$	9						
FILE	1234567890123456789	234					9012
\$\$\$	1 3						
GP1	1						
SAVE	1						
CHKPNT	1						
\$\$\$	6						
GP2	12 45		6				
CHKPNT	12 45		6				
\$\$\$	6						
PARAML			8				
\$\$\$	7						
PURGE			8				
\$\$\$	7						
COND			8				
\$\$\$	7						
PLTSET			8				
\$\$\$	7						
SAVE			8				
\$\$\$	7						
PRTMSG			8				
\$\$\$	7						
PARAM			8				
\$\$\$	7						
PARAM			8				
\$\$\$	7						
COND			8				
\$\$\$	7						
PLOT			8				
\$\$\$	7						
SAVE			8				
\$\$\$	7						
PRTMSG			8				
\$\$\$	7						
LABEL			8				
\$\$\$	7						
CHKPNT			8				
\$\$\$	67						
GP3	12		3				01
SAVE	12		3				01
PARAM	12		3 5				01
CHKPNT	12		3 5				01
\$\$\$	6						
TA1	1234567		3				
SAVE	1234567		3				
PARAM	1234567		3				
COND	1234567		3				
PURGE	1234567		3				
CHKPNT	1234567		3				
\$\$\$	6						
OPTPR1	1234567						
\$\$\$	9						
SAVE	1234567						
\$\$\$	9						
CHKPNT	1234567						
\$\$\$	6 9						
PARAM			9				

STATIC ANALYSIS

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
JUMP	1234567						
SSS		9					
LABEL	1234567						
SSS		9					
COND	123 5678	45	4				1
PARAM	123 6 8						
EQUIV	1234567						
SSS		9					
EMG	123 5678	45	4				1
SAVE	123 5678	45	4				1
CHKPNT	123 5678	45	4				1
SSS	6						
COND	123 6 8						
EMA	123 6 8						
CHKPNT	123 6 8						
SSS	6						
LABEL	123 6 8						
COND	123 5 78	45	4				1
EMA	123 5 78	45	4				1
CHKPNT	123 5 78	45	4				1
SSS	6						
LABEL	123 5 78	45	4				1
COND	123 5 78	45	4				
SSS	8						
COND	123 5 78	45	4				
SSS	8						
GPWG	123 5 78	45	4				
SSS	8						
OFF	123 5 78	45	4				
SSS	8						
LABEL	123 5 78	45	4				1
EQUIV	1234 6 8						
CHKPNT	1234 6 8						
SSS	6						
COND	1234 6 8						
SMA3	1234 6 8						
CHKPNT	1234 6 8						
SSS	6						
LABEL	1234 6 8						
PARAM	1	9012	23	1			
JUMP			23				
SSS	1 3						
LABEL			23				
SSS	1 3						
GP4	1	9012	23	1			
SAVE	1	9012	23	1			
COND	1	9012	23				9
PARAM	1	9012	23				9
PURGE	1	9012	23				9
CHKPNT	1234 6 89012		23				9
SSS	6						
COND	123 6 890		23				
GPSP	123 6 890		23				
SAVE	123 6 890		23				
COND	123 6 890		23				
OFF	123 6 890		23				
LABEL	123 6 890		23				
EQUIV	1234 6 89		23				
CHKPNT	1234 6 89		23				
SSS	6						

RIGID FORMAT RESTART TABLES

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
COND	1234	6 89		23			
MCE1	1	9		23			
CHKPNT	1	9		23			
\$SS		6					
MCE2	1234	6 89		23			
CHKPNT	1234	6 89		23			
\$SS		6					
LABEL	1234	6 89		23			
EQUIV	1234	5 890		23			
CHKPNT	1234	6 890		23			
\$SS		6					
COND	1234	6 890		23			
SCE1	1234	6 890		23			
CHKPNT	1234	6 890		23			
\$SS		6					
LABEL	1234	6 890		23			
EQUIV	1234	6 8901		23			
CHKPNT	1234	6 8901		23			
\$SS		6					
COND	1234	6 8901		23			
SMP1	1234	6 8901		23			
CHKPNT	1234	6 8901		23			
\$SS		6					
LABEL	1234	6 8901		23			
EQUIV	1234	6 89012		23			
CHKPNT	1234	6 89012		23			
\$SS		6					
COND	1234	6 89012		23			
RBMG1	1234	6 89012		23			
CHKPNT	1234	6 89012		23			
\$SS		6					
LABEL	1234	6 89012		23			
RBMG2	1234	6 89012		23			
CHKPNT	1234	6 89012		23			
\$SS		6					
COND	1234	6 89012		23			
RBMG3	1234	6 89012		23			
CHKPNT	1234	6 89012		23			
\$SS		6					
LABEL	1234	6 89012		23			
SSG1	123	5678		23			9012
CHKPNT	123	5678		23			9012
\$SS		6					
EQUIV	123	56789012		23			9012
CHKPNT	123	56789012		23			9012
\$SS		6					
COND	123	56789012		23			9012
SSG2	123	56789012		23			9012
CHKPNT	123	56789012		23			9012
\$SS		6					
LABEL	123	56789012		23			9012
SSG3	123456789012			23			9012
SAVE	123456789012			23			9012
CHKPNT	123456789012			23			9012
\$SS		6					
COND	123456789012	7		23			9012
MATGPR	123456789012	7		23			9012
MATGPR	123456789012	7		23			9012
LABEL	123456789012	7		23			9012

STATIC ANALYSIS

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
SDR1	123456789012		23				9012
CHKPNT	123456789012		23				9012
\$\$\$	6						
COND			23				
\$\$\$	1 3						
REPT			23				
\$\$\$	1 3						
JUMP			23				
\$\$\$	1 3						
PARAM			23				
COND			23				
LABEL			23				
\$\$\$	1 3						
CHKPNT	123456789012						9012
\$\$\$	6						
GPFDR			89				
OFF			89				
COND			23				
EQMCK			23				
OFF			23				
SAVE			23				
LABEL			23				
SDR2			89				
SAVE			89				
CURV			89				
SDR2			89				
CURV			89				
COND			89				
SDR3			89				
OFF			89				
SAVE			89				
OFF			89				
SAVE			89				
XYTRAN			89				
\$\$\$	7						
SAVE			89				
\$\$\$	7						
XYPLOT			89				
\$\$\$	7						
JUMP			89				
\$\$\$	7						
LABEL			89				
COND			89				
\$\$\$	9						
OPTPR2			89				
\$\$\$	9						
SAVE			89				
\$\$\$	9						
EQUIV			89				
\$\$\$	9						
COND			89				
\$\$\$	9						
LABEL			89				
\$\$\$	9						
OFF			9				
SAVE			9				
OFF			9				
SAVE			9				
COND			8				

RIGID FORMAT RESTART TABLES

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
\$\$\$		7					
LABEL			89				
\$\$\$		7					
PLOT			8				
\$\$\$		7					
SAVE			8				
\$\$\$		7					
PRTMSG			8				
\$\$\$		7					
LABEL			8				
\$\$\$		7					
LABEL			8				
\$\$\$		9					
COND			8				
\$\$\$		9					
REPT			8				
\$\$\$		9					
JUMP	1234567890123456789		234				9012
LABEL	1234567890123456789		234				9012
\$\$\$	1 3						
PRTPARM	1234567890123456789		234				9012
\$\$\$	1 3						
LABEL	1234567890123456789		234				9012
\$\$\$	8						
PRTPARM	1234567890123456789		234				9012
\$\$\$	8						
LABEL	1234567890123456789		234				9012
PRTPARM	1234567890123456789		234				9012
LABEL	1234567890123456789		234				9012
PRTPARM	1234567890123456789		234				9012
LABEL	1234567890123456789		234				9012
PRTPARM	1234567890123456789		234				9012
LABEL	1234567890123456789		234				9012
END	1234567890123456789		234				9012

STATIC ANALYSIS

7.2.4 Rigid Format Change Restart Table

DMAP Inst.	64	<u>Bit Position</u> 70	80	DMAP Inst.	64	<u>Bit Position</u> 70	80
BEGIN		45678901234567	345	EQUIV			
FILE		45678901234567	345	CHKPNT			
FILE		45678901234567	345	COND			
GP1				SMA3			
SAVE				CHKPNT			
CHKPNT				LABEL			
GP2				PARAM			
CHKPNT				JUMP			
PARAML				LABEL			
PURGE				GP4			
COND				SAVE			
PLTSET				COND	45678901234567		345
SAVE				PARAM			
PRTMSG				PURGE			
PARAM				CHKPNT			
PARAM				COND			
COND				GPSP			
PLOT				SAVE			
SAVE				COND			
PRTMSG				OFF			
LABEL				LABEL			
CHKPNT				EQUIV			
GP3				CHKPNT			
SAVE				COND			
PARAM	45678901234567		345	MCE1			
CHKPNT				CHKPNT			
TA1				MCE2			
SAVE				CHKPNT			
PARAM	45678901234567		345	LABEL			
COND	45678901234567		345	EQUIV			
PURGE				CHKPNT			
CHKPNT				COND			
OPTPR1				SCE1			
SAVE				CHKPNT			
CHKPNT				LABEL			
PARAM				EQUIV			
JUMP				CHKPNT			
LABEL				COND			
COND				SMP1			
PARAM				CHKPNT			
EQUIV				LABEL			
EMG				EQUIV			
SAVE				CHKPNT			
CHKPNT				COND			
COND				RBMG1			
EMA				CHKPNT			
CHKPNT				LABEL			
LABEL				RBMG2			
COND				CHKPNT			
EMA				COND			
CHKPNT				RBMG3			
LABEL				CHKPNT			
COND				LABEL			
COND				SSG1			
GPWG				CHKPNT			
OFF				EQUIV			
LABEL				CHKPNT			

RIGID FORMAT RESTART TABLES

DMAP Inst.	64	Bit Position 70	80
---------------	----	--------------------	----

COND			
SSG2			
CHKPNT			
LABEL			
SSG3	4		
SAVE	4		
CHKPNT	4		
COND	45678901234567	345	
MATGPR	45678901234567	345	
MATGPR	45678901234567	345	
LABEL	45678901234567	345	
SDR1	45678901234567	345	
CHKPNT	45678901234567	345	
COND	45678901234567	345	
REPT	45678901234567	345	
JUMP	45678901234567	345	
PARAM	45678901234567	345	
COND	45678901234567	345	
LABEL	45678901234567	345	
CHKPNT	45678901234567	345	
GPFOR			
OFF			
COND			
EQMCK			
OFF			
SAVE			
LABEL			
SDR2			
SAVE			
CURV			
SDR2			
CURV			
COND			
SDR3			
OFF			
SAVE			
OFF			
SAVE			
XYTRAN			
SAVE			
XYPLOT			
JUMP			
LABEL			
COND			
OPTPR2			
SAVE			
EQUIV			
COND			
LABEL			
OFF			
SAVE			
OFF			
SAVE			
COND			
LABEL			
PLOT			
SAVE			
PRTMSG			
LABEL			

DMAP Inst.	64	Bit Position 70	80
---------------	----	--------------------	----

LABEL			
COND			
REPT			
JUMP	45678901234567	345	
LABEL	45678901234567	345	
PRTPARM	45678901234567	345	
LABEL	45678901234567	345	
PRTPARM	45678901234567	345	
LABEL	45678901234567	345	
PRTPARM	45678901234567	345	
LABEL	45678901234567	345	
PRTPARM	45678901234567	345	
LABEL	45678901234567	345	
PRTPARM	45678901234567	345	
LABEL	45678901234567	345	
END	45678901234567	345	

STATIC ANALYSIS

7.2.5 File Name Restart Table

DMAP Inst.	94	100	110	120	DMAP Inst.	94	100	110	120
BEGIN					COND		0		
FILE					SMA3		0		
FILE					CHKPNT		0		
GP1	4				LABEL		0		
SAVE	4				PARAM		1		
CHKPNT	4				JUMP		1		
GP2	5				LABEL		0		
CHKPNT	5				GP4		1		
PARAML			5		SAVE		1		
PURGE			5		COND		1		
COND			5		PARAM		1		
PLTSET			5		PURGE		1	3 567 9 1	
SAVE			5		CHKPNT		1	3 567 9 1	
PRTMSG			5		COND		2		
PARAM			5		GPSP		2		
PARAM			5		SAVE		2		
COND					COND		2		
PLOT					OFF		2		
SAVE					LABEL		2		
PRTMSG					EQUIV			4	
LABEL					CHKPNT			4	
CHKPNT			5		COND		34		
GP3	6				MCE1		3		
SAVE	6				CHKPNT		3		
PARAM	6 9				MCE2			4	
CHKPNT	6				CHKPNT			4	
TA1	7				LABEL		34		
SAVE	7				EQUIV			5	
PARAM	7				CHKPNT			5	
COND	7				COND			5	
PURGE	78 2				SCE1			5	
CHKPNT	7				CHKPNT			5	
OPTPR1				7	LABEL			5	
SAVE				7	EQUIV			6	
CHKPNT				7	CHKPNT			6	
PARAM					COND			6	
JUMP				7	SMP1			6	
LABEL				7	CHKPNT			6	
COND	89				LABEL			6	
PARAM	8				EQUIV			7	
EQUIV				8	CHKPNT			7	
EMG				6	COND			7	
SAVE				6	RBMG1			7	
CHKPNT				6	CHKPNT			7	
COND	8				LABEL			7	
EMA	8				RBMG2			8	
CHKPNT	8				CHKPNT			8	
LABEL	8				COND			9	
COND	9				RBMG3			9	
EMA	9				CHKPNT			9	
CHKPNT	9				LABEL			9	
LABEL	9				SSG1			0	
COND					CHKPNT			0	
COND					EQUIV			1	
GPWG					CHKPNT			1	
OFF					COND			1	
LABEL	89				SSG2			1	
EQUIV	n				CHKPNT			1	
CHKPNT	0				LABEL			1	

RIGID FORMAT RESTART TABLES

DMAP Inst.	94	100	Bit Position	110	120	DMAP Inst.	94	100	Bit Position	110	120
SSG3				2		LABEL					
SAVE				2		PRTPARM					
CHKPNT				2		LABEL					
COND						PRTPARM					
MATGPR						LABEL					
MATGPR						PRTPARM					
LABEL						LABEL					
SDR1				3		PRTPARM					
CHKPNT				3		LABEL					
COND						PRTPARM					
REPT						LABEL					
JUMP						END					
PARAM											
COND											
LABEL											
CHKPNT				3							
GPFDOR					9						
OFF					9						
COND											
EQMCK											
OFF											
SAVE											
LABEL											
SDR2				4							
SAVE											
CURV				4							
SDP2				4							
CURV				4							
COND											
SDR3											
OFF											
SAVE											
OFF											
SAVE											
XYTRAN											
SAVE											
XYPLOT											
JUMP											
LABEL											
COND					8						
OPTPR2					8						
SAVE					8						
EQUIV					8						
COND					8						
LABEL					8						
OFF											
SAVE											
OFF											
SAVE											
COND											
LABEL											
PLOT											
SAVE											
PRTMSG											
LABEL											
LABEL											
COND											
REPT											
JUMP											

STATIC ANALYSIS WITH INERTIA RELIEF

7.3 RESTART TABLES FOR STATIC ANALYSIS WITH INERTIA RELIEF

7.3.1 Bit Positions for Card Name Restart Table

Card Name	Bit Pos.	Card Name	Bit Pos.	Card Name	Bit Pos.
AXIC	1	CQDPLT	2	PTRPLT1	3
AXIF	1	CQUAD1	2	PTRSHL	3
CELAS1	1	CQUAD2	2	PTUBE	3
CELAS2	1	CQUADTS	2	PTWIST	3
CELAS3	1	CROD	2	GENEL	4
CELAS4	1	CSHEAR	2	CONM1	5
CMASS1	1	CTETRA	2	CONM2	5
CMASS2	1	CTORDRG	2	PELAS	6
CMASS3	1	CTRAPAX	2	PMASS	7
CMASS4	1	CTRAPRG	2	MAT1	8
CORD1C	1	CTRBSC	2	MAT2	8
CORD1R	1	CTRIA1	2	MAT3	8
CORD1S	1	CTRIA2	2	MATT1	8
CORD2C	1	CTRIAAX	2	MATT2	8
CORD2R	1	CTRIARG	2	MATT3	8
CORD2S	1	CTRIATS	2	TABLEM1	8
GRDSET	1	CTRIM6	2	TABLEM2	8
GRID	1	CTRMEM	2	TABLEM3	8
GRIDB	1	CTRPLT	2	TABLEM4	8
POINTAX	1	CTRPLT1	2	TEMPM1\$	8
RINGAX	1	CTRSHL	2	TEMPM2\$	8
RINGFL	1	CTUBE	2	AXISYM\$	9
SECTAX	1	CTWIST	2	CRIGD1	9
SEQGP	1	CWEDGE	2	CRIGD2	9
SPOINT	1	PBAR	3	CRIGD3	9
ADUM1	2	PCONEAX	3	CRIGDR	9
ADUM2	2	PDUM1	3	MPC	9
ADUM3	2	PDUM2	3	MPCADD	9
ADUM4	2	PDUM3	3	MPCAX	9
ADUM5	2	PDUM4	3	MPC\$	9
ADUM6	2	PDUM5	3	SPC	10
ADUM7	2	PDUM6	3	SPC1	10
ADUM8	2	PDUM7	3	SPCADD	10
ADUM9	2	PDUM8	3	SPCAX	10
BAROR	2	PDUM9	3	SPC\$	10
CBAR	2	PIHEX	3	ASET	11
CCONEAX	2	PQDMEM	3	ASET1	11
CDUM1	2	PQDMEM1	3	OMIT	11
CDUM2	2	PQDMEM2	3	OMIT1	11
CDUM3	2	PQDMEM3	3	OMITAX	11
CDUM4	2	PQOPLT	3	SUPAX	12
CDUM5	2	PQUAD1	3	SUPORT	12
CDUM6	2	PQUAD2	3	TEMP	13
CDUM7	2	PQUADTS	3	TEMPAX	13
CDUM8	2	PROD	3	TEMPO	13
CDUM9	2	PSHEAR	3	TEMPP1	13
CHEXA1	2	PTORDRG	3	TEMPP2	13
CHEXA2	2	PTRAPAX	3	TEMPP3	13
CIHEX1	2	PTRBSC	3	TEMPRB	13
CIHEX2	2	PTRIA1	3	WTMASS	14
CIHEX3	2	PTRIA2	3	GRDPNT	15
CONROD	2	PTRIAAX	3	PLOTEL	16
CQDMEM	2	PTRIATS	3	IRES	17
CQDMEM1	2	PTRIM6	3	PLOT\$	18
CQDMEM2	2	PTRMEM	3	POUT\$	19
CQDMEM3	2	PTRPLT	3	LOOP\$	22

RIGID FORMAT RESTART TABLES

Card Name Bit Pos.

LOOP1\$	23
COUPMASS	24
CPBAR	24
CPQDPLT	24
CPQUAD1	24
CPQUAD2	24
CPROD	24
CPTRBSC	24
CPTRIA1	24
CPTRIA2	24
CPTRPLT	24
CPTUBE	24
DEFORM	59
DEFORM\$	59
LOAD\$	59
RFORCE\$	59
SPCD	59
FORCE	60
FORCE1	60
FORCE2	60
FORCEAX	60
LOAD	60
MOMAX	60
MOMENT	60
MOMENT1	60
MOMENT2	60
PLOAD	60
PLOAD1	60
PLOAD2	60
PLOAD3	60
PRESAX	60
SLOAD	60
GRAV	61
RFORCE	61
TEMPLD\$	62

STATIC ANALYSIS WITH INERTIA RELIEF

7.3.2 Bit Positions for File Name Restart Table

<u>File Name</u>	<u>Bit Pos.</u>	<u>File Name</u>	<u>Bit Pos.</u>
BGPD	94	KLL	107
CSTM	94	KLR	107
EQEXIN	94	KRR	107
GPDT	94	MLL	107
GPL	94	MLR	107
SIL	94	MRR	107
ECT	95	LLL	108
GPTT	96	DM	109
SLT	96	MR	110
GPECT	97	PG	111
EST	97	PL	112
GEI	97	PO	112
GPST	98	PS	112
KGGX	98	QR	112
MGG	99	PLI	113
KGG	100	POI	113
ASET	101	RULV	114
RG	101	RUOV	114
USET	101	ULV	114
YS	101	UOV	114
OGPST	102	PGG	115
GM	103	QG	115
KNN	104	UGV	115
MNN	104	DEF1	116
KFF	105	OES1	116
KFS	105	OPG1	116
KSS	105	OQG1	116
MFF	105	OUGV1	116
GO	106	PUGV1	116
KAA	106	ELSETS	117
KOO	106	GPSETS	117
LOO	106	PLTPAR	117
MAA	106	PLTSETX	117
MOA	106	KDICT	118
MOO	106	KELM	118
		MDICT	118
		MELM	118

RIGID FORMAT RESTART TABLES

7.3.3 Card Name Restart Table

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
BEGIN	1234567890123456789		234				9012
FILE	1234567890123456789		234				9012
\$\$\$	1 3						
GP1	1						
SAVE	1						
CHKPNT	1						
\$\$\$		6					
GP2	12 45		6				
CHKPNT	12 45		6				
\$\$\$		6					
PARAML			8				
\$\$\$		7					
PURGE			8				
\$\$\$		7					
COND			8				
\$\$\$		7					
PLTSET			8				
\$\$\$		7					
SAVE			8				
\$\$\$		7					
PRTMSG			8				
\$\$\$		7					
PARAM			8				
\$\$\$		7					
PARAM			8				
\$\$\$		7					
COND			8				
\$\$\$		7					
PLOT			8				
\$\$\$		7					
SAVE			8				
\$\$\$		7					
PRTMSG			8				
\$\$\$		7					
LABEL			8				
\$\$\$		7					
CHKPNT			8				
\$\$\$		67					
GP3	12		3				01
CHKPNT	12		3				01
\$\$\$		6					
TA1	1234567		3				
SAVE	1234567		3				
COND	12345678		34		4		
PURGE	1234567		3				
CHKPNT	1234567		3				
\$\$\$		6					
PARAM	123 6 8						
PARAM	123 5 78		45		4		1
EMG	123 5678		45		4		1
SAVE	123 5678		45		4		1
CHKPNT	123 5678		45		4		1
\$\$\$		6					
COND	123 6 8						
EMA	123 6 8						
CHKPNT	123 6 8						
\$\$\$		6					
LABEL	123 6 8						

STATIC ANALYSIS WITH INERTIA RELIEF

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
COND	123 5 7 8	45	4				1
EMA	123 5 7 8	45	4				1
CHKPNT	123 5 7 8	45	4				1
\$SS	6						1
COND	123 5 7 8	45	4				
\$SS	8						
GPWG	123 5 7 8	45	4				
\$SS	8						
OFF	123 5 7 8	45	4				
\$SS	8						
LABEL	123 5 7 8	45	4				
\$SS	8						
EQUIV	1234 6 8						
CHKPNT	1234 6 8						
\$SS	6						
COND	1234 6 8						
SMA3	1234 6 8						
CHKPNT	1234 6 8						
\$SS	6						
LABEL	1234 6 8						
PARAM	1	9012	23				
JUMP			23				
\$SS	1 3						
LABEL			23				
\$SS	1 3						
GP4	1	9012	23				9
SAVE	1	9012	23				9
COND	1	9012	23				9
COND	1	9012	23				9
PURGE	1	9012	23				9
CHKPNT	123456789012		23				9
\$SS	6						
COND	123 6 8 90		23				
GPSP	123 6 8 90		23				
SAVE	123 6 8 90		23				
COND	123 6 8 90		23				
OFF	123 6 8 90		23				
LABEL	123 6 8 90		23				
EQUIV	123456789		23				
CHKPNT	123456789		23				
\$SS	6						
COND	123456789		23				
MCE1	1 9		23				
CHKPNT	1 9		23				
\$SS	6						
MCE2	123456789		23				
CHKPNT	123456789		23				
\$SS	6						
LABEL	123456789		23				
EQUIV	1234567890		23				
CHKPNT	1234567890		23				
\$SS	6						
COND	1234567890		23				
SCE1	1234567890		23				
CHKPNT	1234567890		23				
\$SS	6						
LABEL	1234567890		23				
EQUIV	12345678901		23				
CHKPNT	12345678901		23				

RIGID FORMAT RESTART TABLES

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
\$SS	6						
COND	12345678901		23				
SMP1	12345678901		23				
CHKPNT	12345678901		23				
\$SS	6						
LABEL	12345678901		23				
REMG1	123456789012		23				
CHKPNT	123456789012		23				
\$SS	6						
REMG2	1234 6 89012		23				
CHKPNT	1234 6 89012		23				
\$SS	6						
REMG3	1234 6 89012		23				
CHKPNT	1234 6 89012		23				
\$SS	6						
REMG4	123456789012		23				
CHKPNT	123456789012		23				
\$SS	6						
SSG1	123 5678 3		23				9012
CHKPNT	123 5678 3		23				9012
\$SS	6						
SSG2	123 567890123		23				9012
CHKPNT	123 567890123		23				9012
\$SS	6						
SSG4	123 567890123		23				9012
CHKPNT	123 567890123		23				9012
\$SS	6						
SSG3	1234567890123		23				9012
SAVE	1234567890123		23				9012
CHKPNT	1234567890123		23				9012
\$SS	6						
COND	1234567890123	7	23				9012
MATGPR	1234567890123	7	23				9012
MATGPR	1234567890123	7	23				9012
LABEL	1234567890123	7	23				9012
SDR1	1234567890123		23				9012
CHKPNT	1234567890123		23				9012
\$SS	6						
COND			23				
\$SS	1 3						
REPT			23				
\$SS	1 3						
JUMP			23				
\$SS	1 3						
PARAM			23				
COND			23				
LABEL			23				
\$SS	1 3						
CHKPNT		9					
COND			23				
EQMCK			23				
OFF			23				
SAVE			23				
LABEL			23				
SDR2		89					
PARAM		9					
OFF		9					
SAVE		9					
COND		8					

STATIC ANALYSIS WITH INERTIA RELIEF

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
\$\$\$		7					
PLOT			8				
\$\$\$		7					
SAVE			8				
\$\$\$		7					
PRMSG			8				
\$\$\$		7					
LABEL			8				
\$\$\$		7					
JUMP	1234567890123456789		234				9012
LABEL	1234567890123456789		234				9012
PRTPARM	1234567890123456789		234				9012
LABEL	1234567890123456789		234				9012
\$\$\$	1 3						
PRTPARM	1234567890123456789		234				9012
\$\$\$	1 3						
LABEL	1234567890123456789		234				9012
PRTPARM	1234567890123456789		234				9012
LABEL	1234567890123456789		234				9012
PRTPARM	1234567890123456789		234				9012
LABEL	1234567890123456789		234				9012
PRTPARM	1234567890123456789		234				9012
LABEL	1234567890123456789		234				9012
END	1234567890123456789		234				9012

RIGID FORMAT RESTART TABLES

7.3.4 Rigid Format Change Restart Table

DMAP Inst.	63	Bit Position 70	80	DMAP Inst.	63	Bit Position 70	80
BEGIN	3	5678901234567	345	COND			
FILE	3	5678901234567	345	OFF			
GP1				LABEL			
SAVE				EQUIV			
CHKPNT				CHKPNT			
GP2				COND			
CHKPNT				MCE1			
PARAML				CHKPNT			
PURGE				MCE2			
COND				CHKPNT			
PLTSET				LABEL			
SAVE				EQUIV			
PRTMSG				CHKPNT			
PARAM				COND			
PARAM				SCE1			
COND				CHKPNT			
PLOT				LABEL			
SAVE				EQUIV			
PRTMSG				CHKPNT			
LABEL				COND			
CHKPNT				SMP1			
GP3				CHKPNT			
CHKPNT				LABEL			
TA1				RBMG1			
SAVE				CHKPNT			
COND				RBMG2			
PURGE				CHKPNT			
CHKPNT				RBMG3			
PARAM				CHKPNT			
PARAM	3	678		RBMG4			
EMG	3	678		CHKPNT			
SAVE	3	678		SSG1			
CHKPNT	3	678		CHKPNT			
COND				SSG2			
EMA				CHKPNT			
CHKPNT				SSG4			
LABEL				CHKPNT			
COND	3	5678901234567	345	SSG3	3		
EMA	3	678		SAVE	3		
CHKPNT	3	678		CHKPNT	3		
COND				COND	3	5678901234567	345
GPWG				MATGPR	3	5678901234567	345
OFF				MATGPR	3	5678901234567	345
LABEL				LABEL	3	5678901234567	345
EQUIV				SDR1	3	5678901234567	345
CHKPNT				CHKPNT	3	5678901234567	345
COND				COND	3	5678901234567	345
SMA3				REPT	3	5678901234567	345
CHKPNT				JUMP	3	5678901234567	345
LABEL				PARAM	3	5678901234567	345
PARAM				COND	3	5678901234567	345
JUMP				LABEL	3	5678901234567	345
LABEL				CHKPNT			
GP4				COND	3	5678901234567	345
SAVE				EQMCK	3	5678901234567	345
COND	3	5678901234567	345	OFF	3	5678901234567	345
COND	3	5678901234567	345	SAVE	3	5678901234567	345

STATIC ANALYSIS WITH INERTIA RELIEF

DMAP		Bit Position	
Inst.	63	70	80
LABEL	3	5678901234567	345
SDR2			
PARAM			
OFF			
SAVE			
COND			
PLOT			
SAVE			
PRTMSG			
LABEL			
JUMP	3	5678901234567	345
LABEL	3	5678901234567	345
PRTPARM	3	5678901234567	345
LABEL	3	5678901234567	345
PRTPARM	3	5678901234567	345
LABEL	3	5678901234567	345
PRTPARM	3	5678901234567	345
LABEL	3	5678901234567	345
PRTPARM	3	5678901234567	345
LABEL	3	5678901234567	345
PRTPARM	3	5678901234567	345
LABEL	3	5678901234567	345
PRTPARM	3	5678901234567	345
LABEL	3	5678901234567	345
END	3	5678901234567	345

RIGID FORMAT RESTART TABLES

7.3.5 File Name Restart Table

DMAP		Bit Position		DMAP		Bit Position	
Inst.	94	100	110	Inst.	94	100	110
BEGIN				PURGE		1 3 56	2 4
FILE				CHKPNT		1 3 56	2 4
GP1	4			COND		2	
SAVE	4			GPSP		2	
CHKPNT	4			SAVE		2	
GP2	5			COND		2	
CHKPNT	5			OFF		2	
PARAML			7	LABEL		2	
PURGE			7	EQUIV		4	
COND			7	CHKPNT		4	
PLTSET			7	COND		34	
SAVE			7	MCE1		3	
PRTMSG			7	CHKPNT		3	
PARAM			7	MCE2		4	
PARAM			7	CHKPNT		4	
COND				LABEL		34	
PLOT				EQUIV		5	
SAVE				CHKPNT		5	
PRTMSG				COND		5	
LABEL				SCE1		5	
CHKPNT			7	CHKPNT		5	
GP3	6			LABEL		5	
CHKPNT	6			EQUIV		6	
TA1	7			CHKPNT		6	
SAVE	7			COND		6	
COND	7 9			SMP1		6	
PURGE	7 2			CHKPNT		6	
CHKPNT	7			LABEL		6	
PARAM	8			RBMG1		7	
PARAM	9			CHKPNT		7	
EMG			8	RBMG2		8	
SAVE			8	CHKPNT		8	
CHKPNT			8	RBMG3		9	
COND	8			CHKPNT		9	
EMA	8			RBMG4		0	
CHKPNT	8			CHKPNT		0	
LABEL	8			SSG1		1	
COND	9			CHKPNT		1	
EMA	9			SSG2		2	
CHKPNT	9			CHKPNT		2	
COND				SSG4		3	
GPWG				CHKPNT		3	
OFF				SSG3		4	
LABEL				SAVE		4	
EQUIV	0			CHKPNT		4	
CHKPNT	0			COND			
COND	0			MATGPR			
SMA3	0			MATGPR			
CHKPNT	0			LABEL			
LABEL	0			SDR1			5
PARAM	1			CHKPNT			5
JUMP				COND			
LABEL	0			REPT			
GP4	1			JUMP			
SAVE	1			PARAM			
COND	1			COND			
COND	1			LABEL			

STATIC ANALYSIS WITH INERTIA RELIEF

DMAP		Bit Position		
Inst.	94	100	110	120
CHKPNT				
COND				
EQMCK				
OFF				
SAVE				
LABEL				
SDR2				
PARAM				6
OFF				
SAVE				
COND				
PLOT				
SAVE				
PRTMSG				
LABEL				
JUMP				
LABEL				
PRTPARM				
LABEL				
PRTPARM				
LABEL				
PRTPARM				
LABEL				
PRTPARM				
LABEL				
PRTPARM				
LABEL				
PRTPARM				
LABEL				
END				

NORMAL MODES ANALYSIS

7.4 RESTART TABLES FOR NORMAL MODES ANALYSIS

7.4.1 Bit Positions for Card Name Restart Table

Card Name	Bit Pos.	Card Name	Bit Pos.	Card Name	Bit Pos.
AXIC	1	CIHEX1	2	PROD	3
AXIF	1	CIHEX2	2	PSHEAR	3
AXSLOT	1	CIHEX3	2	PTORDRG	3
CELAS1	1	CFLUID2	2	PTRAPAX	3
CELAS2	1	CFLUID3	2	PTRBSC	3
CELAS3	1	CFLUID4	2	PTRIA1	3
CELAS4	1	CONROD	2	PTRIA2	3
CMASS1	1	CQDMEM	2	PTRIAAX	3
CMASS2	1	CQDMEM1	2	PTRIATS	3
CMASS3	1	CQDMEM2	2	PTRIM6	3
CMASS4	1	CQDMEM3	2	PTRMEM	3
CORD1C	1	CQDPLT	2	PTRPLT	3
CORD1R	1	CQUAD1	2	PTRPLT1	3
CORD1S	1	CQUAD2	2	PTRSHL	3
CORD2C	1	CQUADTS	2	PTUBE	3
CORD2R	1	CROD	2	PTWIST	3
CORD2S	1	CSHEAR	2	GENEL	4
FREETPT	1	CSLOT3	2	CONM1	5
GRDSET	1	CSLOT4	2	CONM2	5
GRID	1	CTETRA	2	FSLIST	5
GRIDB	1	CTORDRG	2	PELAS	6
GRIDF	1	CTRAPAX	2	PMASS	7
GRIDS	1	CTRAPRG	2	MAT1	8
POINTAX	1	CTRBSC	2	MAT2	8
PRESPT	1	CTRIA1	2	MAT3	8
RINGAX	1	CTRIA2	2	MATT1	8
RINGFL	1	CTRIAAX	2	MATT2	8
SECTAX	1	CTRIARG	2	MATT3	8
SEQGP	1	CTRIATS	2	TABLEM1	8
SLBDY	1	CTRIM6	2	TABLEM2	8
SPOINT	1	CTRMEM	2	TABLEM3	8
ADUM1	2	CTRPLT	2	TABLEM4	8
ADUM2	2	CTRPLT1	2	TEMPMT\$	8
ADUM3	2	CTRSHL	2	TEMPMX\$	8
ADUM4	2	CTUBE	2	AXISYM\$	9
ADUM5	2	CTWIST	2	CRIGD1	9
ADUM6	2	CWEDGE	2	CRIGD2	9
ADUM7	2	PBAR	3	CRIGD3	9
ADUM8	2	PCONEAX	3	CRIGDR	9
ADUM9	2	PDUM1	3	MPC	9
BAROR	2	PDUM2	3	MPCADD	9
CAXIF2	2	PDUM3	3	MPCAX	9
CAXIF3	2	PDUM4	3	MPC\$	9
CAXIF4	2	PDUM5	3	SPC	10
CBAR	2	PDUM6	3	SPC1	10
CCONEAX	2	PDUM7	3	SPCADD	10
CDUM1	2	PDUM8	3	SPCAX	10
CDUM2	2	PDUM9	3	SPC\$	10
CDUM3	2	PIHEX	3	ASET	11
CDUM4	2	PQDMEM	3	ASET1	11
CDUM5	2	PQDMEM1	3	OMIT	11
CDUM6	2	PQDMEM2	3	OMIT1	11
CDUM7	2	PQDMEM3	3	OMITAX	11
CDUM8	2	PQDPLT	3	SUPAX	12
CDUM9	2	PQUAD1	3	SUPORT	12
CHEXA1	2	PQUAD2	3	TEMP	13
CHEXA2	2	PQUADTS	3	TEMPAX	13

RIGID FORMAT RESTART TABLES

<u>Card Name</u>	<u>Bit Pos.</u>
------------------	-----------------

TEMPO	13
TEMPP1	13
TEMPP2	13
TEMPP3	13
TEMPRB	13
WTMASS	14
GRDPNT	15
PLOTEL	16
PLOTS	18
POUTS	19
COUPMASS	24
CPBAR	24
CPQDPLT	24
CPQUAD1	24
CPQUAD2	24
CPROD	24
CPTRBSC	24
CPTRIA1	24
CPTRIA2	24
CPTRPLT	24
CPTUBE	24
EIGR	61
METHODS	62

NORMAL MODE ANALYSIS

7.4.2 Bit Positions for File Name Restart Table

<u>File Name</u>	<u>Bit Pos.</u>	<u>File Name</u>	<u>Bit Pos.</u>
BGPD T	94	MRR	107
CSTM	94	LLL	108
EQEXIN	94	DM	109
GPD T	94	MR	110
GPL	94	EED	111
SIL	94	EQDYN	111
ECT	95	GPLD	111
GPTT	96	SILD	111
EST	97	USETD	111
GEI	97	LAMA	112
GPECT	97	MI	112
GPST	98	DEIGS	112
KGGX	98	PHIA	112
MGG	99	PHIG	113
KGG	100	QG	113
ASET	101	DEF1	114
RG	101	OES1	114
USET	101	OPHIG	114
OGPST	102	OQG1	114
GM	103	PPHIG	114
KNN	104	BGPD P	115
MNN	104	SIP	115
KFF	105	ELSETS	116
KFS	105	GPSETS	116
MFF	105	PLTPAR	116
GO	106	PLTSETX	116
KAA	106	MAA	117
KLL	107	KDICT	118
KLR	107	KELM	118
KRR	107	MDICT	118
MLL	107	MELM	118
MLR	107		

RIGID FORMAT RESTART TABLES

7.4.3 Card Name Restart Table

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
BEGIN	1234567890123456	89	4				12
FILE	1234567890123456	89	4				12
GP1	1						
SAVE	1						
CHKPNT	1						
\$\$\$	6						
GP2	12 45	6					
CHKPNT	12 45	6					
\$\$\$	6						
PARAML			8				
\$\$\$	7						
PURGE			8				
\$\$\$	7						
COND			8				
\$\$\$	7						
PLTSET			8				
\$\$\$	7						
SAVE			8				
\$\$\$	7						
PRTMSG			8				
\$\$\$	7						
PARAM			8				
\$\$\$	7						
PARAM			8				
\$\$\$	7						
COND			8				
\$\$\$	7						
PLOT			8				
\$\$\$	7						
SAVE			8				
\$\$\$	7						
PRTMSG			8				
\$\$\$	7						
LABEL			8				
\$\$\$	7						
CHKPNT			8				
\$\$\$	67						
GP3	1	3					
CHKPNT	1	3					
\$\$\$	6						
TA1	1234567	3					
SAVE	1234567	3					
COND	12345678	34	4				
PURGE	1234567	3					
CHKPNT	1234567	3					
\$\$\$	6						
PARAM	123 6 8						
PARAM	123 5 7 8						
EMG	123 5678	4	4				
SAVE	123 5678	4	4				
CHKPNT	123 5678	4	4				
\$\$\$	6						
COND	123 6 8						
EMA	123 6 8						
CHKPNT	123 6 8						
\$\$\$	6						
LABEL	123 6 8						
COND	123 5 7 8	4	4				
EMA	123 5 7 8	4	4				

NORMAL MODE ANALYSIS

DMAP Inst.	1	10	20	<u>Bit Position</u> 30	40	50	60
CHKPNT	123 5 7 8	4		4			
\$\$\$	6						
COND	123 5 7 8	45		4			
\$\$\$	8						
GPWG	123 5 7 8	45		4			
\$\$\$	8						
OFF	123 5 7 8	45		4			
\$\$\$	8						
LABEL	123 5 7 8	45		4			
\$\$\$	8						
EQUIV	1234 6 8						
CHKPNT	1234 6 8						
\$\$\$	6						
COND	1234 6 8						
SMA3	1234 6 8						
CHKPNT	1234 6 8						
\$\$\$	6						
LABEL	1234 6 8						
PARAM	1	9012					
GP4	1	9012					
SAVE	1	9012					
COND	1	9012					
PURGE	1	9012					
CHKPNT	123456789012 4			4			
\$\$\$	6						
COND	123 6 890						
GPSP	123 6 890						
SAVE	123 6 890						
COND	123 6 890						
OFF	123 6 890						
LABEL	123 6 890						
EQUIV	123456789	4		4			
CHKPNT	123456789	4		4			
\$\$\$	6						
COND	123456789	4		4			
MCE1	1 9						
CHKPNT	1 9						
\$\$\$	6						
MCE2	123456789	4		4			
CHKPNT	123456789	4		4			
\$\$\$	6						
LABEL	123456789	4		4			
EQUIV	1234567890	4		4			
CHKPNT	1234567890	4		4			
\$\$\$	6						
COND	1234567890	4		4			
SCE1	1234567890	4		4			
CHKPNT	1234567890	4		4			
\$\$\$	6						
LABEL	1234567890	4		4			
EQUIV	1234 6 8901						
EQUIV	12345678901	4		4			
CHKPNT	12345678901	4		4			
\$\$\$	6						
COND	12345678901	4		4			
SMP1	1234 6 8901						
CHKPNT	1234 6 8901						
\$\$\$	6						

RIGID FORMAT RESTART TABLES

DMAP Inst.	Bit Position						
	1	10	20	30	40	50	60
SMP2	12345678901	4		4			
CHKPNT	12345678901	4		4			
\$SS	6						
LABEL	12345678901	4		4			
COND	123456789012	4		4			
RBMG1	123456789012	4		4			
CHKPNT	123456789012	4		4			
\$SS	6						
RBMG2	1234 6 89012						
CHKPNT	1234 6 89012						
\$SS	6						
RBMG3	1234 6 89012						
CHKPNT	1234 6 89012						
\$SS	6						
RBMG4	123456789012	4		4			
CHKPNT	123456789012	4		4			
\$SS	6						
LABEL	123456789012	4		4			
DPD	1 9012						1
SAVE	1 9012						1
COND	1 9012						1
CHKPNT	1 9012						1
\$SS	6						
PARAM	12345678901234		4				
READ	123456789012	4	4				12
SAVE	123456789012	4	4				12
CHKPNT	123456789012	4	4				12
\$SS	6						
PARAM			9				
OFF	123456789012	4	4				12
SAVE	123456789012	4	4				12
COND	123456789012	4	4				12
OFF	123456789012	4	4				12
SAVE	123456789012	4	4				12
SDR1	123456789012	4	4				12
CHKPNT	123456789012	4	4				12
COND	123456789012	4	4				12
EQMCK	123456789012	4	4				12
OFF	123456789012	4	4				12
SAVE	123456789012	4	4				12
LABEL	123456789012	4	4				12
\$SS	6						
PARAM			89				
PARAM			89				
EQUIV			89				
CHKPNT			89				
\$SS	6						
COND			89				
PLTTRAN	1		8				
SAVE	1		8				
CHKPNT	1		8				
\$SS	6						
LABEL			89				
SDR2			89				
OFF			9				
SAVE			9				
COND			8				
\$SS	7						

NORMAL MODE ANALYSIS

DMAP	Bit Position						
Inst.	1	10	20	30	40	50	60
PLOT			8				
\$\$\$	7						
SAVE			8				
\$\$\$	7						
PRTMSG			8				
\$\$\$	7						
LABEL			8				
\$\$\$	7						
JUMP	1234567890123456	89	4				12
LABEL	1234567890123456	89	4				12
PRTPARM	1234567890123456	89	4				12
LABEL	1234567890123456	89	4				12
PRTPARM	1234567890123456	89	4				12
LABEL	1234567890123456	89	4				12
PRTPARM	1234567890123456	89	4				12
LABEL	1234567890123456	89	4				12
END	1234567890123456	89	4				12

RIGID FORMAT RESTART TABLES

7.4.4 Rigid Format Change Restart Table

DMAP Inst.	63	Bit Position 70	80	DMAP Inst.	63	Bit Position 70	80
BEGIN	34	678901234567	345	SAVE			
FILE	34	678901234567	345	COND			
GP1				OFF			
SAVE				LABEL			
CHKPNT				EQUIV			
GP2				CHKPNT			
CHKPNT				COND			
PARAML				MCE1			
PURGE				CHKPNT			
COND				MCE2			
PLTSET				CHKPNT			
SAVE				LABEL			
PRTMSG				EQUIV			
PARAM				CHKPNT			
PAFAM				COND			
COND				SCE1			
PLOT				CHKPNT			
SAVE				LABEL			
PRTMSG				EQUIV			
LABEL				EQUIV			
CHKPNT				CHKPNT			
GP3				COND			
CHKPNT				SMP1			
TA1				CHKPNT			
SAVE				SMP2			
COND				CHKPNT			
PURGE				LABEL			
CHKPNT				COND			
PARAM				RBMG1			
PARAM	3	678		CHKPNT			
EMG	3	678		RBMG2			
SAVE	3	678		CHKPNT			
CHKPNT	3	678		RBMG3			
COND				CHKPNT			
EMA				RBMG4			
CHKPNT				CHKPNT			
LABEL				LABEL			
COND	34	678901234567	345	DPD			
EMA	3	678		SAVE			
CHKPNT	3	678		COND	34	678901234567	345
COND				CHKPNT			
GPWG				PARAM			
OFF				READ			
LABEL				SAVE			
EQUIV				CHKPNT			
CHKPNT				PARAM			
COND				OFF			
SMA3				SAVE			
CHKPNT				COND			
LABEL				OFF			
PARAM				SAVE			
GP4				SDR1	34	678901234567	345
SAVE				CHKPNT	34	678901234567	345
COND				COND	34	678901234567	345
PURGE				EQMCK	34	678901234567	345
CHKPNT				OFF	34	678901234567	345
COND				SAVE	34	678901234567	345
GPSP				LABEL	34	678901234567	345

NORMAL MODES ANALYSIS

DMAP	<u>Bit Position</u>		
Inst.	63	70	80
PARAM			
PARAM			
EQUIV			
CHKPNT			
COND			
PLTTRAN			
SAVE			
CHKPNT			
LABEL			
SDR2			
DFP			
SAVE			
COND			
PLOT			
SAVE			
PRTMSG			
LABEL			
JUMP	34	678901234567	345
LABEL	34	678901234567	345
PRTPARM	34	678901234567	345
LABEL	34	678901234567	345
PRTPARM	34	678901234567	345
LABEL	34	678901234567	345
PRTPARM	34	678901234567	345
LABEL	34	678901234567	345
END	34	678901234567	345

RIGID FORMAT RESTART TABLES

7.4.5 File Name Restart Table

DMAP Inst.	94	100	110	120	DMAP Inst.	94	100	110	120
BEGIN					GPSP		2		
FILE					SAVE		2		
GP1	4				COND		2		
SAVE	4				OFF		2		
CHKPNT	4				LABEL		2		
GP2	5				EQUIV		4		
CHKPNT	5				CHKPNT		4		
PARAML				6	COND		34		
PURGE				6	MCE1		3		
COND				6	CHKPNT		3		
PLTSET				6	MCE2		4		
SAVE				6	CHKPNT		4		
PRTMSG				6	LABEL		34		
PARAM				6	EQUIV		5		
PARAM				6	CHKPNT		5		
COND					COND		5		
PLOT					SCE1		5		
SAVE					CHKPNT		5		
PRTMSG					LABEL		5		
LABEL					EQUIV		6		
CHKPNT				6	EQUIV				7
GP3	6				CHKPNT		6		7
CHKPNT	6				COND		6		7
TA1	7				SMP1		6		
SAVE	7				CHKPNT		6		
COND	7				SMP2				7
PURGE	7	2			CHKPNT				7
CHKPNT	7				LABEL		6		7
PARAM	8				COND		7890		
PARAM	9				RBMG1		7		
EMG				8	CHKPNT		7		
SAVE				8	RBMG2		8		
CHKPNT				8	CHKPNT		8		
COND	8				RBMG3		9		
EMA	8				CHKPNT		9		
CHKPNT	8				RBMG4		0		
LABEL	8				CHKPNT		0		
COND	9				LABEL		7890		
EMA	9				DPD		1		
CHKPNT	9				SAVE		1		
COND					COND		1		
GPWG					CHKPNT		1		
OFF					PARAM		2		
LABEL					READ		2		
EQUIV	0				SAVE		2		
CHKPNT	0				CHKPNT		2		
COND	0				PARAM		2		
SMA3	0				OFF		2		
CHKPNT	0				SAVE		2		
LABEL	0				COND		34		
PARAM	1				OFF		2		
GP4	1				SAVE		2		
SAVE	1				SDR1		3		
COND	1				CHKPNT		3		
PURGE	1 3 567 90	3			COND		3		
CHKPNT	1 3 567 90	3			EQMCK		3		
COND	2				OFF		3		

NORMAL MODE ANALYSIS

DMAP		Bit Position		
Inst.	94	100	110	120
SAVE			3	
LABEL			3	
PAPAM			5	
PARAM			5	
EQUIV			5	
CHKPNT			5	
COND			5	
PLTTRAN			5	
SAVE			5	
CHKPNT			5	
LABEL			5	
SDR2			4	
OFP				
SAVE				
COND				
PLOT				
SAVE				
PRMSG				
LABEL				
JUMP				
LABEL				
PRTPARM				
LABEL				
PRTPARM				
LABEL				
PRTPARM				
LABEL				
END				

STATIC ANALYSIS WITH DIFFERENTIAL STIFFNESS

7.5 RESTART TABLES FOR STATIC ANALYSIS WITH DIFFERENTIAL STIFFNESS

7.5.1 Bit Positions for Card Name Restart Table

Card Name	Bit Pos.	Card Name	Bit Pos.	Card Name	Bit Pos.
AXIC	1	CQOPLT	2	PTRSHL	3
AXIF	1	CQUAD1	2	PTUBE	3
CELAS1	1	CQUAD2	2	PTWIST	3
CELAS2	1	CQUADTS	2	GENEL	4
CELAS3	1	CROD	2	CONM1	5
CELAS4	1	CSHEAR	2	CONM2	5
CMASS1	1	CTETRA	2	PELAS	6
CMASS2	1	CTORDRG	2	PMASS	7
CMASS3	1	CTRAPAX	2	MAT1	8
CMASS4	1	CTRAPRG	2	MAT2	8
CORD1C	1	CTRBSC	2	MAT3	8
CORD1R	1	CTRIA1	2	MATT1	8
CORD1S	1	CTRIA2	2	MATT2	8
CORD2C	1	CTRIAAX	2	MATT3	8
CORD2R	1	CTRIARG	2	TABLEM1	8
CORD2S	1	CTRIATS	2	TABLEM2	8
GRDSET	1	CTRM6	2	TABLEM3	8
GRID	1	CTRMEM	2	TABLEM4	8
GRIDB	1	CTRPLT	2	TEMPMT\$	8
POINTAX	1	CTRPLT1	2	TEMPMX\$	8
RINGAX	1	CTRSHL	2	AXISYM\$	9
RINGFL	1	CTUBE	2	CRIGD1	9
SECTAX	1	CTWIST	2	CRIGD2	9
SEQGP	1	CHEDGE	2	CRIGD3	9
SPOINT	1	PBAR	3	CRIGDR	9
ADUM1	2	PCONEAX	3	MPC	9
ADUM2	2	PDUM1	3	MPCADD	9
ADUM3	2	PDUM2	3	MPCAX	9
ADUM4	2	PDUM3	3	MPC\$	9
ADUM5	2	PDUM4	3	SPC	10
ADUM6	2	PDUM5	3	SPC1	10
ADUM7	2	PDUM6	3	SPCADD	10
ADUM8	2	PDUM7	3	SPCAX	10
ADUM9	2	PDUM8	3	SPC\$	10
BAROR	2	PDUM9	3	ASET	11
CBAR	2	PIHEX	3	ASET1	11
CCONEAX	2	PQDMEM	3	OMIT	11
CDUM1	2	PQDMEM1	3	OMIT1	11
CDUM2	2	PQDMEM2	3	OMITAX	11
CDUM3	2	PQDMEM3	3	SUPAX	12
CDUM4	2	PQOPLT	3	SUPORT	12
CDUM5	2	PQUAD1	3	TEMP	13
CDUM6	2	PQUAD2	3	TEMPAX	13
CDUM7	2	PQUADTS	3	TEMPO	13
CDUM8	2	PROD	3	TEMPP1	13
CDUM9	2	PSHEAR	3	TEMPP2	13
CHEXA1	2	PTORDRG	3	TEMPP3	13
CHEXA2	2	PTRAPAX	3	TEMPRB	13
CIHEX1	2	PTRBSC	3	WTHASS	14
CIHEX2	2	PTRIA1	3	GRDPNT	15
CIHEX3	2	PTRIA2	3	PLOTEL	16
CONROD	2	PTRIAAX	3	IRES	17
CQDMEM	2	PTRIATS	3	PLOT\$	18
CQDMEM1	2	PTRIM6	3	POUT\$	19
CQDMEM2	2	PTRMEM	3	LOOP\$	22
CQDMEM3	2	PTRPLT	3	LOOP1\$	23
		PTRPLT1	3	COUPHASS	24

RIGID FORMAT RESTART TABLES

<u>Card Name</u>	<u>Bit Pos.</u>
CPBAR	24
CPQDPLT	24
CPQUAD1	24
CPQUAD2	24
CPROD	24
CPTRBS3	24
CPTRIA1	24
CPTRIA2	24
CPTRPLT	24
CPTUBE	24
DEFORM	59
DEFORM\$	59
LOAD\$	59
RFORCE\$	59
SPCD	59
FORCE	60
FORCE1	60
FORCE2	60
FORCEAX	60
LOAD	60
MOMAX	60
MOMENT	60
MOMENT1	60
MOMENT2	60
PLOAD	60
PLOAD1	60
PLOAD2	60
PLOAD3	60
PRESAX	60
SLOAD	60
GRAV	61
RFORCE	61
TEMPLO\$	62

STATIC ANALYSIS WITH DIFFERENTIAL STIFFNESS

7.5.2 Bit Positions for File Name Restart Table

<u>File Name</u>	<u>Bit Pos.</u>	<u>File Name</u>	<u>Bit Pos.</u>
BGPD T	94	PGG	111
CSTM	94	QG	111
EQEXIN	94	UGV	111
GPD T	94	OEF1	112
GPL	94	OES1	112
SIL	94	OPG1	112
ECT	95	OQG1	112
GPTT	96	OUGV1	112
SLT	96	PUGV1	112
EST	97	KDDICT	113
GEI	97	KDELM	113
GPECT	97	KDGG	113
GPST	98	KDNN	114
KGGX	98	KDFF	115
MGG	99	KDFS	115
KGG	100	KDSS	115
ASET	101	KDAA	116
RG	101	KBLL	117
USET	101	KBFS	117
YS	101	KBSS	117
OGPST	102	PBL	117
GM	103	PBS	117
KNN	104	YBS	117
KFF	105	LBLL	118
KFS	105	UBLV	119
KSS	105	RUBLV	119
GO	106	QBG	120
KAA	106	UBGV	120
KOO	106	OEFB1	121
LOO	106	OESB1	121
LLL	107	OQBG1	121
PG	108	OUBGV1	121
PL	109	PUBGV1	121
PO	109	ELSETS	122
PS	109	GPSETS	122
RULV	110	PLTPAR	122
RUOV	110	PLTSETX	122
ULV	110	KDICT	123
UOOV	110	KELM	123
		MDICT	123
		MELM	123
		CASEXX	124

RIGID FORMAT RESTART TABLES

7.5.3 Card Name Restart Table

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
BEGIN	1234567890123456789	234					9012
GP1	1						
SAVE	1						
COND	1						
CHKPNT	1						
\$SS		6					
GP2	12 45		6				
CHKPNT	12 45		6				
\$SS		6					
PARAML			8				
\$SS		7					
PURGE			8				
\$SS		7					
COND			8				
\$SS		7					
PLTSET			8				
\$SS		7					
SAVE			8				
\$SS		7					
PRTMSG			8				
\$SS		7					
PARAM			8				
\$SS		7					
PARAM			8				
\$SS		7					
COND			8				
\$SS		7					
PLOT			8				
\$SS		7					
SAVE			8				
\$SS		7					
PRTMSG			8				
\$SS		7					
LABEL			8				
\$SS		7					
CHKPNT			8				
\$SS		67					
GP3	12		3				01
SAVE	12		3				01
PARAM	12		3 5				01
CHKPNT	12		3				01
\$SS		6					
TA1	1234567		3				
SAVE	1234567		3				
COND	12345678		3				
PURGE	1234567		3				
CHKPNT	1234567		3				
\$SS		6					
PARAM	1234 6						
EMG	12345678						
SAVE	12345678						
CHKPNT	12345678						
\$SS		6					
COND	1234 6 8						
EMA	1234 6 8						
CHKPNT	1234 6 8						
\$SS		6					
LABEL	12345 78	4	4				

STATIC ANALYSIS WITH DIFFERENTIAL STIFFNESS

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
COND	12345 78	4	4				
EMA	12345 78	4	4				
CHKPNT	12345 78	4	4				
\$\$\$	6						
LABEL	12345 78	4	4				
COND	123 5 78	45	4				
\$\$\$	8						
COND	123 5 78	45	4				
\$\$\$	8						
GPWG	123 5 78	45	4				
\$\$\$	8						
OFFP	123 5 78	45	4				
\$\$\$	8						
LABEL	123 5 78	45	4				1
\$\$\$	8						
EQUIV	1234 6 8						
CHKPNT	1234 6 8						
\$\$\$	6						
COND	1234 6 8						
SMA3	1234 6 8						
CHKPNT	1234 6 8						
\$\$\$	6						
LABEL	1234 6 8						
PARAM	1	901					
CASE	1	901					
GP4	1	901					9
SAVE	1	901					3
COND	1	901					9
PURGE	1	901					9
CHKPNT	1234 6 8901						9
\$\$\$	6						
COND	1	2					
JUMP	1	2					
LABEL	1	2					
COND	123 6 890						
GPSP	123 6 890						
SAVE	123 6 890						
COND	123 6 890						
OFFP	123 6 890						
LABEL	123 6 890						
EQUIV	1234 6 89						
CHKPNT	1234 6 89						
\$\$\$	6						
COND	1234 6 89						
MCE1	1	9					
CHKPNT	1	9					
\$\$\$	6						
MCE2	1234 6 89						
CHKPNT	1234 6 89						
\$\$\$	6						
LABEL	1234 6 89						
EQUIV	1234 6 890						
CHKPNT	1234 6 890						
\$\$\$	6						
COND	1234 6 890						
SCE1	1234 6 890						
CHKPNT	1234 6 890						
\$\$\$	6						
LABEL	1234 6 890						

RIGID FORMAT RESTART TABLES

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
EQUIV	1234	6 8901					
CHKPNT	1234	6 8901					
SSS		6					
COND	1234	6 8901					
SMP1	1234	6 8901					
CHKPNT	1234	6 8901					
SSS		6					
LABEL	1234	6 8901					
RBMG2	1234	6 8901					
CHKPNT	1234	6 8901					
SSS		6					
SSG1	123	5678	3				9012
CHKPNT	123	5678	3				9012
SSS		6					
EQUIV	123	5678901	3				9012
CHKPNT	123	5678901	3				9012
SSS		6					
COND	123	5678901	3				9012
SSG2	123	5678901	3				9012
CHKPNT	123	5678901	3				9012
SSS		6					
LABEL	123	5678901	3				9012
SSG3	1234	5678901	3				9012
SAVE	1234	5678901	3				9012
CHKPNT	1234	5678901	3				9012
SSS		6					
COND	1234	5678901	3 7				9012
MATGPR	1234	5678901	3 7				9012
MATGPR	1234	5678901	3 7				9012
LABEL	1234	5678901	3 7				9012
SDR1	1234	5678901	3				9012
CHKPNT	1234	5678901	3				9012
SSS		6					
SDR2			9				
PARAM			9				
OFF			9				
SAVE			9				
COND			8				
SSS		7					
PLOT			8				
SSS		7					
SAVE			8				
SSS		7					
PRTMSG			8				
SSS		7					
LABEL			8				
SSS		7					
TA1	1234	5678901					9012
DSMG1	1234	5678901					9012
CHKPNT	1234	5678901					9012
SSS		6					
PARAM	1234	5678901					9012
PARAM	1234	5678901					9012
PARAMR	1234	5678901					9012
PARAML	1234	5678901					9012
JUMP	1234	5678901					9012
LABEL	1234	5678901					9012
EQUIV	1234	5678901					9012
CHKPNT	1234	5678901					9012

STATIC ANALYSIS WITH DIFFERENTIAL STIFFNESS

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
SSS	6						
PARAM	12345678901						9012
EQUIV	12345678901						9012
CHKPNT	12345678901						9012
SSS	6						
COND	12345678901						9012
MCE2	12345678901						9012
CHKPNT	12345678901						9012
SSS	6						
LABEL	12345678901						9012
EQUIV	12345678901						9012
CHKPNT	12345678901						9012
SSS	6						
COND	12345678901						9012
SCE1	12345678901						9012
CHKPNT	12345678901						9012
SSS	6						
LABEL	12345678901						9012
EQUIV	12345678901						9012
CHKPNT	12345678901						9012
SSS	6						
COND	12345678901						9012
SMP2	12345678901						9012
CHKPNT	12345678901						9012
SSS	6						
LABEL	12345678901						9012
ADD	12345678901						9012
ADD	12345678901						9012
ADD	12345678901						9012
COND	12345678901						9012
MPYAD	12345678901						9012
MPYAD	12345678901						9012
UMERGE	12345678901						9012
EQUIV	12345678901						9012
COND	12345678901						9012
UMERGE	12345678901						9012
LABEL	12345678901						9012
ADD	12345678901						9012
EQUIV	12345678901						9012
LABEL	12345678901						9012
ADD	12345678901						9012
REMG2	12345678901		23				9012
SAVE	12345678901		23				9012
CHKPNT	12345678901		23				9012
SSS	6						
PRTPARM	12345678901		23				9012
PRTPARM	12345678901		23				9012
JUMP	12345678901		23				9012
LABEL	12345678901		23				9012
PARAM	12345678901		23				9012
SSG2	12345678901		23				9012
SSG3	12345678901		23				9012
SAVE	12345678901		23				9012
CHKPNT	12345678901		23				9012
SSS	6						
COND	12345678901	7	23				9012
MATGPR	12345678901	7	23				9012
LABEL	12345678901	7	23				9012
SCR1	12345678901		23				9012

RIGID FORMAT RESTART TABLES

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
CHKPNT	12345678901		23				9012
\$\$\$	6						
ADD	12345678901		23				9012
DSMG1	12345678901		23				9012
CHKPNT	12345678901		23				9012
\$\$\$	6						
MPYAD	12345678901		23				9012
DSCHK	12345678901		23				9012
SAVE	12345678901		23				9012
COND	12345678901		23				9012
COND	12345678901		23				9012
EQUIV	12345678901		23				9012
EQUIV	12345678901		23				9012
EQUIV	12345678901		23				9012
REPT	12345678901		23				9012
TABPT	12345678901		23				9012
LABEL	12345678901		23				9012
ADD	12345678901		23				9012
CHKPNT	12345678901		23				9012
\$\$\$	6						
EQUIV	12345678901		23				9012
CHKPNT	12345678901		23				9012
\$\$\$	6						
EQUIV	12345678901		23				9012
REPT	12345678901		23				9012
TABPT	12345678901		23				9012
LABEL	12345678901		23				9012
CHKPNT			9				
SDR2			99				
OFF			9				
SAVE			9				
COND			8				
\$\$\$	7						
PLOT			8				
\$\$\$	7						
SAVE			8				
\$\$\$	7						
PRTMSG			8				
\$\$\$	7						
LABEL			8				
\$\$\$	7						
JUMP	1234567890123456789		234				9012
LABEL	1234567890123456789		234				9012
PRTPARM	1234567890123456789		234				9012
LABEL	1234567890123456789		234				9012
PPTPARM	1234567890123456789		234				9012
LABEL	1234567890123456789		234				9012
\$\$\$	8						
PRTPARM	1234567890123456789		234				9012
\$\$\$	8						
LABEL	1234567890123456789		234				9012
PRTPARM	1234567890123456789		234				9012
LABEL	1234567890123456789		234				9012
END	1234567890123456789		234				9012

STATIC ANALYSIS WITH DIFFERENTIAL STIFFNESS

7.5.4 Rigid Format Change Restart Table

DMAP Inst.	63	Bit Position 70	80	DMAP Inst.	63	Bit Position 70	80
BEGIN		345 78901234567	345	PURGE			
GP1				CHKPNT			
SAVE				COND	345	901234567	345
COND				JUMP	345	901234567	345
CHKPNT				LABEL	345	901234567	345
GP2				COND			
CHKPNT				GPSP			
PARAML				SAVE			
PURGE				COND			
COND				OFF			
PLTSET				LABEL			
SAVE				EQUIV			
PRTMSG				CHKPNT			
PAPAM				COND			
PARAM				MCE1			
COND				CHKPNT			
PLOT				MCE2			
SAVE				CHKPNT			
PRTMSG				LABEL			
LABEL				EQUIV			
CHKPNT				CHKPNT			
GP3				COND			
SAVE				SCE1			
PARAM	345	78901234567	345	CHKPNT			
CHKPNT				LABEL			
TA1				EQUIV			
SAVE				CHKPNT			
COND	345	78901234567	345	COND			
PURGE				SMP1			
CHKPNT				CHKPNT			
PARAM				LABEL			
EMG				RBMG2			
SAVE				CHKPNT			
CHKPNT				SSG1			
COND				CHKPNT			
EMA				EQUIV			
CHKPNT				CHKPNT			
LABEL				COND			
COND				SSG2			
EMA				CHKPNT			
CHKPNT				LABEL			
LABEL				SSG3	4		
COND				SAVE	4		
COND				CHKPNT	4		
GPWG				COND	45	8901234567	345
OFF				MATGPR	45	8901234567	345
LABEL				MATGPR	45	8901234567	345
EQUIV				LABEL	45	8901234567	345
CHKPNT				SDR1			
COND				CHKPNT			
SMA3				SDR2			
CHKPNT				PARAM			
LABEL				OFF			
PARAM				SAVE			
CASE				COND			
GP4				PLOT			
SAVE				SAVE			
COND	345	901234567	345	PRTMSG			
				LABEL			

RIGID FORMAT RESTART TABLES

DMAP Inst.	63	<u>Bit Position</u> 70	80
TA1			
DSMG1			
CHKPNT			
PARAM			
PARAM			
PARAMR			
PARAML			
JUMP			
LABEL			
EQUIV			
CHKPNT			
PARAM			
EQUIV			
CHKPNT			
COND			
MCE2			
CHKPNT			
LABEL			
EQUIV			
CHKPNT			
COND			
SCE1			
CHKPNT			
LABEL			
EQUIV			
CHKPNT			
COND			
SMP2			
CHKPNT			
LABEL			
ADD			
ADD			
ADD			
COND			
MPYAD			
MPYAD			
UMERGE			
EQUIV			
COND			
UMERGE			
LABEL			
ADD			
EQUIV			
LABEL			
ADD			
REMG2			
SAVE			
CHKPNT			
PRTPARM			
PRTPARM			
JUMP			
LABEL			
PARAM			
SSG2			
SSG3			
SAVE			
CHKPNT			
COND			
MATGPP			

Inst.	63	<u>Bit Position</u> 70	80
LABEL			
SDR1	345	78901234567	345
CHKPNT	345	78901234567	345
ADD			
DSMG1			
CHKPNT			
MPYAD			
DSCHK			
SAVE			
COND			
COND			
EQUIV			
EQUIV			
EQUIV			
REPT			
TABPT			
LABEL			
ADD			
CHKPNT			
EQUIV			
CHKPNT			
EQUIV			
REPT			
TABPT			
LABEL			
CHKPNT			
SDR2			
OFF			
SAVE			
COND			
PLOT			
SAVE			
PRTMSG			
LABEL			
JUMP	345	78901234567	345
LABEL	345	78901234567	345
PRTPARM	345	78901234567	345
LABEL	345	78901234567	345
PRTPARM	345	78901234567	345
LABEL	345	78901234567	345
PRTPARM	345	78901234567	345
LABEL	345	78901234567	345
PRTPARM	345	78901234567	345
LABEL	345	78901234567	345
PRTPARM	345	78901234567	345
LABEL	345	78901234567	345
END	345	78901234567	345

STATIC ANALYSIS WITH DIFFERENTIAL STIFFNESS

7.5.5 File Name Restart Table

DMAP				DMAP			
Inst.	94	Bit Position		Inst.	94	Bit Position	
		100	110			100	110
BEGIN				PURGE		1 3 56	901 5 7
GP1	4			CHKPNT		1 3 56	901 5 7
SAVE	4			COND			
COND	4			JUMP			
CHKPNT	4			LABEL			
GP2	5			COND		2	
CHKPNT	5			GPSP		2	
PARAML				SAVE		2	
PURGE				COND		2	
COND				OFF		2	
PLTSET				LABEL		2	
SAVE				EQUIV			4
PRTMSG				CHKPNT			4
PARAM				COND		34	
PARAM				MCE1		3	
COND				CHKPNT		3	
PLOT				MCE2			4
SAVE				CHKPNT			4
PRTMSG				LABEL		34	
LABEL				EQUIV			5
CHKPNT				CHKPNT			5
GP3				COND			5
SAVE	6			SCE1			5
PARAM	6 9			CHKPNT			5
CHKPNT	6			LABEL			5
TA1	7			EQUIV			6
SAVE	7			CHKPNT			6
COND	7 9			COND			6
PURGE	7 2			SMP1			6
CHKPNT	7			CHKPNT			6
PARAM	8			LABEL			6
EMG				RBMG2			7
SAVE				CHKPNT			7
CHKPNT				SSG1			8
COND	8			CHKPNT			8
EMA	8			EQUIV			9
CHKPNT	8			CHKPNT			9
LABEL	8			COND			9
COND	9			SSG2			9
EMA	9			CHKPNT			9
CHKPNT	9			LABEL			9
LABEL	9			SSG3			0
COND				SAVE			0
COND				CHKPNT			0
GPWG				COND			
OFF				MATGPR			
LABEL	7 9			MATGPR			
EQUIV	0			LABEL			
CHKPNT	0			SDR1			1
COND	0			CHKPNT			1
SMA3	0			SDR2			2
CHKPNT	0			PARAM			
LABEL	0			OFF			
PARAM	1			SAVE			
CASE	1			COND			
GP4	1			PLOT			
SAVE	1			SAVE			
COND	1			PRTMSG			

RIGID FORMAT RESTART TABLES

DMAP Inst.	94	Bit Position 100 110	120	DMAP Inst.	94	Bit Position 100 110	120
LABEL				MATGPR			
TA1		3		LABEL			
DSMG1		3		SDR1			0
CHKPNT		3		CHKPNT			0
PARAM				ADD			
PARAM				DSMG1			
PARAMR				CHKPNT			
PARAML				MPYAD			
JUMP				DSCHK			
LABEL				SAVE			
EQUIV		4		COND			
CHKPNT		4		COND			
PARAM		4		EQUIV			
EQUIV		4		EQUIV			
CHKPNT		4		EQUIV			
COND		4		REPT			
MCE2		4		TABPT			
CHKPNT		4		LABEL			
LABEL		4		ADD			
EQUIV		5		CHKPNT			
CHKPNT		5		EQUIV			
COND		5		CHKPNT			
SCE1		5		EQUIV			
CHKPNT		5		REPT			
LABEL		5		TABPT			
EQUIV		6		LABEL			
CHKPNT		6		CHKPNT			
COND		6		SDR2			1
SMP2		6		OFF			
CHKPNT		6		SAVE			
LABEL		6		COND			
ADD		7		PLOT			
ADD		7		SAVE			
ADD		7		PRTMSG			
COND				LABEL			
MPYAD				JUMP			
MPYAD				LABEL			
UMERGE				PRTPARM			
EQUIV				LABEL			
COND				PRTPARM			
UMERGE				LABEL			
LABEL				PRTPARM			
ADD				LABEL			
EQUIV				PRTPARM			
LABEL				LABEL			
ADD				END			
RBMG2			8				
SAVE			8				
CHKPNT			8				
PRTPARM			8				
PRTPARM			8				
JUMP							
LABEL							
PAPAM							
SSG2			9				
SSG3			9				
SAVE			9				
CHKPNT			9				
COND							

BUCKLING ANALYSIS

7.6 RESTART TABLES FOR BUCKLING ANALYSIS

7.6.1 Bit Positions for Card Name Restart Table

Card Name	Bit Pos.	Card Name	Bit Pos.	Card Name	Bit Pos.
AXIC	1	CQDPLT	2	PTRPLT1	3
AXIF	1	CQUAD1	2	PTRSHL	3
CELAS1	1	CQUAD2	2	PTUBE	3
CELAS2	1	CQUADTS	2	PTWIST	3
CELAS3	1	CROD	2	GENEL	4
CELAS4	1	CSHEAR	2	CONM1	5
CMASS1	1	CTETRA	2	CONM2	5
CMASS2	1	CTORDRG	2	PELAS	6
CMASS3	1	CTRAPAX	2	PMASS	7
CMASS4	1	CTRAPRG	2	MAT1	8
CORD1C	1	CTRBSC	2	MAT2	8
CORD1R	1	CTRIA1	2	MAT3	8
CORD1S	1	CTRIA2	2	MATT1	8
CORD2C	1	CTRIAAX	2	MATT2	8
CORD2R	1	CTRIARG	2	MATT3	8
CORD2S	1	CTRIATS	2	TABLEM1	8
GRDSET	1	CTRIM6	2	TABLEM2	8
GRID	1	CTRMEM	2	TABLEM3	8
GRIDB	1	CTRPLT	2	TABLEM4	8
POINTAX	1	CTRPLT1	2	TEMPMT\$	8
RINGAX	1	CTRSHL	2	TEMPMX\$	8
RINGFL	1	CTUBE	2	AXISYM\$	9
SECTAX	1	CTWIST	2	CRIGD1	9
SEQGP	1	CWEDGE	2	CRIGD2	9
SPOINT	1	PBAR	3	CRIGD3	9
ADUM1	2	PCONEAX	3	CRIGDR	9
ADUM2	2	PDUM1	3	MPC	9
ADUM3	2	PDUM2	3	MPCADD	9
ADUM4	2	PDUM3	3	MPCAX	9
ADUM5	2	PDUM4	3	MPC\$	9
ADUM6	2	PDUM5	3	SPC	10
ADUM7	2	PDUM6	3	SPC1	10
ADUM8	2	PDUM7	3	SPCADD	10
ADUM9	2	PDUM8	3	SPCAX	10
BAROR	2	PDUM9	3	SPC\$	10
CBAR	2	PIHEX	3	ASET	11
CCONEAX	2	PQDMEM	3	ASET1	11
CDUM1	2	PQDMEM1	3	OMIT	11
CDUM2	2	PQDMEM2	3	OMIT1	11
CDUM3	2	PQDMEM3	3	OMITAX	11
CDUM4	2	PQDPLT	3	SUPAX	12
CDUM5	2	PQUAD1	3	SUPPORT	12
CDUM6	2	PQUAD2	3	TEMP	13
CDUM7	2	PQUADTS	3	TEMPAX	13
CDUM8	2	PROD	3	TEMPO	13
CDUM9	2	PSHEAR	3	TEMPP1	13
CHEXA1	2	PTORDRG	3	TEMPP2	13
CHEXA2	2	PTRAPAX	3	TEMPP3	13
CIHEX1	2	PTRBSC	3	TEMPRE	13
CIHEX2	2	PTRIA1	3	WTMASS	14
CIHEX3	2	PTRIA2	3	GRDPNT	15
CONROD	2	PTRIAAX	3	PLOTET	16
CQDMEM	2	PTRIATS	3	IRES	17
CQDMEM1	2	PTRIM6	3	PLOT\$	18
CQDMEM2	2	PTRMEM	3	POUT\$	19
CQDMEM3	2	PTRPLT	3	COUPMASS	24

RIGID FORMAT RESTART TABLES

<u>Card Name</u>	<u>Bit Pos.</u>
CPBAR	24
CPQDPLT	24
CPQUAD1	24
CPQUAD2	24
CPR0D	24
CPTRBSC	24
CPTRIA1	24
CPTRIA2	24
CPTRPLT	24
CPTUBE	24
GRAV	57
RFORCE	57
TEMPLD\$	58
DEFORM	59
DEFORM\$	59
LOAD\$	59
RFORCE\$	59
SPCD	59
FORCE	60
FORCE1	60
FORCE2	60
FORCEAX	60
LOAD	60
MOMAX	60
MOMENT	60
MOMENT1	60
MOMENT2	60
PLOAD	60
PLOAD1	60
PLOAD2	60
PLOAD3	60
PRESAX	60
SLOAD	60
EIGB	61
METHOD\$	62

BUCKLING ANALYSIS

7.6.2 Bit Positions for File Name Restart Table

<u>File Name</u>	<u>Bit Pos.</u>	<u>File Name</u>	<u>Bit Pos.</u>
BGPD T	94	GPLD	117
CSTM	94	SILD	117
EQEXIN	94	USETD	117
GPDT	94	LAMA	118
GPL	94	OEIGS	118
SIL	94	PHIA	118
ECT	95	QG	119
GPTT	96	PHIG	119
SLT	96	OB EF1	120
EST	97	OBES1	120
GEI	97	OBQG1	120
GPECT	97	OPHIG	120
GPST	98	PPHIG	120
KGGX	98	KDAAM	121
MGG	99	ELSETS	122
KGG	100	GPSETS	122
ASET	101	PLTPAR	122
RG	101	PLTSETX	122
USET	101	KDICT	123
YS	101	KELM	123
OGPST	102	MDICT	123
GM	103	MELM	123
KNN	104		
KFF	105		
KFS	105		
KSS	105		
GO	106		
KAA	106		
XOO	106		
LOO	106		
LLL	107		
PG	108		
PL	109		
PO	109		
PS	109		
RULV	110		
RUOV	110		
ULV	110		
UOOV	110		
PGG	111		
QG	111		
UGV	111		
OE F1	112		
OES1	112		
OPG1	112		
OOG1	112		
OUGV1	112		
PUGV1	112		
KDGG	113		
KDNN	114		
KOFF	115		
KDFS	115		
KDSS	115		
KDAA	116		
EED	117		
EQDYN	117		

RIGID FORMAT RESTART TABLES

7.6.3 Card Name Restart Table

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
BEGIN	1234567890123456789			4			789012
FILE	1234567890123456789			4			789012
GP1	1						
SAVE	1						
CHKPNT	1						
\$\$\$		6					
GP2	12 45		6				
CHKPNT	12 45		6				
\$\$\$		6					
PARAML			8				
\$\$\$		7					
PURGE			8				
\$\$\$		7					
COND			8				
\$\$\$		7					
PLTSET			8				
\$\$\$		7					
SAVE			8				
\$\$\$		7					
PRTMSG			8				
\$\$\$		7					
PARAM			8				
\$\$\$		7					
PARAM			8				
\$\$\$		7					
COND			8				
\$\$\$		7					
PLOT			8				
\$\$\$		7					
SAVE			8				
\$\$\$		7					
PRTMSG			8				
\$\$\$		7					
LABEL			8				
\$\$\$		7					
CHKPNT			8				
\$\$\$		67					
GP3	12		3				7 0
SAVE	12		3				7 0
PARAM	12		3				7 0
CHKPNT	12		3				7 0
\$\$\$		6					
TA1	1234567		3				
SAVE	1234567		3				
COND	12345678		3				
PURGE	1234567		3				
CHKPNT	1234567		3				
\$\$\$		6					
PARAM	123 6 8						
EMG	123 5678		45				7
SAVE	123 5678		45				7
CHKPNT	123 5678		45				7
\$\$\$		6					
COND	123 6 8						
EMA	123 6 8						
CHKPNT	123 6 8						
\$\$\$		6					
LABEL	123 6 8						
COND	123 5 78		45				7

BUCKLING ANALYSIS

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
EMA	123 5 7 8	45					7
CHKPNT	123 5 7 8	45					7
\$\$\$	6						
LABEL	123 5 7 8	45					7
COND	123 5 7 8	45	4				
\$\$\$	8						
COND	123 5 7 8	45	4				
\$\$\$	8						
GPWG	123 5 7 8	45	4				
\$\$\$	8						
OFF	123 5 7 8	45	4				
\$\$\$	8						
LABEL	123 5 7 8	45					7
EQUIV	1234 6 8						
CHKPNT	1234 6 8						
\$\$\$	6						
COND	1234 6 8						
SMA3	1234 6 8						
CHKPNT	1234 6 8						
\$\$\$	6						
LABEL	1234 6 8						
PARAM	1	901					9
GP4	1	901					9
SAVE	1	901					9
COND	1	901					9
PARAM	1	901					9
PURGE	1	901					9
CHKPNT	1234 6 8 901						9
\$\$\$	6						
COND	1	2					
JUMP	1	2					
LABEL	1	2					
COND	123 6 8 90						
GPSP	123 6 8 90						
SAVE	123 6 8 90						
COND	123 6 8 90						
OFF	123 6 8 90						
LABEL	123 6 8 90						
EQUIV	1234 6 8 9						
CHKPNT	1234 6 8 9						
\$\$\$	6						
COND	1234 6 8 9						
MCE1	1	9					
CHKPNT	1	9					
\$\$\$	6						
MCE2	1234 6 8 9						
CHKPNT	1234 6 8 9						
\$\$\$	6						
LABEL	1234 6 8 9						
EQUIV	1234 6 8 90						
CHKPNT	1234 6 8 90						
\$\$\$	6						
COND	1234 6 8 90						
SCE1	1234 6 8 90						
CHKPNT	1234 6 8 90						
\$\$\$	6						
LABEL	1234 6 8 90						
EQUIV	1234 6 8 901						
CHKPNT	1234 6 8 901						

RIGID FORMAT RESTART TABLES

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
\$\$\$		6					
COND	1234	6 8901					
SMP1	1234	6 8901					
CHKPNT	1234	6 8901					
\$\$\$		6					
LABEL	1234	6 8901					
RBMG2	1234	6 8901					
CHKPNT	1234	6 8901					
\$\$\$		6					
SSG1	123	5678	3				7890
CHKPNT	123	5678	3				7890
\$\$\$		6					
EQUIV	123	5678901	3				7890
CHKPNT	123	5678901	3				7890
\$\$\$		6					
COND	123	5678901	3				7890
SSG2	123	5678901	3				7890
CHKPNT	123	5678901	3				7890
\$\$\$		6					
LABEL	123	5678901	3				7890
SSG3	1234	5678901	3				7890
SAVE	1234	5678901	3				7890
CHKPNT	1234	5678901	3				7890
\$\$\$		6					
COND	1234	5678901	3 7				7890
MATGPR	1234	5678901	3 7				7890
MATGPR	1234	5678901	3 7				7890
LABEL	1234	5678901	3 7				7890
SDR1	1234	5678901	3				7890
CHKPNT	1234	5678901	3				7890
\$\$\$		6					
SDR2			9				
PARAM							
OFF			9				
SAVE			9				
COND			8				
\$\$\$		7					
PLOT			8				
\$\$\$		7					
SAVE			8				
\$\$\$		7					
PRTMSG			8				
\$\$\$		7					
LABEL			8				
\$\$\$		7					
TA1	1234	5678901					7890
DSMG1	1234	5678901					7890
CHKPNT	1234	5678901					7890
\$\$\$		6					
EQUIV	1234	5678901					7890
CHKPNT	1234	5678901					7890
\$\$\$		6					
COND	1234	5678901					7890
MCE2	1234	5678901					7890
CHKPNT	1234	5678901					7890
\$\$\$		6					
LABEL	1234	5678901					7890
EQUIV	1234	5678901					7890
CHKPNT	1234	5678901					7890

BUCKLING ANALYSIS

DMAP	Bit Position						
Inst.	1	10	20	30	40	50	60
\$SS	6						
COND	12345678901						7890
SCE1	12345678901						7890
CHKPNT	12345678901						7890
\$SS	6						
LABEL	12345678901						7890
EQUIV	12345678901						7890
CHKPNT	12345678901						7890
\$SS	6						
COND	12345678901						7890
SMP2	12345678901						7890
CHKPNT	12345678901						7890
\$SS	6						
LABEL	12345678901						7890
ADD	12345678901						7890
CHKPNT	12345678901						7890
\$SS	6						
OPD	12345678901						78901
SAVE	12345678901						78901
COND	12345678901						78901
CHKPNT	12345678901						78901
\$SS	6						
PARAM	12345678901						7890
READ	12345678901						789012
SAVE	12345678901						789012
CHKPNT	12345678901						789012
\$SS	6						
OFF	12345678901						789012
SAVE	12345678901						789012
COND	12345678901						789012
SDR1	12345678901						789012
CHKPNT	12345678901						789012
\$SS	6						
SDR2			89				
OFF			9				
SAVE			9				
COND			8				
\$SS	7						
PLOT			8				
\$SS	7						
SAVE			8				
\$SS	7						
PRTMSG			8				
\$SS	7						
LABEL			8				
\$SS	7						
JUMP	1234567890123456789			4			789012
LABEL	1234567890123456789			4			789012
PRTPARM	1234567890123456789			4			789012
LABEL	1234567890123456789			4			789012
PRTPARM	1234567890123456789			4			789012
LABEL	1234567890123456789			4			789012
PRTPARM	1234567890123456789			4			789012
LABEL	1234567890123456789			4			789012
PRTPARM	1234567890123456789			4			789012
LABEL	1234567890123456789			4			789012
\$SS	8						
PRTPARM	1234567890123456789			4			789012

RIGID FORMAT RESTART TABLES

DMAP Inst.	1	10	20	<u>Bit Position</u> 30	40	50	60
SSS		8					
LABEL	1234567890	123456789	4				769012
PRTPARM	1234567890	123456789	4				789012
LABEL	1234567890	123456789	4				799012
END	1234567890	123456789	4				789012

BUCKLING ANALYSIS

7.6.4 Rigid Format Change Restart Table

DMAP Inst.	Bit Position			DMAP Inst.	Bit Position		
	63	70	80		63	70	80
BEGIN	3456	8901234567	345	CHKPNT			
FILE	3456	8901234567	345	COND	345	901234567	345
GP1				JUMP	345	901234567	345
SAVE				LABEL	345	901234567	345
CHKPNT				COND			
GP2				GPSP			
CHKPNT				SAVE			
PARAML				COND			
PURGE				OFF			
COND				LABEL			
PLTSET				EQUIV			
SAVE				CHKPNT			
PRTMSG				COND			
PARAM				MCE1			
PARAM				CHKPNT			
COND				MCE2			
PLOT				CHKPNT			
SAVE				LABEL			
PRTMSG				EQUIV			
LABEL				CHKPNT			
CHKPNT				COND			
GP3				SCE1			
SAVE				CHKPNT			
PARAM	3456	8901234567	345	LABEL			
CHKPNT				EQUIV			
TA1				CHKPNT			
SAVE				COND			
COND	3456	8901234567	345	SMP1			
PURGE				CHKPNT			
CHKPNT				LABEL			
PARAM				RBMG2			
EMG				CHKPNT			
SAVE				SSG1			
CHKPNT				CHKPNT			
COND				EQUIV			
EMA				CHKPNT			
CHKPNT				COND			
LABEL				SSG2			
COND				CHKPNT			
EMA				LABEL			
CHKPNT				SSG3	4		
LABEL				SAVE	4		
COND				CHKPNT	4		
COND				COND	45	8901234	
GPWG				MATGPR	45	8901234	
OFF				MATGPR	45	8901234	
LABEL				LABEL	45	8901234	
EQUIV				SDR1			
CHKPNT				CHKPNT			
COND				SDR2			
SMA3				PARAM			
CHKPNT				OFF			
LABEL				SAVE			
PARAM				COND			
GP4				PLOT			
SAVE				SAVE			
COND	3456	8901234567	345	PRTMSG			
PARAM				LABEL			
PURGE				TA1			

RIGID FORMAT RESTART TABLES

DMAP Inst.	63	<u>Bit Position</u> 70	80
DSMG1			
CHKPNT			
EQUIV			
CHKPNT			
COND			
MCE2			
CHKPNT			
LABEL			
EQUIV			
CHKPNT			
COND			
SCE1			
CHKPNT			
LABEL			
EQUIV			
CHKPNT			
COND			
SMP2			
CHKPNT			
LABEL			
ADD			
CHKPNT			
OPD			
SAVE			
COND	345	78901234567	345
CHKPNT			
PAPAM			
READ			
SAVE			
CHKPNT			
OFF			
SAVE			
COND			
SDR1	345	78901234567	345
CHKPNT	345	78901234567	345
SDR2			
OFF			
SAVE			
COND			
PLOT			
SAVE			
PRTMSG			
LABEL			
JUMP	345	78901234567	345
LABEL	345	78901234567	345
PRTPARM	345	78901234567	345
LABEL	345	78901234567	345
PRTPARM	345	78901234567	345
LABEL	345	78901234567	345
PRTPARM	345	78901234567	345
LABEL	345	78901234567	345
PRTPARM	345	78901234567	345
LABEL	345	78901234567	345
PRTPARM	345	78901234567	345
LABEL	345	78901234567	345
PRTPARM	345	78901234567	345
LABEL	345	78901234567	345
END	345	78901234567	345

BUCKLING ANALYSIS

7.6.5 File Name Restart Table

DMAP		Bit Position	
Inst.	94	100	110
BEGIN			
FILE			
GP1	4		
SAVE	4		
CHKPNT	4		
GP2	5		
CHKPNT	5		
PARAML			
PURGE			
COND			
PLTSET			
SAVE			
PRTMSG			
PARAM			
PARAM			
COND			
PLOT			
SAVE			
PRTMSG			
LABEL			
CHKPNT			
GP3	6		
SAVE	6		
PARAM	6 9		
CHKPNT	6		
TA1	7		
SAVE	7		
COND	7 9		
PURGE	7	2	
CHKPNT	7		
PARAM	8		
EMG			
SAVE			
CHKPNT			
COND	8		
EMA	8		
CHKPNT	8		
LABEL	8		
COND	9		
EMA	9		
CHKPNT	9		
LABEL	9		
COND			
COND			
GPWG			
OFF			
LABEL	7 9		
EQUIV	0		
CHKPNT	0		
COND	0		
SMA3	0		
CHKPNT	0		
LABEL	0		
PARAM	1		
GP4	1		
SAVE	1		
COND	1		
PARAM	1		

DMAP		Bit Position		
Inst.	94	100	110	120
PURGE		1 3 56	901	
CHKPNT		1 3 56	901	
COND				
JUMP				
LABEL				
COND		2		
GPSP		2		
SAVE		2		
COND		2		
OFF		2		
LABEL		2		
EQUIV		4		
CHKPNT		4		
COND		34		
MCE1		3		
CHKPNT		3		
MCE2		4		
CHKPNT		4		
LABEL		34		
EQUIV		5		
CHKPNT	2	5		
COND		5		
SCE1		5		
CHKPNT		5		
LABEL		5		
EQUIV		6		
CHKPNT		6		
COND		6		
SMP1		6		
CHKPNT		6		
LABEL		6		
REMG2	3	7		
CHKPNT	3	7		
SSG1	3	8		
CHKPNT		8		
EQUIV		9		
CHKPNT		9		
COND		9		
SSG2		9		
CHKPNT		9		
LABEL		9		
SSG3		0		
SAVE		0		
CHKPNT		0		
COND				
MATGPR				
MATGPR				
LABEL				
SDR1			1	
CHKPNT			1	
SDR2			2	
PARAM				
OFF				
SAVE				
COND				
PLOT				
SAVE				
PRTMSG				

RIGID FORMAT RESTART TABLES

DMAP		Bit Position		
Inst.	94	100	110	120
LABEL				
TA1			3	
DSMG1			3	
CHKPNT			3	
EQUIV			4	
CHKPNT			4	
COND			4	
MCE2			4	
CHKPNT			4	
LABEL			4	
EQUIV			5	
CHKPNT			5	
COND			5	
SCE1			5	
CHKPNT			5	
LABEL			5	
EQUIV			6	
CHKPNT			6	
COND			6	
SMP2			6	
CHKPNT			6	
LABEL			6	
ADD				1
CHKPNT				1
DPD			7	
SAVE			7	
COND			7	
CHKPNT			7	
PARAM			8	
READ			8	
SAVE			8	
CHKPNT			8	
OFF			8	
SAVE			8	
COND			9	
SDP1			9	
CHKPNT			9	
SDR2				0
OFF				
SAVE				
COND				
PLOT				
SAVE				
PRTMSG				
LABEL				
JUMP				
LABEL				
PRTPARM				
LABEL				
PRTPARM				
LABEL				
PRTPARM				
LABEL				
PRTPARM				
LABEL				
PRTPARM				
LABEL				
PRTPARM				
END				

PIECEWISE LINEAR ANALYSIS

7.7 RESTART TABLES FOR PIECEWISE LINEAR ANALYSIS

7.7.1 Bit Positions for Card Name Restart Table

Card Name	Bit Pos.	Card Name	Bit Pos.	Card Name	Bit Pos.
AXIC	1	CQUAD1	2	PTUBE	3
AXIF	1	CQUAD2	2	PTWIST	3
CELAS1	1	CQUADTS	2	GENEL	4
CELAS2	1	CROD	2	CONM1	5
CELAS3	1	CSHEAR	2	CONM2	5
CELAS4	1	CTETRA	2	PELAS	6
CMASS1	1	CTORDRG	2	PMASS	7
CMASS2	1	CTRAPAX	2	MAT1	8
CMASS3	1	CTRAPRG	2	MAT2	8
CMASS4	1	CTRBSC	2	MAT3	8
CORD1C	1	CTRIA1	2	MATS1	8
CORD1R	1	CTRIA2	2	MATS2	8
CORD1S	1	CTRIAAX	2	MATT1	8
CORD2C	1	CTRIARG	2	MATT2	8
CORD2R	1	CTRIATS	2	MATT3	8
CORD2S	1	CTRIM6	2	TABLEM1	8
GRDSET	1	CTRMEM	2	TABLEM2	8
GRID	1	CTRPLT	2	TABLEM3	8
GRIDB	1	CTRPLT1	2	TABLEM4	8
POINTAX	1	CTRSHL	2	TABLES1	8
RINGAX	1	CTUBE	2	TABLES2	8
RINGFL	1	CTWIST	2	TABLES3	8
SECTAX	1	CWEDGE	2	TABLES4	8
SEQGP	1	PBAR	3	TEMPMT\$	8
SPOINT	1	PCONEAX	3	TEMPMX\$	8
ADUM1	2	PDUM1	3	AXISYM\$	9
ADUM2	2	PDUM2	3	CRIGD1	9
ADUM3	2	PDUM3	3	CRIGD2	9
ADUM4	2	PDUM4	3	CRIGD3	9
ADUM5	2	PDUM5	3	CRIGDR	9
ADUM6	2	PDUM6	3	MPC	9
ADUM7	2	PDUM7	3	MPCADD	9
ADUM8	2	PDUM8	3	MPCAX	9
ADUM9	2	PDUM9	3	MPC\$	9
BAROR	2	PIHEX	3	SPC	10
CBAR	2	PQDMEM	3	SPC1	10
CCONEAX	2	PQDMEM1	3	SPCADD	10
CDUM1	2	PQDMEM2	3	SPCAX	10
CDUM2	2	PQDMEM3	3	SPC\$	10
CDUM3	2	PQDPLT	3	ASET	11
CDUM4	2	PQUAD1	3	ASET1	11
CDUM5	2	PQUAD2	3	OMIT	11
CDUM6	2	PQUADTS	3	OMIT1	11
CDUM7	2	PROD	3	OMITAX	11
CDUM8	2	PSHEAR	3	SUPAX	12
CDUM9	2	PTORDRG	3	SUPORT	12
CHEXA1	2	PTRAPAX	3	TEMP	13
CHEXA2	2	PTRBSC	3	TEMPAX	13
CIHEX1	2	PTRIA1	3	TEMPO	13
CIHEX2	2	PTRIA2	3	TEMPP1	13
CIHEX3	2	PTRIAAX	3	TEMPP2	13
CONROD	2	PTRIATS	3	TEMPP3	13
CQDMEM	2	PTRIM6	3	TEMPRB	13
CQDMEM1	2	PTRMEM	3	WTMASS	14
CQDMEM2	2	PTRPLT	3	GROPNT	15
CQDMEM3	2	PTRPLT1	3	PLOTET	16
CQDPLT	2	PTRSHL	3	IRES	17

RIGID FORMAT RESTART TABLES

Card Name Bit Pos.

PLOTS	18
POUTS	19
LOOPS	22
LOOP1S	23
COUPMASS	24
CPBAR	24
CPROD	24
CPQDPLT	24
CPQUAD1	24
CPQUAD2	24
CPROD	24
CPTRBSC	24
CPTRIA1	24
CPTRIA2	24
CPTRPLT	24
CPTUBE	24
PLCOS	58
PLFACT	58
DEFORM	59
DEFORMS	59
LOADS	59
RFORCES	59
SPCD	59
FORCE	60
FORCE1	60
FORCE2	60
FORCEAX	60
LOAD	60
MOMAX	60
MOMENT	60
MOMENT1	60
MOMENT2	60
PLOAD	60
PLOAD1	60
PLOAD2	60
PLOAD3	60
PRESAX	60
SLOAD	60
GRAV	61
RFORCE	61
TEMPLOS	62

PIECEWISE LINEAR ANALYSIS

7.7.2 Bit Positions for File Name Restart Table

<u>File Name</u>	<u>Bit Pos.</u>	<u>File Name</u>	<u>Bit Pos.</u>
BGPD	94	LOO	108
CSTM	94	KLL	109
EQEXIN	94	KLR	109
GPD	94	KRR	109
GPL	94	LLL	110
SIL	94	DM	111
ECT	95	PG	112
GPTT	96	PL	113
SLT	96	PO	113
ECPT	97	IPS	113
EST	97	QR	113
GEI	97	RULV	114
GPCT	97	RUOV	114
GPST	98	IULV	114
KGGX	98	UOOV	114
MGG	99	PGG	115
KGGXL	100	DELTAQG	115
ECPTNL	100	DELTAUGV	115
ESTL	100	UGV1	116
ESTNL	100	QG1	116
KGG	101	ONLES	117
KGGL	101	ESTNL1	117
ASET	102	KGGNL	118
RG	102	ECPTNL1	118
USET	102	KGGSUM	119
YS	102	DEF1	120
PG1	103	OES1	120
OGPST	104	OPG1	120
GM	105	OQG1	120
KNN	106	OUGV1	120
KFF	107	PUGV1	120
KFS	107	ELSETS	121
KSS	107	GPSETS	121
GO	108	PLTPAR	121
KAA	108	PLTSETX	121
KOO	108		

RIGID FORMAT RESTART TABLES

7.7-3 Card Name Restart Table

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
BEGIN	1234567890123456789	234					89012
FILE	1234567890123456789	234					89012
GP1	1						
SAVE	1						
CHKPNT	1						
\$SS	6						
GP2	12 45	6					
CHKPNT	12 45	6					
\$SS	6						
PARAML			8				
\$SS	7						
PURGE			8				
\$SS	7						
COND			8				
\$SS	7						
PLTSET			8				
\$SS	7						
SAVE			8				
\$SS	7						
PRTMSG			8				
\$SS	7						
PARAM			8				
\$SS	7						
PARAM			8				
\$SS	7						
COND			8				
\$SS	7						
PLOT			8				
\$SS	7						
SAVE			8				
\$SS	7						
PRTMSG			8				
\$SS	7						
LABEL			8				
\$SS	7						
CHKPNT			8				
\$SS	67						
GP3	12	3					01
SAVE	12	3					01
PARAM	12	3 5					01
CHKPNT	12	3					01
\$SS	6						
TA1	1234567	3					
SAVE	1234567	3					
PARAM	1234567	3					
COND	1234567	3					
PUPGE	1234567	3					
CHKPNT	1234567	3					
\$SS	6						
COND	123 5678	45	4				1
PARAM	123 6 8						
PARAM	123 6 8						
EMG	123 5678	45	4				1
SAVE	12345678	45	4				1
CHKPNT	123 5678	45	4				1
\$SS	6						
COND	123 6 8						
EMA	123 6 8						
CHKPNT	123 6 8						

PIECEWISE LINEAR ANALYSIS

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
\$\$\$	6						
LABEL	123 6 8						
COND	123 5 7 8	45	4				1
EMA	123 5 7 8	45	4				1
CHKPNT	123 5 7 8	45	4				1
\$\$\$	6						
LABEL	123 5 7 8	45	4				1
COND	123 5 7 8	45	4				
\$\$\$	8						
COND	123 5 7 8	45	4				
\$\$\$	8						
GPWG	123 5 7 8	45	4				
\$\$\$	8						
OFF	123 5 7 8	45	4				
\$\$\$	8						
LABEL	123 5 7 8	45	4				1
PLA1	123 6 8						
SAVE	123 6 8						
COND	123 6 8						
PURGE	123 6 8						
CHKPNT	123 6 8						
\$\$\$	6						
PARAM	123 6 8						
PARAM	123 6 8						
EQUIV	1234 6 8						
CHKPNT	1234 6 8						
\$\$\$	6						
COND	1234 6 8						
SMA3	1234 6 8						
CHKPNT	1234 6 8						
\$\$\$	6						
SMA3	1234 6 8						
CHKPNT	1234 6 8						
\$\$\$	6						
LABEL	1234 6 8						
PARAM	1 9012						
GP4	1 9012						9
SAVE	1 9012						9
PARAM	1 9012						9
PURGE	1 9012						9
CHKPNT	1234 6 8 9012		23				9
\$\$\$	6						
SSG1	123 5679						9012
CHKPNT	123 5678						9012
\$\$\$	6						
EQUIV	123 5678						9012
CHKPNT	123 5678						9012
\$\$\$	6						
COND	123 6 8 90						
GPSP	123 6 8 90						
SAVE	123 6 8 90						
COND	123 6 8 90						
OFF	123 6 8 90						
LABEL	123 6 8 90						
PARAM			23				
EQUIV	1234 6 8 9						
CHKPNT	1234 6 8 9						
\$\$\$	6						
COND	1234 6 8 9						

RIGID FORMAT RESTART TABLES

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
MCE1	1	9					
CHKPNT	1	9					
SSS		6					
PARAM		9					
JUMP			23				
LABEL	1234	6 89	23				
EQUIV	1234	6 89	23				
CHKPNT	1234	6 89	23				
SSS		6					
COND	1234	6 89	23				
MCE2	1234	6 89	23				
CHKPNT	1234	6 89	23				
SSS		6					
LABEL	1234	6 89	23				
EQUIV	1234	6 890	23				
CHKPNT	1234	6 890	23				
SSS		6					
COND	1234	6 890	23				
SCE1	1234	6 890	23				
CHKPNT	1234	6 890	23				
SSS		6					
LABEL	1234	6 890	23				
EQUIV	1234	6 8901	23				
CHKPNT	1234	6 8901	23				
SSS		6					
COND	1234	6 8901	23				
SMP1	1234	6 8901	23				
CHKPNT	1234	6 8901	23				
SSS		6					
LABEL	1234	6 8901	23				
EQUIV	1234	6 89012	23				
CHKPNT	1234	6 89012	23				
SSS		6					
COND	1234	6 89012	23				
REMG1	1234	6 89012	23				
CHKPNT	1234	6 89012	23				
SSS		6					
LABEL	1234	6 89012	23				
DECOMP	1234	6 89012	23				
SAVE	1234	6 89012	23				
COND	1234	6 89012	23				
CHKPNT	1234	6 89012	23				
SSS		6					
COND	1234	6 89012	23				
REMG3	1234	6 89012	23				
CHKPNT	1234	6 89012	23				
SSS		6					
LABEL	1234	6 89012	23				
ADD	123	5678 3	23				89012
CHKPNT	123	5678 3	23				89012
SSS		6					
COND	123	567890123	23				89012
SSG2	123	567890123	23				89012
CHKPNT	123	567890123	23				89012
SSS		6					
LABEL	123	567890123	23				89012
SSG3	1234567890123		23				89012
SAVE	1234567890123		23				89012
CHKPNT	1234567890123		23				89012

PIECEWISE LINEAR ANALYSIS

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
\$\$\$	6						
COND	1234567890123	7	23				89012
MATGPR	1234567890123	7	23				89012
MATGPR	1234567890123	7	23				89012
LABEL	1234567890123	7	23				89012
SDR1	1234567890123		23				89012
CHKPNT	1234567890123		23				89012
\$\$\$	6						
PLA2	123456789012		23				89012
SAVE	123456789012		23				89012
CHKPNT	123456789012		23				89012
\$\$\$	6						
EQUIV	123456789012		23				89012
COND			23				
PLA3			23				
CHKPNT			23				
\$\$\$	6						
OFF			23				
SAVE			23				
LABEL			23				
PARAM	123456789012		23				89012
COND	123456789012		23				89012
PLA4	123456789012		23				89012
SAVE	123456789012		23				89012
CHKPNT	123456789012		23				89012
\$\$\$	6						
EQUIV	123456789012		23				89012
CHKPNT	123456789012		23				89012
\$\$\$	6						
COND	123456789012		23				89012
ADD	123456789012		23				89012
CHKPNT	123456789012		23				89012
\$\$\$	6						
LABEL	123456789012		23				89012
EQUIV	123456789012		23				89012
CHKPNT	123456789012		23				89012
\$\$\$	6						
EQUIV	123456789012		23				89012
CHKPNT	123456789012		23				89012
\$\$\$	6						
COND	123456789012		23				89012
PLA2	123456789012		23				89012
PLA2	123456789012		23				89012
LABEL	123456789012		23				89012
REPT	123456789012		23				89012
JUMP	123456789012		23				89012
LABEL	1234 6 89012		23				
PRTPARM	1234 6 89012		23				
LABEL	123456789012		23				89012
SDR2		89					
OFF		9					
SAVE		9					
COND		8					
\$\$\$	7						
PLOT		8					
\$\$\$	7						
SAVE		8					
\$\$\$	7						
PRTMSG		8					

RIGID FORMAT RESTART TABLES

DMAP Inst.	1	10	20	<u>Bit Position</u> 30	40	50	60
SSS	7						
LABEL			8				
SSS	7						
JUMP	1234567890123456789		234				89012
LABEL	1234567890123456789		234				89012
PRTPARM	1234567890123456789		234				89012
LABEL	1234567890123456789		234				89012
PRTPARM	1234567890123456789		234				89012
LABEL	1234567890123456789		234				89012
SSS	8						
PRTPARM	1234567890123456789		234				89012
SSS	8						
LABEL	1234567890123456789		234				89012
PRTPARM	1234567890123456789		234				89012
LABEL	1234567890123456789		234				89012
END	1234567890123456789		234				89012

PIECEWISE LINEAR ANALYSIS

7.7.4 Rigid Format Change Restart Table

DMAP Inst.	63	Bit Position 70	80	DMAP Inst.	63	Bit Position 70	80
BEGIN		34567 901234		CHKPNT			
FILE		34567 901234		COND			
GP1				SMA3			
SAVE				CHKPNT			
CHKPNT				SMA3			
GP2				CHKPNT			
CHKPNT				LABEL			
PARAML				PARAM			
PURGE				GP4			
COND				SAVE			
PLTSET				PARAM			
SAVE				PURGE			
PRTMSG				CHKPNT			
PARAM				SSG1			
PARAM				CHKPNT			
COND				EQUIV			
PLOT				CHKPNT			
SAVE				COND			
PRTMSG				GPSP			
LABEL				SAVE			
CHKPNT				COND			
GP3				OFF			
SAVE				LABEL			
PARAM	34567	901234		PARAM			
CHKPNT				EQUIV			
TA1				CHKPNT			
SAVE				COND			
PARAM	34567	901234		MCE1			
COND	34567	901234		CHKPNT			
PURGE				PARAM			
CHKPNT				JUMP			
COND				LABEL			
PARAM				EQUIV			
PARAM				CHKPNT			
EMG				COND			
SAVE				MCE2			
CHKPNT				CHKPNT			
COND				LABEL			
EMA				EQUIV			
CHKPNT				CHKPNT			
LABEL				COND			
COND				SCE1			
EMA				CHKPNT			
CHKPNT				LABEL			
LABEL				EQUIV			
COND				CHKPNT			
COND				COND			
GPWG				SMP1			
OFF				CHKPNT			
LABEL				LABEL			
PLA1				EQUIV			
SAVE				CHKPNT			
COND				COND			
PURGE				RBMG1			
CHKPNT				CHKPNT			
PARAM				LABEL			
PARAM				DECOMP			
EQUIV				SAVE			

RIGID FORMAT RESTART TABLES

DMAP Inst.	63	<u>Bit Position</u> 70	80
COND			
CHKPNT			
COND			
RBMG3			
CHKPNT			
LABEL			
ADD	34	67	
CHKPNT	34	67	
COND	34	67	
SSG2	34	67	
CHKPNT	34	67	
LABEL	34	67	
SSG3	34	67	
SAVE	34	67	
CHKPNT	34	67	
COND	34567	901234	
MATGPR	34567	901234	
MATGPR	34567	901234	
LABEL	34567	901234	
SCR1	34567	901234	
CHKPNT	34567	901234	
PLA2			
SAVE			
CHKPNT			
EQUIV			
COND			
PLA3			
CHKPNT			
OFF			
SAVE			
LABEL			
PARAM			
COND			
PLA4			
SAVE			
CHKPNT			
EQUIV			
CHKPNT			
COND			
ADD			
CHKPNT			
LABEL			
EQUIV			
CHKPNT			
EQUIV			
CHKPNT			
COND			
PLA2			
PLA2			
LABEL			
REPT			
JUMP			
LABEL			
PRTPARM			
LABEL			
SDR2			
OFF			
SAVE			
COND			
PLOT			

DMAP Inst.	63	<u>Bit Position</u> 70	80
SAVE			
PRTMSG			
LABEL			
JUMP	34567	901234	
LABEL	34567	901234	
PRTPARM	34567	901234	
LABEL	34567	901234	
PRTPARM	34567	901234	
LABEL	34567	901234	
PRTPARM	34567	901234	
LABEL	34567	901234	
PRTPARM	34567	901234	
LABEL	34567	901234	
END	34567	901234	

PIECEWISE LINEAR ANALYSIS

7.7-5 File Name Restart Table

DMAP		Bit Position		DMAP		Bit Position	
Inst.	94	100	110	Inst.	94	100	110
BEGIN				CHKPNT		1	
FILE				COND		1	
GP1	4			SMA3		1	
SAVE	4			CHKPNT		1	
CHKPNT	4			SMA3		1	
GP2	5			CHKPNT		1	
CHKPNT	5			LABEL		1	
PARAML			1	PARAM		2	
PURGE			1	GP4		2	
COND			1	SAVE		2	
PLTSET			1	PARAM		2	
SAVE			1	PURGE		2	5 789 1 345
PRTMSG			1	CHKPNT		2	5 789 1 345
PARAM			1	SSG1		3	
PARAM			1	CHKPNT		3	
COND				EQUIV		3	
PLOT				CHKPNT		3	
SAVE				COND		4	
PRTMSG				GPSP		4	
LABEL				SAVE		4	
CHKPNT			1	COND		4	
GP3	6			OFF		4	
SAVE	6			LABEL		4	
PARAM	6 9			PARAM			
CHKPNT	6			EQUIV		6	
TA1	7			CHKPNT		6	
SAVE	7			COND		56	
PARAM	7			MCE1		5	
COND	7			CHKPNT		5	
PURGE	78			PARAM			
CHKPNT	7			JUMP			
COND	89			LABEL			
PARAM	8			EQUIV		6	
PARAM	8			CHKPNT		6	
EMG			6	COND		6	
SAVE			6	MCE2		6	
CHKPNT			6	CHKPNT		6	
COND	8			LABEL		56	
EMA	8			EQUIV		7	
CHKPNT	8			CHKPNT		7	
LABEL	8			COND		7	
COND	9			SCE1		7	
EMA	9			CHKPNT		7	
CHKPNT	9			LABEL		7	
LABEL	9			EQUIV		8	
COND				CHKPNT		8	
COND				COND		8	
GPWG				SMP1		8	
OFF				CHKPNT		8	
LABEL	89			LABEL		8	
PLA1	0			EQUIV		9	
SAVE	0			CHKPNT		9	
COND	0			COND		9	
PURGE	0			RRMG1		9	
CHKPNT	0			CHKPNT		9	
PARAM	0			LABEL		9	
PARAM	0			DECOMP		0	
EQUIV	1			SAVE		0	

RIGID FORMAT RESTART TABLES

DMAP Inst.	94	Bit Position 100	110	120	DMAP Inst.	94	Bit Position 100	110	120
COND			0		PLOT				
CHKPNT			0		SAVE				
COND			1		PRTMSG				
RBMG3			1		LABEL				
CHKPNT			1		JUMP				
LABEL			1		LABEL				
ADD			2		PRTPARM				
CHKPNT			2		LABEL				
COND			3		PRTPARM				
SSG2			3		LABEL				
CHKPNT			3		PRTPARM				
LABEL			3		LABEL				
SSG3			4		PRTPARM				
SAVE			4		LABEL				
CHKPNT			4		END				
COND									
MATGPR									
MATGPR									
LABEL									
SDR1			5						
CHKPNT			5						
PLA2			6						
SAVE									
CHKPNT									
EQUIV				7					
COND				7					
PLA3				7					
CHKPNT				7					
OFF									
SAVE									
LABEL				7					
PARAM				8					
COND				8					
PLA4				8					
SAVE				8					
CHKPNT				8					
EQUIV				9					
CHKPNT				9					
COND				9					
ADD				9					
CHKPNT				9					
LABEL				9					
EQUIV									
CHKPNT									
EQUIV									
CHKPNT									
COND									
PLA2									
PLA2									
LABEL									
REPT									
JUMP									
LABEL									
PRTPARM									
LABEL									
SDR2				0					
OFF									
SAVE									
COND									

DIRECT COMPLEX EIGENVALUE ANALYSIS

7.8 RESTART TABLES FOR DIRECT COMPLEX EIGENVALUE ANALYSIS

7.8.1 Bit Positions for Card Name Restart Table

Card Name	Bit Pos.	Card Name	Bit Pos.	Card Name	Bit Pos.
AXIC	1	CIHEX2	2	PTRIAAX	3
AXIF	1	CIHEX3	2	PTRIATS	3
CDAMP1	1	CONROD	2	PTRIM6	3
CDAMP2	1	CQDMEM	2	PTRMEM	3
CDAMP3	1	CQDMEM1	2	PTRPLT	3
CDAMP4	1	CQDMEM2	2	PTRPLT1	3
CELAS1	1	CQDMEM3	2	PTRSHL	3
CELAS2	1	CQDPLT	2	PTUBE	3
CELAS3	1	CQUAD1	2	PTWIST	3
CELAS4	1	CQUAD2	2	GENEL	4
CMASS1	1	CQUADTS	2	CONM1	5
CMASS2	1	CROD	2	CONM2	5
CMASS3	1	CSHEAR	2	FSLIST	5
CMASS4	1	CTETRA	2	PELAS	6
CORD1C	1	CTORDRG	2	PMASS	7
CORD1R	1	CTRAPAX	2	MAT1	8
CORD1S	1	CTRAPRG	2	MAT2	8
CORD2C	1	CTRBSC	2	MAT3	8
CORD2R	1	CTRIA1	2	MATT1	8
CORD2S	1	CTRIA2	2	MATT2	8
FREET	1	CTRIAAX	2	MATT3	8
GRDSET	1	CTRIARG	2	TABLEM1	8
GRID	1	CTRIATS	2	TABLEM2	8
GRIOB	1	CTRIM6	2	TABLEM3	8
POINTAX	1	CTRMEM	2	TABLEM4	8
PREPT	1	CTRPLT	2	TEMPMT3	8
RINGAX	1	CTRPLT1	2	TEMPMX3	8
RINGFL	1	CTRSHL	2	AXISYM3	9
SECTAX	1	CTUBE	2	CRIGD1	9
SEQGP	1	CTWIST	2	CRIGD2	9
SPOINT	1	CWEDGE	2	CRIGD3	9
ADUM1	2	PBAR	3	CRIGDR	9
ADUM2	2	PCONEAX	3	MPC	9
ADUM3	2	PDUM1	3	MPCADD	9
ADUM4	2	PDUM2	3	MPCAX	9
ADUM5	2	PDUM3	3	MPCS	9
ADUM6	2	PDUM4	3	SPC	10
ADUM7	2	PDUM5	3	SPC1	10
ADUM8	2	PDUM6	3	SPCADD	10
ADUM9	2	PDUM7	3	SPCAX	10
BAPOR	2	PDUM8	3	SPCS	10
CBAR	2	PDUM9	3	ASET	11
CCONEAX	2	PIHEX	3	ASET1	11
CDUM1	2	PQDMEM	3	DMIT	11
CDUM2	2	PQDMEM1	3	DMIT1	11
CDUM3	2	PQDMEM2	3	DMITAX	11
CDUM4	2	PQDMEM3	3	PARAM	12
CDUM5	2	PQDPLT	3	SUPAX	12
CDUM6	2	PQUAD1	3	SUPRT	12
CDUM7	2	PQUAD2	3	TEMP	13
CDUM8	2	PQUADTS	3	TEMPAX	13
CDUM9	2	PROD	3	TEMPO	13
CFLUID2	2	PSHEAR	3	TEMPP1	13
CFLUID3	2	PTORDRG	3	TEMPP2	13
CFLUID4	2	PTRAPAX	3	TEMPP3	13
CHEXA1	2	PTRBSC	3	TEMPRB	13
CHEXA2	2	PTRIA1	3	WTMASS	14
CIHEX1	2	PTRIA2	3	GRDPNT	15

RIGID FORMAT RESTART TABLES

Card Name Bit Pos.

PLOTEL	16
PLOTS	18
POUTS	19
ADUTS	21
LOOPS	22
LOOP13	23
COUPMASS	24
CPBAR	24
CPQDPLT	24
CPQUAD1	24
CPQUAD2	24
CPRDD	24
CPTRBSC	24
CPTRIA1	24
CPTRIA2	24
CPTRPLT	24
CPTUBE	24
NOLOPS	25
BDYLIST	52
FLSYM	52
G	56
EPCINT	57
SEQEP	57
TF	57
CVISC	58
PDAMP	59
PVISC	59
DMIAX	60
DMIG	60
B2PPS	60
K2PPS	60
M2PPS	60
TF3	60
EIGC	61
EIGP	61
CMETHODS	62

DIRECT COMPLEX EIGENVALUE ANALYSIS

7.8.2 Bit Positions for File Name Restart Table

<u>File Name</u>	<u>Bit Pos.</u>	<u>File Name</u>	<u>Bit Pos.</u>
BGPDT	94	DQPC1	114
CSTM	94	BDPCOL	118
EQEXIN	94	ABFL	119
GPD	94	KBFL	119
GPL	94	ELSETS	120
SIL	94	GPSETS	120
ECT	95	PLTPAR	120
GPTT	96	PLTSETX	120
EST	97	MAA	121
GEI	97	BAA	122
GPECT	97	K4AA	123
GPST	98	BDICT	124
KGGX	98	BELM	124
MGG	99	K4DICT	124
KGG	100	K4ELM	124
ASET	101	KDICT	124
RG	101	KELM	124
USET	101	MDICT	124
YS	101	MELM	124
DGPST	102	BGG	125
GM	103	K4GG	126
BNN	104		
K4NN	104		
KNN	104		
MNN	104		
BFF	105		
K4FF	105		
KFF	105		
KFS	105		
MFF	105		
GO	106		
KAA	106		
EED	107		
EQDYN	107		
GPLD	107		
SILD	107		
TFPOOL	107		
USETD	107		
CASEXX	108		
B2PP	109		
K2PP	109		
M2PP	109		
B2DD	110		
BDD	110		
GMD	110		
GDD	110		
K2DD	110		
KDD	110		
M2DD	110		
MOD	110		
CLAMA	111		
LCEIGS	111		
PHID	111		
OPHID	112		
CPHIP	113		
QPC	113		
DCPHIP	114		
QFPC1	114		
QESC1	114		

RIGID FORMAT RESTART TABLES

7.8.3 Card Name Restart Table

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
BEGIN	1234567890123456	89	1234			2	6789012
FILE	1234567890123456	89	1234			2	6789012
GP1	1						
SAVE	1						
PURGE	1						
CHKPNT	1						
\$\$\$		6					
COND	1						6
GP2	12 45		6				9
CHKPNT	12 45		6				
\$\$\$		6					
PARAML			8				
\$\$\$		7					
PURGE			8				
\$\$\$		7					
COND			8				
\$\$\$		7					
PLTSET			8				
\$\$\$		7					
SAVE			8				
\$\$\$		7					
PRTMSG			8				
\$\$\$		7					
PARAM			8				
\$\$\$		7					
PARAM			8				
\$\$\$		7					
COND			8				
\$\$\$		7					
PLOT			8				
\$\$\$		7					
SAVE			8				
\$\$\$		7					
PRTMSG			8				
\$\$\$		7					
LABEL			8				
\$\$\$		7					
CHKPNT			8				
\$\$\$		67					
GP3	1		3				
CHKPNT	1		3				
\$\$\$		6					
TA1	1234567		3				39
SAVE	1234567		3				39
PURGE	1234567		3				39
CHKPNT	1234567		3				39
\$\$\$		6					
COND	123 5678		345		4		
PARAM	123 6 8						
PAPAM	123 5 78		4		4		
PARAM	123 8						
PARAM	123 6 8						
EMG	123 5678						
SAVE	123 5678						
CHKPNT	123 5678						
\$\$\$		6					
COND	123 6 8						
EMA	123 6 8						
CHKPNT	123 6 8						

DIRECT COMPLEX EIGENVALUE ANALYSIS

DMAP Inst.	1	10	20	<u>Bit Position</u> 30	40	50	60
SSS		6					
LABEL	123	6 8					
COND	123	5 7 8	4	4			
EMA	123	5 7 8	4	4			
CHKPNT	123	5 7 8	4	4			
SSS		6					
LABEL	123	5 7 8	4	4			
COND	123	8					89
EMA	123	8					89
CHKPNT	123	8					89
SSS		6					
LABEL	123	8					89
COND	123	8					
EMA	123	6 8					
CHKPNT	123	6 8					
SSS		6					
LABEL	123	6 8					
PURGE	123	5 7 8	4	4			
PURGE	123	8					89
CHKPNT	123	5 7 8	4	4			89
SSS		6					
COND	123	5 7 8	45	4			
SSS		8					
COND	123	5 7 8	45	4			
SSS		8					
GPWG	123	5 7 8	45	4			
SSS		8					
DFP	123	5 7 8	45	4			
SSS		8					
LABEL	123	5 7	45	4			
EQUIV	1234	6 8					
CHKPNT	1234	6 8					
SSS		6					
COND	1234	6 8					
SMA3	1234	6 8					
CHKPNT	1234	6 8					
SSS		6					
LABEL	1234	6 8					
PARAM	1	901					
GP4	1	901					
SAVE	1	901					
PURGE	1	901					
CHKPNT	12345678901		4	4			
SSS		6					
COND	123	6 8 0					
COND	123	6 8 9 0					
GPSP	123	6 8 9 0					
SAVE	123	6 8 9 0					
COND	123	6 8 9 0					
DFP	123	6 8 9 0					
LABEL	123	6 8 9 0					
EQUIV	123456789		4	4			
CHKPNT	123456789		4	4			
SSS		6					
COND	123456789		4	4			
MCE1	1	9					
CHKPNT	1	9					
SSS		6					
MCE2	123456789		4	4			89

RIGID FORMAT RESTART TABLES

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
CHKPNT	123456789	4	4				89
SSS	5						
LABEL	123456789	4	4				89
EQUIV	1234567890	4	4				89
CHKPNT	1234567890	4	4				89
SSS	6						
COND	1234567890	4	4				89
SCE1	1234567890	4	4				89
CHKPNT	1234567890	4	4				89
SSS	6						
LABEL	1234567890	4	4				89
EQUIV	12345678901	4	4				89
CHKPNT	12345678901	4	4				89
SSS	6						
COND	12345678901	4	4				89
SMP1	1234 5 8901						
CHKPNT	1234 5 8901						
SSS	6						
COND	12345678901	4	4				
SMP2	12345678901	4	4				
CHKPNT	12345678901	4	4				
SSS	6						
LABEL	12345678901	4	4				
COND	1234 6 8901						89
SMP2	1234 6 8901						89
CHKPNT	1234 6 8901						89
SSS	6						
LABEL	1234 6 8901						89
COND	1234 6 8901						
SMP2	1234 6 8901						
CHKPNT	1234 6 8901						
SSS	6						
LABEL	12345678901	4	4				89
OPD	1 901						7 1
SAVE	1 901						7 1
EQUIV	12345678901	4	234		2	67890	
CHKPNT	1 901						7 1
SSS	6						
PARAM			23				
PARAM	12345678901234 6	9	123		2	6789012	
BMG	1				2		
SAVE	1				2		
PARAM	1		23		2	7 0	
PURGE	1				2		
COND	1				2		
MTRXIN	1				2		
SAVE	1				2		
LABEL	1				2		
CHKPNT	1				2		
SSS	6						
PARAM			9				
JUMP			23				
SSS	1 3						
LABEL	1234567890123456	89	123		2	6789012	
SSS	1 3						
PURGE			9	123			
CASE	1234567890123456	9	123	5	2	6789012	
SAVE	1234567890123456	9	123	5	2	6789012	
CHKPNT	1234567890123456	9	123	5	2	6789012	

DIRECT COMPLEX EIGENVALUE ANALYSIS

DMAP	Bit Position						
Inst.	1	10	20	30	40	50	60
\$\$\$		6					
MTRXIN	1		23			2	7 C
SAVE	1		23			2	7 0
PARAM	1		23			2	7 0
PARAM	1		23			2	7 0
EQUIV	1		23			2	7 0
ADD5	1		23			2	7 0
COND	1		23			2	7 0
TRNSP	1		23			2	7 0
ADD	1		23			2	7 0
LABEL	1		23			2	7 0
PARAM	1		23			2	7 0
PARAM	1		23			2	7 0
PARAM	1		23			2	7 0
PURGE	12345678901	4	234			2	67890
EQUIV	12345678901	4	234			2	67890
CHKPNT	12345678901	4	234			2	67890
\$\$\$		6					
COND	12345678901	4	234			2	67890
GKAD	12345678901	4	234			2	67890
LABEL	12345678901	4	234			2	67890
EQUIV	12345678901	4	234			2	67890
CHKPNT	12345678901	4	234			2	67890
\$\$\$		6					
COND	12345678901	4	234			2	67890
CEAD	12345678901	4	234			2	6789012
SAVE	12345678901	4	234			2	6789012
CHKPNT	12345678901	4	234			2	6789012
\$\$\$		6					
DFP	12345678901	4	234			2	6789012
SAVE	12345678901	4	234			2	6789012
COND			9 1				
VDR			9 1				
SAVE			9 1				
COND			1				
DFP			1				
SAVE			1				
LABEL			1				
COND	12345678901	4	234			2	6789012
EQUIV	12345678901	4	234			2	6789012
COND	12345678901	4	234			2	6789012
SDR1	12345678901	4	234			2	6789012
LABEL	12345678901	4	234			2	6789012
CHKPNT	12345678901	4	234			2	6789012
\$\$\$		6					
SDR2			9				
DFP			9				
SAVE			9				
LABEL	12345678901		234				
COND			23			2	6789012
\$\$\$	1 3						
REPT			23			2	6789012
\$\$\$	1 3						
JUMP			23			2	
\$\$\$	1 3						
JUMP	1234567890123456 89	1234					6789012
LABEL			23				
\$\$\$	1 3						

RIGID FORMAT RESTART TABLES

DMAP Inst.	1	10	20	<u>Bit Position</u> 30	40	50	60
PRTPARM				23		2	6789012
\$\$\$	1 3						
LABEL	1234567890123456	89	1234			2	6789012
PRTPARM	1234567890123456	89	1234			2	6789012
LABEL	1234567890123456	89	1234			2	6789012
\$\$\$	8						
PRTPARM	1234567890123456	89	1234			2	6789012
\$\$\$	8						
LABEL	1234567890123456	89	1234			2	6789012
END	1234567890123456	89	1234			2	6789012

DIRECT COMPLEX EIGENVALUE ANALYSIS

7.8.4 Rigid Format Change Restart Table

DMAP	Bit Position			DMAP	Bit Position		
Inst.	63	70	80	Inst.	63	70	80
BEGIN		345678	01234567	345	OFF		
FILE		345678	01234567	345	LABEL		
GP1				EQUIV			
SAVE				CHKPNT			
PURGE				COND			
CHKPNT				SMA3			
COND				CHKPNT			
GP2				LABEL			
CHKPNT				PARAM			
PARAM				GP4			
PURGE				SAVE			
COND				PURGE			
PLTSET				CHKPNT			
SAVE				COND			
PRTMSG				COND			
PARAM				GPSP			
PARAM				SAVE			
COND				COND			
PLOT				OFF			
SAVE				LABEL			
PRTMSG				EQUIV			
LABEL				CHKPNT			
CHKPNT				COND			
GP3				MCE1			
CHKPNT				CHKPNT			
TA1				MCE2			
SAVE				CHKPNT			
PURGE				LABEL			
CHKPNT				EQUIV			
COND				CHKPNT			
PARAM				COND			
PARAM	3	678		SCE1			
PARAM	3	678		CHKPNT			
PARAM				LABEL			
EMG	3	678		EQUIV			
SAVE	3	678		CHKPNT			
CHKPNT	3	678		COND			
COND				SMP1			
EMA				CHKPNT			
CHKPNT				COND			
LABEL				SMP2			
COND	3	678		CHKPNT			
EMA	3	678		LABEL			
CHKPNT	3	678		COND			
LABEL	3	678		SMP2			
COND	3	678		CHKPNT			
EMA	3	678		LABEL			
CHKPNT	3	678		COND			
LABEL	3	678		SMP2			
COND				CHKPNT			
EMA				LABEL			
CHKPNT				DPD			
LABEL				SAVE			
PURGE	3	678		EQUIV			
PURGE	3	678		CHKPNT			
CHKPNT	3	678		PARAM			
COND				PARAM	345678	01234567	345
COND				BMG			
GPWG				SAVE			

RIGID FORMAT RESTART TABLES

DMAP Inst.	63	Bit Position 70	80
PARAM			
PURGE			
COND			
MTRX IN			
SAVE			
LABEL			
CHKPNT			
PARAM			
JUMP	345678	01234567	345
LABEL	345678	01234567	345
PURGE			
CASE	345678	01234567	345
SAVE	345678	01234567	345
CHKPNT	345678	01234567	345
MTRX IN			
SAVE			
PARAM			
PARAM			
EQUIV			
ADD5			
COND			
TRNSP			
ADD			
LABEL			
PARAM			
PARAM			
PURGE			
EQUIV			
CHKPNT			
COND		1	
GKAD		1	
LABEL		1	
EQUIV			
CHKPNT			
COND	345678	01234567	345
CEAD			
SAVE			
CHKPNT			
OFF			
SAVE			
COND			
VDR			
SAVE			
COND			
OFF			
SAVE			
LABEL			
COND	345678	01234567	345
EQUIV	345678	01234567	345
COND	345678	01234567	345
SDR1	345678	01234567	345
LABEL			
CHKPNT			
SCP2			
OFF			
SAVE			
LABEL			
COND	345678	01234567	345
PEPT	345678	01234567	345

DMAP Inst.	63	Bit Position 70	80
JUMP	345678	01234567	345
JUMP	345678	01234567	345
LABEL	345678	01234567	345
PRTPARM	345678	01234567	345
LABEL	345678	01234567	345
PRTPARM	345678	01234567	345
LABEL	345678	01234567	345
PRTPARM	345678	01234567	345
LABEL	345678	01234567	345
END	345678	01234567	345

DIRECT COMPLEX EIGENVALUE ANALYSIS

7.8.5 File Name Restart Table

Bit Position				Bit Position			
Inst.	94	100	110	Inst.	94	100	110
BEGIN				GPWG			
FILE				DFP			
GP1	4			LABEL	8		
SAVE	4			EQUIV	0		
PURGE				CHKPNT	0		
CHKPNT				COND	0		
COND				SMA3	0		
GP2	5			CHKPNT	0		
CHKPNT	5			LABEL	0		
PARAML			0	PARAM	1		
PURGE			0	GP4	1		
COND			0	SAVE	1		
PLTSET			0	PURGE	1 3 56	0 3	
SAVE			0	CHKPNT	1 3 56	0 3	
PRTMSG			0	COND	2		
PARAM			0	COND	2		
PARAM			0	GPSP	2		
COND			0	SAVE	2		
PLOT				COND	2		
SAVE				DFP	2		
PRTMSG				LABEL	2		
LABEL				EQUIV		4	
CHKPNT			0	CHKPNT		4	
GP3	6			COND		34	
CHKPNT	6			MCE1		3	
TA1	7			CHKPNT		3	
SAVE	7			MCE2		4	
PURGE	7 8 9	2 4 5 6		CHKPNT		4	
CHKPNT	7		123	LABEL		34	
COND	8			EQUIV		5	
PARAM	8			CHKPNT		5	
PARAM	9			COND		5	
PARAM			5	SCE1		5	
PARAM			6	CHKPNT		5	
EMG			4	LABEL		5	
SAVE			4	EQUIV		6	123
CHKPNT			4	CHKPNT		6	123
COND	8			COND		6	123
EMA	8			SMP1		6	
CHKPNT	8			CHKPNT		6	
LABEL	8			COND			1
COND	9			SMP2			1
EMA	9			CHKPNT			1
CHKPNT	9			LABEL			1
LABEL	9			COND			2
COND			5	SMP2			2
EMA			5	CHKPNT			2
CHKPNT			5	LABEL			2
LABEL			5	COND			3
COND			6	SMP2			3
EMA			6	CHKPNT			3
CHKPNT			5	LABEL	6		123
LABEL			6	DPD	7		
PURGE	9			SAVE	7		
PURGE			5	EQUIV		0	
CHKPNT	9			CHKPNT	7		
COND				PARAM			
COND				PARAM		8	

RIGID FORMAT RESTART TABLES

DMAP Inst.	94	Bit Position		120	DMAP Inst.	94	Bit Position		120
		100	110				100	110	
BMG				8	COND				
SAVE				9	REPT				
PARAM			9	9	JUMP				
PURGE				8	JUMP				
COND				9	LABEL				
MTRXIN				9	PRTPARM				
SAVE				9	LABEL				
LABEL				9	PRTPARM				
CHKPNT				9	LABEL				
PARAM					PRTPARM				
JUMP			9		LABEL				
LABEL			8		END				
PURGE									
CASE			8						
SAVE			8						
CHKPNT			8						
MTRXIN			9						
SAVE			9						
PARAM			9						
PARAM			9						
EQUIV			9						
ADD5			9						
COND			9						
TRNSP			9						
ADD			9						
LABEL			9						
PARAM			9						
PARAM			9						
PARAM			9						
PURGE			0						
EQUIV			0						
CHKPNT			0						
COND			0						
GKAD			0						
LABEL			0						
EQUIV			1						
CHKPNT			1						
COND			1						
CEAD			1						
SAVE			1						
CHKPNT			1						
DFP			1						
SAVE			1						
COND			2						
VDR			2						
SAVE			2						
COND			2						
DFP			2						
SAVE			2						
LABEL			2						
COND			3						
EQUIV			3						
COND			3						
SDR1			3						
LABEL			3						
CHKPNT			3						
SDR2			4						
DFP									
SAVE									
LABEL			23						

DIRECT FREQUENCY AND RANDOM RESPONSE

7.9 RESTART TABLES FOR DIRECT FREQUENCY AND RANDOM RESPONSE

7.9.1 Bit Positions for Card Name Restart Table

Card Name	Bit Pos.	Card Name	Bit Pos.	Card Name	Bit Pos.
AXIC	1	CIHEX1	2	PTRIA1	3
AXIF	1	CIHEX2	2	PTRIA2	3
CDAMP1	1	CIHEX3	2	PTRIAAX	3
CDAMP2	1	CONROD	2	PTRIATS	3
CDAMP3	1	CQDMEM	2	PTRIM6	3
CDAMP4	1	CQDMEM1	2	PTPMEM	3
CELAS1	1	CQDMEM2	2	PTRPLT	3
CELAS2	1	CQDMEM3	2	PTRPLT1	3
CELAS3	1	CQOPLT	2	PTRSHL	3
CELAS4	1	CQUAD1	2	PTUBE	3
CMASS1	1	CQUAD2	2	PTWIST	3
CMASS2	1	CQUADTS	2	GENEL	4
CMASS3	1	CROD	2	CONM1	5
CMASS4	1	CSHEAR	2	CONM2	5
CORD1C	1	CTETRA	2	FSLIST	5
CORD1R	1	CTORDRG	2	PELAS	6
CORD1S	1	CTRAPAX	2	PMASS	7
CORD2C	1	CTRAPRG	2	MAT1	8
CORD2R	1	CTRBSC	2	MAT2	8
CORD2S	1	CTRIA1	2	MAT3	8
FREETPT	1	CTRIA2	2	MATT1	8
GRDSET	1	CTRIAAX	2	MATT2	8
GRID	1	CTRIARG	2	MATT3	8
GRIDB	1	CTRIATS	2	TABLEM1	8
POINTAX	1	CTRIM6	2	TABLEM2	8
PRESPT	1	CTRMEM	2	TABLEM3	8
RINGAX	1	CTRPLT	2	TABLEM4	8
RINGFL	1	CTRPLT1	2	TEMPMT\$	8
SECTAX	1	CTRSHL	2	TEMPMX\$	8
SEGGP	1	CTUBE	2	AXISYM\$	9
SPOINT	1	CTWIST	2	CRIGD1	9
ADUM1	2	CWEDGE	2	CRIGD2	9
ADUM2	2	PBAR	3	CRIGD3	9
ADUM3	2	PCONEAX	3	CRIDGR	9
ADUM4	2	PDUM1	3	MPC	9
ADUM5	2	PDUM2	3	MPCADD	9
ADUM6	2	PDUM3	3	MPCAX	9
ADUM7	2	PDUM4	3	MPC\$	9
ADUM8	2	PDUM5	3	SPC	10
ADUM9	2	PDUM6	3	SPC1	10
BAROR	2	PDUM7	3	SPCADD	10
CBAR	2	PDUM8	3	SPCAX	10
CCONEAX	2	PDUM9	3	SPC\$	10
CDUM1	2	PIHEX	3	ASET	11
CDUM2	2	PQDMEM	3	ASET1	11
CDUM3	2	PQDMEM1	3	OMIT	11
CDUM4	2	PQDMEM2	3	OMIT1	11
CDUM5	2	PQDMEM3	3	OMITAX	11
CDUM6	2	PQDPLT	3	PARAM	12
CDUM7	2	PQUAD1	3	SUPAX	12
CDUM8	2	PQUAD2	3	SUPORT	12
CDUM9	2	PQUADTS	3	TEMP	13
CFLUID2	2	PROD	3	TEMPAX	13
CFLUID3	2	PSHEAR	3	TEMPO	13
CFLUID4	2	PTORDRG	3	TEMPP1	13
CHEXA1	2	PTRAPAX	3	TEMPP2	13
CHEXA2	2	PTRBSC	3	TEMPP3	13

RIGID FORMAT RESTART TABLES

Card Name	Bit Pos.
-----------	----------

TEMPRB	13
WTMASS	14
GRDPNT	15
PLOTEL	16
PLOTS	18
POUTS	19
XYOUTS	20
AOUTS	21
LOOP\$	22
LOOP1\$	23
COUPMASS	24
CPBAR	24
CPQOPLT	24
CPQUAD1	24
CPQUAD2	24
CPROD	24
CPTRBSC	24
CPTRIA1	24
CPTRIA2	24
CPTRPLT	24
CPTUBE	24
NOLLOOP\$	25
RANDOM\$	26
AXYOUT\$	27
BOYLIST	52
FLSYM	52
RANDPS	55
RANDT1	55
RANDT2	55
TABRND1	54
TABRND2	54
TABRND3	54
TABRND4	54
G	56
EPOINT	57
SEQEP	57
TF	57
CVISC	58
PDAMP	59
PVISC	59
B2PP\$	60
DMIAx	60
D4IG	60
K2PP\$	60
M2PP\$	60
TF\$	60
DAREA	61
DELAY	61
OLOAD	61
OPHASE	61
FREQ	61
FREQ1	61
FREQ2	61
RLOAD1	61
RLOAD2	61
TABLED1	61
TABLED2	61
TABLED3	61
TABLED4	61
DECOMOPT	62
DLOAD\$	62
FREQ\$	62

DIRECT FREQUENCY AND RANDOM RESPONSE

7.9.2 Bit Positions for File Name Restart Table

<u>File Name</u>	<u>Bit Pos.</u>	<u>File Name</u>	<u>Bit Pos.</u>
BGPD	94	MDD	110
CSTM	94	PDF	111
EQEXIN	94	PPF	111
GPD	94	PSF	111
GPL	94	UDVF	111
SIL	94	OUVVC1	112
ECT	95	OUVVC2	113
GPTT	96	QPC	114
EST	97	UPVC	114
GEI	97	OEFC1	115
GPECT	97	OESC1	115
GPST	98	OPPC1	115
KGGX	98	OQPC1	115
MGG	99	OUPVC1	115
KGG	100	OEFC2	116
ASET	101	OESC2	116
RG	101	OPPC2	116
USET	101	OQPC2	116
YS	101	OUPVC2	116
OGPST	102	AUTO	117
GM	103	PSDF	117
BNN	104	BDPOOL	118
K4NN	104	ABFL	119
KNN	104	KBFL	119
MNN	104	ELSETS	120
BFF	105	GPSETS	120
K4FF	105	PLTPAR	120
KFF	105	PLTSETX	120
KFS	105	MAA	121
MFF	105	BAA	122
GO	106	K4AA	123
KAA	106	BDICT	124
DLT	107	BELM	124
EQDYN	107	K4DICT	124
FRL	107	K4ELM	124
GPLD	107	KDICT	124
PSDL	107	KELM	124
SILD	107	MDICT	124
TFPOOL	107	MELM	124
USETD	107	BGG	125
CASEXX	108	K4GG	126
B2PP	109		
K2PP	109		
M2PP	109		
B2DD	110		
BDD	110		
GMD	110		
GDD	110		
K2DD	110		
KDD	110		
M2DD	110		

RIGID FORMAT RESTART TABLES

7.9.3 Card Name Restart Table

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
BEGIN	1234567890123456	8901234				2 456789012	
FILE	1234567890123456	8901234				2 456789012	
GP1	1						
SAVE	1						
PURGE	1						
CHKPNT	1						
\$SS		6					
COND	1						
GP2	12 45		6				8
CHKPNT	12 45		6				8
\$SS		6					
PARAML			8				
\$SS		7					
PURGE			8				
\$SS		7					
COND			8				
\$SS		7					
PLTSET			8				
\$SS		7					
SAVE			8				
\$SS		7					
PRTMSG			8				
\$SS		7					
PARAM			8				
\$SS		7					
PARAM			8				
\$SS		7					
COND			8				
\$SS		7					
PLOT			8				
\$SS		7					
SAVE			8				
\$SS		7					
PRTMSG			8				
\$SS		7					
LABEL			8				
\$SS		7					
CHKPNT			8				
\$SS		67					
GP3	1		3				
CHKPNT	1		3				
\$SS		6					
TA1	1234567		3				39
SAVE	1234567		3				89
PUPGE	1234567		3				89
CHKPNT	1234567		3				89
\$SS		6					
COND	123 5678		345		4		
PARAM	123 6 8						
PARAM	123 5 7 8		4		4		
PAPAM	123 8						
PARAM	123 6 8						
EMG	123 5678						
SAVE	123 5678						
CHKPNT	123 5678						
\$SS		6					
COND	123 6 8						
EMA	123 6 8						

DIRECT FREQUENCY AND RANDOM RESPONSE

DMAP Inst.	<u>Bit Position</u>						
	1	10	20	30	40	50	60
CHKPNT	123	6 8					
SSS		6					
LABEL	123	6 8					
COND	123	5 7 8	4	4			
EMA	123	5 7 8	4	4			
CHKPNT	123	5 7 8	4	4			
SSS		6					
LABEL	123	5 7 8	4	4			
COND	123	8					89
EMA	123	8					89
CHKPNT	123	8					89
SSS		6					
LABEL	123	8					89
COND	123	8					
EMA	123	6 8					
CHKPNT	123	6 8					
SSS		6					
LABEL	123	6 8					
PURGE	123	5 7 8	4	4			
PURGE	123	8					89
CHKPNT	123	5 7 8	4	4			89
SSS		6					
COND	123	5 7 8	45	4			
SSS		8					
COND	123	5 7 8	45	4			
SSS		8					
GPWG	123	5 7 8	45	4			
SSS		8					
OFF	123	5 7 8	45	4			
SSS		8					
LABEL	123	5 7 8	45	4			
EQUIV	1234	6 8					
CHKPNT	1234	6 8					
SSS		6					
COND	1234	6 8					
SMA3	1234	6 8					
CHKPNT	1234	6 8					
SSS		5					
LABEL	1234	6 8					
PARAM	1	901					
GP4	1	901					
SAVE	1	901					
PURGE	1	901					
CHKPNT	12345678901	4	4				
SSS		5					
COND	123	6 8 0					
COND	123	6 8 90					
GPSP	123	6 8 90					
SAVF	123	6 8 90					
COND	123	6 8 90					
OFF	123	6 8 90					
LABEL	123	6 8 90					
EQUIV	123456789	4	4				
CHKPNT	123456789	4	4				
SSS		6					
COND	123456789	4	4				
MCE1	1	9					
CHKPNT	1	9					

RIGID FORMAT RESTART TABLES

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
\$SS	6						
MCE2	123456789	4	4				89
CHKPNT	123456789	4	4				89
\$SS	6						
LABEL	123456789	4	4				89
EQUIV	1234567890	4	4				89
CHKPNT	1234567890	4	4				89
\$SS	6						
COND	1234567890	4	4				89
SCE1	1234567890	4	4				89
CHKPNT	1234567890	4	4				89
\$SS	6						
LABEL	1234567890	4	4				89
EQUIV	1234 6 8901						
EQUIV	12345 78901	4	4				
EQUIV	1234 8901						89
EQUIV	1234 6 8901						
CHKPNT	12345678901	4	4				89
\$SS	6						
COND	12345678901	4	4				89
SMP1	1234 6 8901						
CHKPNT	1234 6 8901						
\$SS	6						
COND	12345678901	4	4				
SMP2	12345678901	4	4				
CHKPNT	12345678901	4	4				
\$SS	6						
LABEL	12345678901	4	4				
COND	1234 6 8901						89
SMP2	1234 6 8901						89
CHKPNT	1234 6 8901						89
\$SS	6						
LABEL	1234 6 8901						89
COND	1234 6 8901						
SMP2	1234 6 8901						
CHKPNT	1234 6 8901						
\$SS	6						
LABEL	12345678901	4	4				89
OPD	1 901					5 7	1
SAVE	1 901					5 7	1
EQUIV	12345678901	4	234		2	67890	
CHKPNT	1 901					5 7	1
\$SS	6						
PARAM			23				
PARAM	12345678901234 6	90123	7		2	456789012	
BHG	1				2		
SAVE	1				2		
PARAM	1		23		2	7 0	
PURGE	1				2		
COND	1				2		
MTRXIN	1				2		
SAVE	1				2		
LABEL	1				2		
CHKPNT	1				2		
\$SS	6						
PARAM			9 1				
JUMP			23				
\$SS	1 3						
LABEL	1234567890123456	890123			2	6789012	

DIRECT FREQUENCY AND RANDOM RESPONSE

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
\$SS	1 3						
PURGE			90123	7			
CASE	12345678901234	6	90123	5 7		2	456789012
SAVE	12345678901234	6	90123	5 7		2	456789012
CHKPNT	12345678901234	6	90123	5 7		2	456789012
\$SS	6						
MTRXIN	1		23			2	7 0
SAVE	1		23			2	7 0
PARAM	1		23			2	7 0
PARAM	1		23			2	7 0
EQUIV	1		23			2	7 0
ADD5	1		23			2	7 0
COND	1		23			2	7 0
TRNSP	1		23			2	7 0
ADD	1		23			2	7 0
LABEL	1		23			2	7 0
PARAM	1		23			2	7 0
PARAM	1		23			2	7 0
PARAM	1		23			2	7 0
PURGE	12345678901	4	234			2	67890
EQUIV	12345678901	4	234			2	67890
CHKPNT	12345678901	4	234			2	67890
\$SS	6						
COND	12345678901	4	234			2	67890
GKAD	12345678901	4	234			2	67890
LABEL	12345678901	4	234			2	67890
EQUIV	12345678901	4	234			2	67890
CHKPNT	12345678901	4	234			2	67890
\$SS	6						
COND	12345678901	4	234			2	6789012
COND	12345678901	4	234			2	6789012
FRRO	12345678901	4	234			2	6789012
EQUIV	12345678901	4	234			2	6789012
CHKPNT	12345678901	4	234			2	6789012
\$SS	6						
VDR			901	7			
SAVE			901	7			
COND			1	7			
COND			1	7			
CHKPNT			1	7			
\$SS	6						
SDR3			1	7			
OFF			1				
SAVE			1				
CHKPNT			1	7			
\$SS	6						
XYTRAN				7			
\$SS	7						
SAVE				7			
\$SS	7						
XYPLOT				7			
\$SS	7						
JUMP			1	7			
LABEL			1	7			
OFF			1				
SAVE			1				
LABEL			1	7			
COND	12345678901	4	234			2	6789012
EQUIV	12345678901	4	234			2	6789012

RIGID FORMAT RESTART TABLES

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
COND	12345678901	4	234			2	6789012
SDP1	12345678901	4	234			2	6789012
LABEL	12345678901	4	234			2	6789012
CHKPNT	12345678901	4	234			2	6789012
\$\$\$	6						
SDR2			90				
SAVE			90				
COND			90				
SDR3			90				
CHKPNT			90				
\$\$\$	6						
OFF			9				
SAVE			9				
XYTRAN			0				
\$\$\$	7						
SAVE			0				
\$\$\$	7						
XYPLOT			0				
\$\$\$	7						
COND			0			45	
\$\$\$	7						
RANDOM				6		45	
\$\$\$	7						
SAVE				6		45	
\$\$\$	7						
CHKPNT				6			
\$\$\$	67						
COND			0	6		45	
\$\$\$	7						
XYTRAN			0				
\$\$\$	7						
SAVE			0				
\$\$\$	7						
XYPLOT			0				
\$\$\$	7						
JUMP			0				
LABEL			9				
OFF			9				
SAVE			9				
LABEL			0				
COND			23				
\$\$\$	1 3						
REPT			23				
\$\$\$	1 3						
JUMP			23				
\$\$\$	1 3						
JUMP	1234567890123456	8901234				2	6789012
LABEL			23				
\$\$\$	1 3						
PRTPARM			23				
\$\$\$	1 3						
LABEL	1234567890123456	8901234				2	6789012
PRTPARM	1234567890123456	8901234				2	6789012
LABEL	1234567890123456	8901234				2	6789012
PRTPARM	1234567890123456	8901234				2	6789012
LABEL	1234567890123456	8901234				2	6789012
\$\$\$	8						
PRTPARM	1234567890123456	8901234				2	6789012
\$\$\$	8						
LABEL	1234567890123456	8901234				2	6789012
END	1234567890123456	8901234				2	6789012

DIRECT FREQUENCY AND RANDOM RESPONSE

7.9.4 Rigid Format Change Restart Table

DMAP Inst.	63	Bit Position 70	80	DMAP Inst.	63	Bit Position 70	80
BEGIN		3456789 1234567	345	GPWG			
FILE		3456789 1234567	345	OFF			
GP1				LABEL			
SAVE				EQUIV			
PURGE				CHKPNT			
CHKPNT				COND			
COND				SMA3			
GP2				CHKPNT			
CHKPNT				LABEL			
PARAML				PARAM			
PURGE				GP4			
COND				SAVE			
PLTSET				PURGE			
SAVE				CHKPNT			
PRTMSG				COND			
PARAM				COND			
PARAM				GPSP			
COND				SAVE			
PLOT				COND			
SAVE				OFF			
PRTMSG				LABEL			
LABEL				EQUIV			
CHKPNT				CHKPNT			
GP3				COND			
CHKPNT				MCE1			
TA1				CHKPNT			
SAVE				MCE2			
PURGE				CHKPNT			
CHKPNT				LABEL			
COND				EQUIV			
PARAM				CHKPNT			
PARAM	3	678		COND			
PARAM	3	678		SCE1			
PARAM				CHKPNT			
EMG	3	678		LABEL			
SAVE	3	678		EQUIV			
CHKPNT	3	678		EQUIV			
COND				EQUIV			
EMA				EQUIV			
CHKPNT				CHKPNT			
LABEL				COND			
COND	3	678		SMP1			
EMA	3	678		CHKPNT			
CHKPNT	3	678		COND			
LABEL	3	678		SMP2			
COND	3	678		CHKPNT			
EMA	3	678		LABEL			
CHKPNT	3	678		COND			
LABEL	3	678		SMP2			
COND				CHKPNT			
EMA				LABEL			
CHKPNT				COND			
LABEL				SMP2			
PURGE	3	678		CHKPNT			
PURGE	3	678		LABEL			
CHKPNT	3	678		DPD			
COND				SAVE			
COND				EQUIV			

RIGID FORMAT RESTART TABLES

DMAP
Inst. 63 Bit Position 70 80

CHKPNT			
PARAM			
PARAM	3456789	1234567	345
BMG			
SAVE			
PARAM			
PURGE			
COND			
MTRXIN			
SAVE			
LABEL			
CHKPNT			
PARAM			
JUMP	3456789	1234567	345
LABEL	3456789	1234567	345
PURGE			
CASE	3456789	1234567	345
SAVE	3456789	1234567	345
CHKPNT	3456789	1234567	345
MTRXIN			
SAVE			
PARAM			
PARAM			
EQUIV			
ADDS			
COND			
TRNSP			
ADD			
LABEL			
PARAM			
PARAM			
PARAM			
PURGE			
EQUIV			
CHKPNT			
COND		1	
GKAD		1	
LABEL		1	
EQUIV			
CHKPNT			
COND	3456789	1234567	345
CCND	3456789	1234567	345
FRRD			
EQUIV			
CHKPNT			
VDR			
SAVE			
COND			
COND			
CHKPNT			
SDR3			
OFF			
SAVE			
CHKPNT			
XYTRAN			
SAVE			
XYPLOT			
JUMP			
LABEL			

DMAP Inst.	63	<u>Bit Position</u> 70	80
OFF			
SAVE			
LABEL			
COND	3456789	1234567	345
EQUIV	3456789	1234567	345
COND	3456789	1234567	345
SDR1	3456789	1234567	345
LABEL	3456789	1234567	345
CHKPNT	3456789	1234567	345
SDP2			
SAVE			
COND			
SDR3			
CHKPNT			
OFF			
SAVE			
XYTRAN			
SAVE			
XYPLOT			
COND	3456789	1234567	345
RANDOM	3456789	1234567	345
SAVE	3456789	1234567	345
CHKPNT	3456789	1234567	345
COND			
XYTRAN			
SAVE			
XYPLOT			
JUMP			
LABEL			
OFF			
SAVE			
LABEL			
COND	3456789	1234567	345
REPT	3456789	1234567	345
JUMP	3456789	1234567	345
JUMP	3456789	1234567	345
LABEL	3456789	1234567	345
PRTPARM	3456789	1234567	345
LABEL	3456789	1234567	345
PRTPARM	3456789	1234567	345
LABEL	3456789	1234567	345
PRTPARM	3456789	1234567	345
LABEL	3456789	1234567	345
PRTPARM	3456789	1234567	345
LABEL	3456789	1234567	345
PRTPARM	3456789	1234567	345
LABEL	3456789	1234567	345
END	3456789	1234567	345

DIRECT FREQUENCY AND RANDOM RESPONSE

7.9.5 File Name Restart Table

DMAP Inst.	94	100	110	120	DMAP Inst.	94	100	110	142
BEGIN					COND				
FILE					COND				
GP1	4				GPWG				
SAVE	4				OFF				
PURGE					LABEL	8			
CHKPNT					EQUIV	0			
COND					CHKPNT	0			
GP2	5				COND	0			
CHKPNT	5				SMA3	0			
PARAML				0	CHKPNT	0			
PURGE				0	LABEL	0			
COND				0	PARAM	1			
PLTSET				0	GP4	1			
SAVE				0	SAVE	1			
PRTMSG				0	PURGE	1	3	56	01 4
PARAM				0	CHKPNT	1	3	56	01 4
PAPAM				0	COND		2		
COND					COND		2		
PLOT					GPSP		2		
SAVE					SAVE		2		
PRTMSG					COND		2		
LABEL					OFF		2		
CHKPNT				0	LABEL		2		
GP3	6				EQUIV		4		
CHKPNT	6				CHKPNT		4		
TA1	7				COND		34		
SAVE	7				MCE1		3		
PURGE	7 8 9	2	4 5 6	123	CHKPNT		3		
CHKPNT	7				MCE2		4		
COND	8				CHKPNT		4		
PARAM	8				LABEL		34		
PARAM	9				EQUIV		5		
PARAM				5	CHKPNT		5		
PARAM				6	COND		5		
EMG				4	SCE1		5		
SAVE				4	CHKPNT		5		
CHKPNT				4	LABEL		5		
COND	8				EQUIV		6		
EMA	8				EQUIV				1
CHKPNT	8				EQUIV				2
LABEL	8				EQUIV				3
COND	9				CHKPNT		6		123
EMA	9				COND		6		123
CHKPNT	9				SMP1		6		
LABEL	9				CHKPNT		6		
COND				5	COND				1
EMA				5	SMP2				1
CHKPNT				5	CHKPNT				1
LABEL				5	LABEL				1
COND				6	COND				2
EMA				6	SMP2				2
CHKPNT				6	CHKPNT				2
LABEL				6	LABEL				2
PURGE	9				COND				3
PURGE				5	SMP2				3
CHKPNT	9				CHKPNT				3

RIGID FORMAT RESTART TABLE

DMAP Inst.	94	Bit Position 100 110	120	DMAP Inst.	94	Bit Position 100 110	120
LABEL		6	123	SAVE			
OPD		7		XYPLOT			
SAVE		7		JUMP			
EQUIV		0		LABEL			
CHKPNT		7		OFF			
PARAM				SAVE			
PARAM		8		LABEL			
BMG			8	COND			4
SAVE			8	EQUIV			4
PARAM		9	9	COND			4
PURGE			8	SDR1			4
COND			9	LABEL			4
MTRXIN			9	CHKPNT			4
SAVE			9	SDR2			5
LABEL			9	SAVE			5
CHKPNT			9	COND			6
PARAM				SDR3			6
JUMP		8		CHKPNT			6
LABEL		8		OFF			
PURGE				SAVE			
CASE		8		XYTRAN			
SAVE		8		SAVE			
CHKPNT		8		XYPLOT			
MTRXIN		9		COND			7
SAVE		9		RANDOM			7
PARAM		9		SAVE			7
PARAM		9		CHKPNT			7
EQUIV		9		COND			
ADD5		9		XYTRAN			
COND		9		SAVE			
TRNSP		9		XYPLOT			
ADD		9		JUMP			
LABEL		3		LABEL			6
PARAM		9		OFF			
PARAM		9		SAVE			
PARAM		9		LABEL			4
PURGE		0		COND			
EQUIV		0		REPT			
CHKPNT		0		JUMP			
COND		0		JUMP			
GKAD		0		LABEL			
LABEL		0		PRTPARM			
EQUIV		1		LABEL			
CHKPNT		1		PRTPARM			
COND		1		LABEL			
COND		1		PRTPARM			
FRRD		1		LABEL			
EQUIV		1		PRTPARM			
CHKPNT		1		LABEL			
VDR		2		END			
SAVE		2					
COND		2					
COND		2					
CHKPNT		2					
SDR3		3					
OFF							
SAVE							
CHKPNT		3					
XYTRAN							

DIRECT TRANSIENT RESPONSE

7.10 RESTART TABLES FOR DIRECT TRANSIENT RESPONSE

7.10.1 Bit Positions for Card Name Restart Table

Card Name	Bit Pos.	Card Name	Bit Pos.	Card Name	Bit Pos.
AXIC	1	CIHEX1	2	PTRIA1	3
AXIF	1	CIHEX2	2	PTRIA2	3
CDAMP1	1	CIHEX3	2	PTRIAAX	3
CDAMP2	1	CONROD	2	PTRIAATS	3
CDAMP3	1	CQDMEM	2	PTRIM6	3
CDAMP4	1	CQDMEM1	2	PTRIM6	3
CELAS1	1	CQDMEM2	2	PTRMEM	3
CELAS2	1	CQDMEM3	2	PTRPLT	3
CELAS3	1	CQDPLT	2	PTRPLT1	3
CELAS4	1	CQUAD1	2	PTRSHL	3
CMASS1	1	CQUAD2	2	PTUBE	3
CMASS2	1	CQUADTS	2	PTWIST	3
CMASS3	1	CROD	2	GENEL	4
CMASS4	1	CSHEAR	2	CONM1	5
CORD1C	1	CTETRA	2	CONM2	5
CORD1R	1	CTORDRG	2	FSLIST	5
CORD1S	1	CTRAPAX	2	PELAS	6
CORD2C	1	CTRAPRG	2	PMASS	7
CORD2R	1	CTR8SC	2	MAT1	8
CORD2S	1	CTRIA1	2	MAT2	8
FREETPT	1	CTRIA2	2	MAT3	8
GRDSET	1	CTRIAAX	2	MATT1	8
GRID	1	CTRIARG	2	MATT2	8
GRID8	1	CTRIATS	2	MATT3	8
POINTAX	1	CTRIM6	2	TABLEM1	8
PRESPT	1	CTRMEM	2	TABLEM2	8
RINGAX	1	CTRPLT	2	TABLEM3	8
RINGFL	1	CTRPLT1	2	TABLEM4	8
SECTAX	1	CTRSHL	2	TEMPMT\$	8
SEQGP	1	CTUBE	2	TEMPMX\$	8
SPOINT	1	CTWIST	2	AXISYM\$	9
ADUM1	2	CWEDGE	2	CRIGD1	9
ADUM2	2	PBAR	3	CRIGD2	9
ADUM3	2	PCONEAX	3	CRIGD3	9
ADUM4	2	PDUM1	3	CRIGDR	9
ADUM5	2	PDUM2	3	MPC	9
ADUM6	2	PDUM3	3	MPCADD	9
ADUM7	2	PDUM4	3	MPCAX	9
ADUM8	2	PDUM5	3	MPC\$	9
ADUM9	2	PDUM6	3	SPC	10
BAROR	2	PDUM7	3	SPC1	10
CBAR	2	PDUM8	3	SPCADD	10
CCONEAX	2	PDUM9	3	SPCAX	10
CDUM1	2	PIHEX	3	SPC\$	10
CDUM2	2	PQDMEM	3	ASET	11
CDUM3	2	PQDMEM1	3	ASET1	11
CDUM4	2	PQDMEM2	3	OMIT	11
CDUM5	2	PQDMEM3	3	OMIT1	11
CDUM6	2	PODPLT	3	OMITAX	11
CDUM7	2	PQUAD1	3	SUPAX	12
CDUM8	2	PQUAD2	3	SUPPORT	12
CDUM9	2	PQUADTS	3	TEMP	13
CFLUID2	2	PROD	3	TEMPAX	13
CFLUID3	2	PSHEAR	3	TEMPO	13
CFLUID4	2	PTORDRG	3	TEMPP1	13
CHEXA1	2	PTRAPAX	3	TEMPP2	13
CHEXA2	2	PTRBSC	3	TEMPP3	13

RIGID FORMAT RESTART TABLES

Card Name Bit Pos.

TEMPRB	13
WTMASS	14
GRDPNT	15
PLOTEL	16
PLOT\$	18
POUT\$	19
XYOUT\$	20
AOUT\$	21
LOOP\$	22
LOOP1\$	23
COUPMASS	24
CPBAR	24
CPQDPLT	24
CPQUAD1	24
CPQUAD2	24
CPRD	24
CPTRBSC	24
CPTRIA1	24
CPTRIA2	24
CPTRPLT	24
CPTUBE	24
NOLLOOP\$	25
AXYOUT\$	27
BOYLIST	52
FLSYM	52
G	56
W3	56
W4	56
EPOINT	57
SEQEP	57
TF	57
CVISC	58
PDAMP	59
PVISC	59
OMIAX	60
DMIG	60
B2PP\$	60
K2PP\$	60
M2PP\$	60
TF\$	60
DAREA	61
DELAY	61
DLOAD	61
FORCE	61
FORCE1	61
FORCE2	61
GRAV	61
MOMENT	61
MOMENT1	61
MOMENT2	61
NOLIN1	61
NOLIN2	61
NOLIN3	61
NOLIN4	61
PLOAD	61
PLOAD1	61
PLOAD2	61
SLOAD	61
TABLED1	61
TABLED2	61
TABLED3	61

Card Name Bit Pos.

TABLED4	61
TIC	61
TLOAD1	61
TLOAD2	61
TSTEP	61
DLOAD\$	62
IC\$	62
NLFORCE	62
TSTEP\$	62

DIRECT TRANSIENT RESPONSE

7.10.2 Bit Positions for File Name Restart Table

<u>File Name</u>	<u>Bit Pos.</u>	<u>File Name</u>	<u>Bit Pos.</u>
BGPD	94	M2PP	109
CSTM	94	B2DD	110
EQEXIN	94	BDD	110
GPD	94	GMD	110
GPL	94	GDD	110
SIL	94	K2DD	110
ECT	95	KDD	110
GPTT	96	M2DD	110
SLT	96	MDD	110
EST	97	PD	111
GEI	97	PDT	111
GPECT	97	PPT	111
GPST	98	PST	111
KGGX	98	TOL	111
MGG	99	OUOV1	112
KGG	100	OPNL1	112
ASET	101	OUOV2	113
RG	101	OPNL2	113
USET	101	QP	114
YS	101	UPV	114
OGPST	102	OEF1	115
GM	103	OES1	115
BNN	104	OPP1	115
K4NN	104	OQP1	115
KNN	104	OUPV1	115
MNN	104	PUGV	115
BFF	105	OEF2	116
K4FF	105	OES2	116
KFF	105	OPP2	116
KFS	105	OQP2	116
MFF	105	OUPV2	116
GO	106	BDPOOL	118
KAA	106	ABFL	119
DLT	107	KBFL	119
EQDYN	107	ELSETS	120
GPLD	107	GPSETS	120
NLFT	107	PLTPAR	120
SILD	107	PLTSETX	120
TFPOOL	107	MAA	121
TRL	107	BAA	122
USETD	107	K4AA	123
CASEXX	108	BDICT	124
B2PP	109	BELM	124
K2PP	109	K4DICT	124
		K4ELM	124
		KDICT	124
		KELM	124
		MDICT	124
		HELM	124
		BGG	125
		K4GG	126
		PNLD	127
		UDVT	127

RIGID FORMAT RESTART TABLES

7.10.3 Card Name Restart Table

DMAPI	Bit Position					
Inst.	1	10	20	30	40	50 60
BEGIN	1234567890123456	8901234				2 6789012
FILE	1234567890123456	8901234				2 6789012
GP1	1					
SAVE	1					
PURGE	1					
CHKPNT	1					
SSS	6					
COND	1		9 1			1
GP2	12 45		6			8
CHKPNT	12 45		6			8
SSS	6					
PARAML			8			
SSS	7					
PURGE			8			
SSS	7					
COND			3			
SSS	7					
PLTSET			8			
SSS	7					
SAVE			8			
SSS	7					
PRTMSG			8			
SSS	7					
PARAM			8			
SSS	7					
PARAM			8			
SSS	7					
COND			8			
SSS	7					
PLOT			8			
SSS	7					
SAVE			8			
SSS	7					
PRTMSG			3			
SSS	7					
LABEL			8			
SSS	7					
CHKPNT			8			
SSS	67					
GP3	12		3			1
CHKPNT	12		3			1
SSS	6					
TA1	1234567		3			39
SAVE	1234567		3			89
PURGE	1234567		3			99
CHKPNT	1234567		3			89
SSS	6					
COND	123 5678		345		4	
PARAM	123 5 8					
PARAM	123 5 7 8		4		4	
PARAM	123 8					
PARAM	123 6 8					
EMG	123 5679					
SAVE	123 5679					
CHKPNT	123 5678					
SSS	6					
COND	123 6 8					
EMA	123 5 8					
CHKPNT	123 6 8					

DIRECT TRANSIENT RESPONSE

DMAP Inst.	1	10	20	<u>Bit Position</u> 30	40	50	60
\$\$\$		6					
LABEL	123	6 8					
COND	123	5 7 8	4	4			
EMA	123	5 7 8	4	4			
CHKPNT	123	5 7 8	4	4			
\$\$\$		6					
LABEL	123	5 7 8	4	4			
COND	123	8					89
EMA	123	8					89
CHKPNT	123	8					89
\$\$\$		6					
LABEL	123	8					89
COND	123	8					
EMA	123	6 8					
CHKPNT	123	6 8					
\$\$\$		6					
LABEL	123	6 8					
PURGE	123	5 7 8	4	4			
PURGE	123	8					89
CHKPNT	123	5 7 8	4	4			89
\$\$\$		6					
COND	123	5 7 8	45	4			
\$\$\$		8					
COND	123	5 7 8	45	4			
\$\$\$		8					
GPWG	123	5 7 8	45	4			
\$\$\$		8					
OFF	123	5 7 8	45	4			
\$\$\$		8					
LABEL	123	5 7	45	4			
EQUIV	1234	6 8					
CHKPNT	1234	6 8					
\$\$\$		6					
COND	1234	6 8					
SMA3	1234	6 8					
CHKPNT	1234	6 8					
\$\$\$		6					
LABEL	1234	6 8					
PARAM	1	901					
GP4	1	901					
SAVE	1	901					
PURGE	1	901					
CHKPNT	12345678901	4	4				
\$\$\$		6					
COND	123	6 8 0					
COND	123	6 8 90					
GPSP	123	6 8 90					
SAVE	123	6 8 90					
COND	123	6 8 90					
OFF	123	6 8 90					
LABEL	123	6 8 90					
EQUIV	123456789	4	4				
CHKPNT	123456789	4	4				
\$\$\$		6					
COND	123456789	4	4				
MCE1	1	9					
CHKPNT	1	9					
\$\$\$		6					
MCE2	123456789	4	4				89

RIGID FORMAT RESTART TABLES

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
CHKPNT	123456789	4	4				39
\$SS	6						
LABEL	123456789	4	4				39
EQUIV	1234567890	4	4				39
CHKPNT	1234567890	4	4				39
\$SS	6						
COND	1234567890	4	4				39
SCE1	1234567890	4	4				39
CHKPNT	1234567890	4	4				39
\$SS	6						
LABEL	1234567890	4	4				39
EQUIV	1234 6 8901						
EQUIV	12345 78901	4	4				
EQUIV	1234 8901						39
EQUIV	1234 6 8901						
CHKPNT	12345678901	4	4				39
\$SS	6						
COND	12345678901	4	4				39
SMP1	1234 6 8901						
CHKPNT	1234 6 8901						
\$SS	6						
COND	12345678901	4	4				
SMP2	12345678901	4	4				
CHKPNT	12345678901	4	4				
\$SS	6						
LABEL	12345678901	4	4				
COND	1234 6 8901						39
SMP2	1234 6 8901						39
CHKPNT	1234 6 8901						39
\$SS	6						
LABEL	1234 6 8901						39
COND	1234 6 8901						
SMP2	1234 6 8901						
CHKPNT	1234 6 8901						
\$SS	6						
LABEL	12345678901	4	4				39
DPO	1 901						7 1
SAVE	1 901						7 1
PURGE	1						7 1
EQUIV	12345678901	4	4			2	67390
CHKPNT	1 901					2	7 01
\$SS	6						
BMG	1					2	
SAVE	1					2	
PARAM	1					2	7 0
PURGE	1					2	
COND	1					2	
MTPXIN	1					2	
SAVE	1					2	
LABEL	1					2	
CHKPNT	1					2	
\$SS	6						
MTPXIN	1					2	7 0
SAVE	1					2	7 0
PARAM	1					2	7 0
PAFAM	1					2	7 0
EQUIV	1					2	7 0
ADDS	1					2	7 0
COND	1					2	7 0

DIRECT TRANSIENT RESPONSE

DMAP	Bit Position									
Inst.	1	10	20	30	40	50	60			
TRNSP	1					2	7 0			
ADD	1					2	7 0			
LABEL	1					2	7 0			
PARAM	1					2	7 0			
PARAM	1					2	7 0			
PARAM	1					2	7 0			
PURGE	12345678901	4	4			2	7890			
EQUIV	12345678901	4	4			2	7890			
CHKPNT	12345678901	4	4			2	7890			
\$\$\$	6									
COND	12345678901	4	4			2	7890			
GKAD	12345678901	4	4			2	67890			
LABEL	12345678901	4	4			2	67890			
EQUIV	12345678901	4	4			2	67890			
CHKPNT	12345678901	4	4			2	67890			
\$\$\$	6									
COND	12345678901	4	4			2	67890			
PAPAM		4	234							
PARAM	12345678901234	6	901234	7		2	6789012			
PARAM			901	7						
JUMP		4	234							
\$\$\$	1 3									
LABEL	1234567890123456	89012345				2	6789012			
\$\$\$	1 3									
PURGE		4	901234	7						
CASE	12345678901234	6	9012345	7		2	6789012			
SAVE	12345678901234	6	9012345	7		2	6789012			
CHKPNT	12345678901234	6	9012345	7		2	6789012			
\$\$\$	6									
PARAM	12345678901	4	4			2	67890			
\$\$\$	4									
TRLG	12345678901	4	234			2	6789012			
SAVE	12345678901	4	234			2	6789012			
CHKPNT	12345678901	4	234			2	6789012			
\$\$\$	6									
EQUIV	12345678901	4	234			2	6789012			
CHKPNT	12345678901	4	234			2	6789012			
\$\$\$	6									
TRD	12345678901	4	234			2	6789012			
SAVE	12345678901	4	234			2	6789012			
CHKPNT	12345678901	4	234			2	6789012			
\$\$\$	6									
VDR			901	7						
SAVE			901	7						
CHKPNT			1	7						
\$\$\$	6									
COND			1	7						
SDR3			1	7						
OFF			1							
SAVE			1							
CHKPNT			1	7						
\$\$\$	6									
XYTRAN				7						
\$\$\$	7									
SAVE				7						
\$\$\$	7									
XYPLOT				7						
\$\$\$	7									
LABEL			1							
PARAM	12345678901	4	234			2	6789012			

RIGID FORMAT RESTART TABLES

DMAP Inst.	1		10	20	Bit Position 30	40	50	60
COND	12345678901	4		234			2	6789012
EQUIV	12345678901	4		234			2	6789012
COND	12345678901	4		234			2	6789012
SDR1	12345678901	4		234			2	6789012
LABEL	12345678901	4		234			2	6789012
CHKPNT	12345678901	4		234			2	6789012
\$\$\$	6							
SDR2				890				
SDR3				890				
CHKPNT				890				
\$\$\$	6							
OFF				9				
SAVE				9				
COND				8				
\$\$\$	7							
PLOT				8				
\$\$\$	7							
SAVE				8				
\$\$\$	7							
PRTMSG				8				
\$\$\$	7							
LABEL				8				
\$\$\$	7							
XYTRAN				0				
\$\$\$	7							
SAVE				0				
\$\$\$	7							
XYPLOT				0				
\$\$\$	7							
LABEL				0				
COND				23				
\$\$\$	1 3							
REPT				23				
\$\$\$	1 3							
JUMP				23				
\$\$\$	1 3							
JUMP	1234567890123456	8901234					2	6789012
LABEL				23				
\$\$\$	1 3							
PRTPARM				23				
\$\$\$	1 3							
LABEL	1234567890123456	8901234					2	6789012
PRTPARM	1234567890123456	8901234					2	6789012
LABEL	1234567890123456	8901234					2	6789012
\$\$\$	8							
PRTPARM	1234567890123456	8901234					2	6789012
\$\$\$	8							
LABEL	1234567890123456	8901234					2	6789012
END	1234567890123456	8901234					2	6789012

DIRECT TRANSIENT RESPONSE

7.10-4 Rigid Format Change Restart Table

DMAP Inst.	63	Bit Position 70	80	DMAP Inst.	63	Bit Position 70	80
BEGIN		34567890	234567	345	GPWG		
FILE		34567890	234567	345	OFF		
GP1				LABEL			
SAVE				EQUIV			
PURGE				CHKPNT			
CHKPNT				COND			
COND				SMA3			
GP2				CHKPNT			
CHKPNT				LABEL			
PARAML				PARAM			
PURGE				GP4			
COND				SAVE			
PLTSET				PURGE			
SAVE				CHKPNT			
PRTMSG				COND			
PAFAM				COND			
PARAM				GPSP			
COND				SAVE			
PLOT				COND			
SAVE				OFF			
PRTMSG				LABEL			
LABEL				EQUIV			
CHKPNT				CHKPNT			
GP3				COND			
CHKPNT				MCE1			
TA1				CHKPNT			
SAVE				MCE2			
PURGE				CHKPNT			
CHKPNT				LABEL			
COND				EQUIV			
PARAM				CHKPNT			
PARAM	3	678		COND			
PARAM	3	678		SCE1			
PAPAM				CHKPNT			
EMG	3	678		LABEL			
SAVE	3	678		EQUIV			
CHKPNT	3	678		EQUIV			
COND				EQUIV			
EMA				EQUIV			
CHKPNT				CHKPNT			
LABEL				COND			
COND	3	678		SMP1			
EMA	3	678		CHKPNT			
CHKPNT	3	678		COND			
LABEL	3	678		SMP2			
COND	3	678		CHKPNT			
EMA	3	678		LABEL			
CHKPNT	3	678		COND			
LABEL	3	678		SMP2			
COND				CHKPNT			
EMA				LABEL			
CHKPNT				COND			
LABEL				SMP2			
PURGE	3	678		CHKPNT			
PURGE	3	678		LABEL			
CHKPNT	3	678		DPD			
COND				SAVE			
COND				PURGE			

RIGID FORMAT RESTART TABLES

DMAP Inst.	63	<u>Bit Position</u> 70	80
EQUIV			
CHKPNT			
BMG			
SAVE			
PARAM			
PURGE			
COND			
MTRXIN			
SAVE			
LABEL			
CHKPNT			
MTRXIN			
SAVE			
PARAM			
PARAM			
EQUIV			
ACDS			
COND			
TRNSP			
ADD			
LABEL			
PARAM			
PARAM			
PARAM			
PURGE			
EQUIV			
CHKPNT			
COND		90	
GKAD		90	
LABEL		90	
EQUIV			
CHKPNT			
COND	34567890	234567	345
PARAM			
PARAM	34567890	234567	345
PARAM			
JUMP	34567890	234567	345
LABEL	34567890	234567	345
PURGE			
CASE	34567890	234567	345
SAVE	34567890	234567	345
CHKPNT	34567890	234567	345
PARAM			
TRLG			
SAVE			
CHKPNT			
EQUIV			
CHKPNT			
TRD			
SAVE			
CHKPNT			
VDR			
SAVE			
CHKPNT			
COND			
SDR3			
OFF			
SAVE			
CHKPNT			

DMAP Inst.	63	<u>Bit Position</u> 70	80
XYTRAN			
SAVE			
XYPLOT			
LABEL			
PARAM	34567890	234567	345
COND	34567890	234567	345
EQUIV	34567890	234567	345
COND	34567890	234567	345
SDR1	34567890	234567	345
LABEL	34567890	234567	345
CHKPNT	34567890	234567	345
SDR2			
SDR3			
CHKPNT			
OFF			
SAVE			
COND			
PLOT			
SAVE			
PRTMSG			
LABEL			
XYTRAN			
SAVE			
XYPLOT			
LABEL			
COND	34567890	234567	345
REPT	34567890	234567	345
JUMP	34567890	234567	345
JUMP	34567890	234567	345
LABEL	34567890	234567	345
PRTPARM	34567890	234567	345
LABEL	34567890	234567	345
PRTPARM	34567890	234567	345
LABEL	34567890	234567	345
PRTPARM	34567890	234567	345
LABEL	34567890	234567	345
END	34567890	234567	345

DIRECT TRANSIENT RESPONSE

7.10.5 File Name Restart Table

DMAP Inst.	94	Bit Position		120	DMAP Inst.	94	100	110	120
BEGIN					GPWG				
FILE					OFF				
GP1	4				LABEL	8			
SAVE	4				EQUIV		0		
PURGE					CHKPNT		0		
CHKPNT					COND		0		
COND					SMA3		0		
GP2	5				CHKPNT		0		
CHKPNT	5				LABEL		0		
PARAML				0	PARAM		1		
PURGE				0	GP4		1		
COND				0	SAVE		1		
PLTSET				0	PURGE		1 3 56	01 4	
SAVE				0	CHKPNT		1 3 56	01 4	
PRTMSG				0	COND		2		
PARAM				0	COND		2		
PARAM				0	GPSP		2		
COND					SAVE		2		
PLOT					COND		2		
SAVE					OFF		2		
PRTMSG					LABEL		2		
LABEL					EQUIV			4	
CHKPNT				0	CHKPNT			4	
GP3	6				COND		34		
CHKPNT	6				MCE1		3		
TA1	7				CHKPNT		3		
SAVE	7				MCE2		4		
PURGE	7 8 9	2 4 5 6		123	CHKPNT		4		
CHKPNT	7				LABEL		34		
COND	8				EQUIV		5		
PARAM	8				CHKPNT		5		
PARAM	9				COND		5		
PARAM				5	SCE1		5		
PARAM				6	CHKPNT		5		
EMG				4	LABEL		5		
SAVE				4	EQUIV			6	
CHKPNT				4	EQUIV				1
COND	8				EQUIV				2
EMA	8				EQUIV				3
CHKPNT	8				CHKPNT		6		123
LABEL	8				COND		6		123
COND	9				SMP1		6		
EMA	9				CHKPNT		6		
CHKPNT	9				COND				1
LABEL	9				SMP2				1
COND				5	CHKPNT				1
EMA				5	LABEL				1
CHKPNT				5	COND				2
LABEL				5	SMP2				2
COND				6	CHKPNT				2
EMA				6	LABEL				2
CHKPNT				6	COND				3
LABEL				6	SMP2				3
PURGE	9				CHKPNT				3
PURGE				5	LABEL		6		123
CHKPNT	9				DPD		7		
COND					SAVE		7		
COND					PURGE		7		

RIGID FORMAT RESTART TABLES

Bit Position				Bit Position			
94	100	110	120	94	100	110	120
DMAP				DMAP			
Inst.				Inst.			
EQUIV		0		XYTRAN			
CHKPNT	7		8	SAVE			
BMG			8	XYPLOT			
SAVE			9	LABEL		4	
PARAM	9		8	PARAM		4	
PURGE			9	COND		4	
COND			9	EQUIV		4	
MTRXIN			9	COND		4	
SAVE			9	SDR1		4	
LABEL			9	LABEL		4	
CHKPNT			9	CHKPNT		4	
MTRXIN	9			SDR2		5	
SAVE	9			SDR3		6	
PARAM	9			CHKPNT		6	
PAPAM	9			OFF			
EQUIV	9			SAVE			
ADDS	9			COND			
COND	9			PLOT			
TRNSP	9			SAVE			
ADD	9			PRTMSG			
LABEL	9			LABEL			
PARAM	9			XYTRAN			
PARAM	9			SAVE			
PARAM	9			XYPLOT			
PURGE	0			LABEL		4	
EQUIV	0			COND			
CHKPNT	0			REPT			
COND	0			JUMP			
GKAD	0			JUMP			
LABEL	0			LABEL			
EQUIV	0			PRTPARM			
CHKPNT	0			LABEL			
COND	1			PRTPARM			
PARAM				LABEL			
PARAM	8			PRTPARM			
PAPAM				LABEL			
JUMP	8			END			
LABEL	8						
PURGE							
CASE	8						
SAVE	8						
CHKPNT	8						
PAPAM							
TRLG		1					
SAVE		1					
CHKPNT		1					
EQUIV		1					
CHKPNT		1					
TRD				7			
SAVE				7			
CHKPNT				7			
VDR		2					
SAVE		2					
CHKPNT		2					
COND		3					
SDR3		3					
OFF							
SAVE							
CHKPNT		3					

MODAL COMPLEX EIGENVALUE ANALYSIS

7.11 RESTART TABLES FOR MODAL COMPLES EIGENVALUE ANALYSIS

7.11.1 Bit Positions for Card Name Restart Table

Card Name	Bit Pos.	Card Name	Bit Pos.	Card Name	Bit Pos.
AXIC	1	CQDPLT	2	PTRPLT1	3
AXIF	1	CQUAD1	2	PTRSHL	3
CELAS1	1	CQUAD2	2	PTUBE	3
CELAS2	1	CQUADTS	2	PTWIST	3
CELAS3	1	CROD	2	GENEL	4
CELAS4	1	CSHEAR	2	CONM1	5
CMASS1	1	CTETRA	2	CONM2	5
CMASS2	1	CTORDRG	2	PELAS	6
CMASS3	1	CTRAPAX	2	PMASS	7
CMASS4	1	CTRAPRG	2	MAT1	8
CORD1C	1	CTRBSC	2	MAT2	8
CORD1R	1	CTRIA1	2	MAT3	8
COPD1S	1	CTRIA2	2	MATT1	8
CORD2C	1	CTRIAAX	2	MATT2	8
CORD2R	1	CTRIARG	2	MATT3	8
CORD2S	1	CTRIATS	2	TABLEM1	8
GRDSET	1	CTRM6	2	TABLEM2	8
GRID	1	CTRMEM	2	TABLEM3	8
GRIDB	1	CTRPLT	2	TABLEM4	8
POINTAX	1	CTRPLT1	2	TEMPMT\$	8
RINGAX	1	CTRSHL	2	TEMPMX\$	8
RINGFL	1	CTUBE	2	AXISYM\$	9
SECTAX	1	CTWIST	2	CRIGD1	9
SEQGP	1	CWEDGE	2	CRIGD2	9
SPOINT	1	PBAR	3	CRIGD3	9
ADUM1	2	PCONEAX	3	CRIGDR	9
ADUM2	2	PDUM1	3	MPC	9
ADUM3	2	PDUM2	3	MPCADD	9
ADUM4	2	PDUM3	3	MPCAX	9
ADUM5	2	PDUM4	3	MPC\$	9
ADUM6	2	PDUM5	3	SPC	10
ADUM7	2	PDUM6	3	SPC1	10
ADUM8	2	PDUM7	3	SPCADD	10
ADUM9	2	PDUM8	3	SPCAX	10
BAROR	2	PDUM9	3	SPC\$	10
CBAR	2	PIHEX	3	ASET	11
CCONEAX	2	PQDMEM	3	ASET1	11
CDUM1	2	PQDMEM1	3	OMIT	11
CDUM2	2	PQDMEM2	3	OMIT1	11
CDUM3	2	PQDMEM3	3	OMITAX	11
CDUM4	2	PQDPLT	3	SUPAX	12
CDUM5	2	PQUAD1	3	SUPORT	12
CDUM6	2	PQUAD2	3	TEMP	13
CDUM7	2	PQUADTS	3	TEMPAX	13
CDUM8	2	PROD	3	TEMPO	13
CDUM9	2	PSHEAR	3	TEMPP1	13
CHEXA1	2	PTORDRG	3	TEMPP2	13
CHEXA2	2	PTRAPAX	3	TEMPP3	13
CIHEX1	2	PTRBSC	3	TEMPRB	13
CIHEX2	2	PTRIA1	3	WTMASS	14
CIHEX3	2	PTRIA2	3	GRDPNT	15
CONROD	2	PTRIAAX	3	PLOTTEL	16
CQDMEM	2	PTRIATS	3	PLOTS\$	18
CQDMEM1	2	PTRIM6	3	POUT\$	19
CQDMEM2	2	PTRMEM	3	AOUT\$	21
CQDMEM3	2	PTRPLT	3	LOOP\$	22

RIGID FORMAT RESTART TABLES

Card Name Bit Pos.

LOOP1\$	23
COUPMASS	24
CPBAR	24
CPOPLT	24
CPQUAD1	24
CPQUAD2	24
CPROD	24
CPTRBSC	24
CPTRIA1	24
CPTRIA2	24
CPTRPLT	24
CPTUBE	24
NLOOP\$	25
EPOINT	56
SEQEP	56
TF	56
DMIG	57
DMIAx	57
B2PP\$	57
K2PP\$	57
M2PP\$	57
TF\$	57
EIGR	58
METHOD\$	59
EIGC	60
EIGP	60
CMETHOD\$	61
LFREQ	62
LMODES	62
HFREQ	62
SDAMP\$	62
TABDMP1	62

MODAL COMPLEX EIGENVALUE ANALYSIS

7.11.2 Bit Positions for File Name Restart Table

<u>File Name</u>	<u>Bit Pos.</u>	<u>File Name</u>	<u>Bit Pos.</u>
BGPD	94	USETD	111
CSTM	94	LAMA	112
EQEXIN	94	MI	112
GPD	94	PHIA	112
GPL	94	OEIGS	112
SIL	94	CASEXX	113
ECT	95	B2PP	114
GPTT	96	K2PP	114
EST	97	M2PP	114
GEI	97	GMD	115
GPECT	97	GOD	115
GPST	98	B2DD	115
KGGX	98	K2DD	115
MGG	99	M2DD	115
KGG	100	BHH	116
ASET	101	KHH	116
RG	101	MHH	116
USET	101	PHIDH	116
OGPST	102	CLAMA	117
GM	103	OCEIGS	117
KNN	104	PHIH	117
MNN	104	OPHIH	118
KFF	105	CPHID	119
KFS	105	CPHIP	120
MFF	105	QPC	120
GO	106	QCPHIP	121
KAA	106	OEFC1	121
KLL	107	OESC1	121
KLR	107	OQPC1	121
KRR	107	ELSETS	122
MLL	107	GPSETS	122
MLR	107	PLTPAR	122
MRR	107	PLTSETX	122
LLL	108	MAA	123
DM	109	KDICT	124
MR	110	KELM	124
EED	111	MDICT	124
EQDYN	111	MELM	124
GPLD	111		
SILD	111		
TFPOOL	111		

RIGID FORMAT RESTART TABLES

7.11.3 Card Name Restart Table

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
BEGIN	1234567890123456	89	1234				6789012
FILE	1234567890123456	89	1234				6789012
GP1	1						
SAVE	1						
CHKPNT	1						
\$SS		6					
GP2	12 45		6				
CHKPNT	12 45		6				
\$SS		6					
PARAML			8				
\$SS		7					
PURGE			8				
\$SS		7					
COND			8				
\$SS		7					
PLTSET			8				
\$SS		7					
SAVE			8				
\$SS		7					
PRTMSG			8				
\$SS		7					
PARAM			8				
\$SS		7					
PARAM			8				
\$SS		7					
COND			8				
\$SS		7					
PLOT			8				
\$SS		7					
SAVE			8				
\$SS		7					
PRTMSG			8				
\$SS		7					
LABEL			8				
\$SS		7					
CHKPNT			8				
\$SS		67					
GP3	1		3				
CHKPNT	1		3				
\$SS		6					
TA1	1234567		3				
SAVE	1234567		3				
COND	12345678		34		4		
PUPGE	1234567		3				
CHKPNT	1234567		3				
\$SS		6					
PARAM	123 6 8						
PAPAM	123 5 7 8		4		4		
EMG	123 5678		4		4		
SAVE	123 5678		4		4		
CHKPNT	123 5678		4		4		
\$SS		6					
COND	123 6 8						
EMA	123 6 8						
CHKPNT	123 6 8						
\$SS		6					
LABEL	123 6 8						
COND	123 5 7 8		4		4		
EMA	123 5 7 8		4		4		

MODAL COMPLEX EIGENVALUE ANALYSIS

DMAP	Bit Position						
Inst.	1	10	20	30	40	50	60
CHKPNT	123 5 7 8	4		4			
\$\$\$	6						
COND	123 5 7 8	45		4			
\$\$\$	8						
GPWG	123 5 7 8	45		4			
\$\$\$	8						
OFF	123 5 7 8	45		4			
\$\$\$	8						
LABEL	123 5 7 8	45		4			
\$\$\$	8						
EQUIV	1234 6 8						
CHKPNT	1234 6 8						
\$\$\$	6						
COND	1234 6 8						
SMA3	1234 6 8						
CHKPNT	1234 6 8						
\$\$\$	6						
LABEL	1234 6 8						
PARAM	1	9012					
GP4	1	9012					
SAVE	1	9012					
PARAM	1	9012					
PURGE	1	9012					
CHKPNT	123456789012 4			4			
\$\$\$	6						
COND	123 6 8 90						
GPSP	123 6 8 90						
SAVE	123 6 8 90						
COND	123 6 8 90						
OFF	123 6 8 90						
LABEL	123 6 8 90						
EQUIV	123456789	4		4			
CHKPNT	123456789	4		4			
\$\$\$	6						
COND	123456789	4		4			
MCE1	1 9						
CHKPNT	1 9						
\$\$\$	6						
MCE2	123456789	4		4			
CHKPNT	123456789	4		4			
\$\$\$	6						
LABEL	123456789	4		4			
EQUIV	1234567890	4		4			
CHKPNT	1234567890	4		4			
\$\$\$	6						
COND	1234567890	4		4			
SCE1	1234567890	4		4			
CHKPNT	1234567890	4		4			
\$\$\$	6						
LABEL	1234567890	4		4			
EQUIV	1234 6 8 901						
EQUIV	12345 7 8 901	4		4			
CHKPNT	12345678901	4		4			
\$\$\$	6						
COND	12345678901	4		4			
SMP1	1234 6 8 901						
CHKPNT	1234 6 8 901						
\$\$\$	6						
SMP2	12345678901 4			4			

RIGID FORMAT RESTART TABLES

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
CHKPNT	12345678901	4		4			
\$\$\$	6						
LABEL	12345678901	4		4			
COND	123456789012	4		4			
RBMG1	123456789012	4		4			
CHKPNT	123456789012	4		4			
\$\$\$	6						
RBMG2	1234 6 89012						
CHKPNT	1234 6 89012						
\$\$\$	6						
RBMG3	1234 6 89012						
CHKPNT	1234 6 89012						
\$\$\$	6						
RBMG4	123456789012	4		4			
CHKPNT	123456789012	4		4			
\$\$\$	6						
LABEL	123456789012	4		4			
DPO	1 9012						6 8 0
SAVE	1 9012						6 8 0
COND	1 9012						6 8 0
EQUIV	123456789012	4	234				6789
CHKPNT	1 9012						6 3
\$\$\$	6						
PARAM	123456789012	4		4			
READ	123456789012	4		4			89
SAVE	123456789012	4		4			89
CHKPNT	123456789012	4		4			89
\$\$\$	6						
PARAM			9				
OFF	123456789012	4		4			89
SAVE	123456789012	4		4			89
COND	123456789012	4		4			89
OFF	123456789012	4		4			89
SAVE	123456789012	4		4			89
PARAM			23				
PARAM	12345678901234	6	23				6789012
JUMP			23				
\$\$\$	1 3						
LABEL	1234567890123456	89	123				6789012
\$\$\$	1 3						
PURGE			23				
CASE	12345678901234	6	9	123	5		6789012
SAVE	12345678901234	6	9	123	5		6739012
CHKPNT	12345678901234	6	9	123	5		6789012
\$\$\$	6						
MTRXIN	1		23				67
SAVE	1		23				67
PURGE	12345678901		23				67
EQUIV	12345678901		23				67
CHKPNT	12345678901		23				67
\$\$\$	6						
GKAD	12345678901	4	234				67
CHKPNT	12345678901	4	234				67
\$\$\$	6						
GKAM	123456789012	4	234				6789 2
SAVE	123456789012	4	234				6789 2
CHKPNT	123456789012	4	234				6789 2
\$\$\$	6						
CEAD	123456789012	4	234				6739012

MODAL COMPLEX EIGENVALUE ANALYSIS

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
SAVE	123456789012 4		234				6789012
CHKPNT	123456789012 4		234				6789012
\$SS	6						
OFF	123456789012 4		234				6789012
SAVE	123456789012 4		234				6789012
COND			9 1				
VDR			9 1				
SAVE			9 1				
COND			1				
OFF			1				
SAVE			1				
LABEL			1				
COND	123456789012 4		234				6789012
DDR1	123456789012 4		234				6789012
CHKPNT	123456789012 4		234				6789012
\$SS	6						
EQUIV	123456789012 4		234				6789012
COND	123456789012 4		234				6789012
SDR1	123456789012 4		234				6789012
LABEL	123456789012 4		234				6789012
CHKPNT	123456789012 4		234				6789012
\$SS	6						
SDR2			9				
OFF			9				
SAVE			9				
LABEL	123456789012						6789012
COND			23				
\$SS	1 3						
REPT			23				
\$SS	1 3						
JUMP			23				
\$SS	1 3						
JUMP	1234567890123456 89	1234					6789012
LABEL			23				
\$SS	1 3						
PRTPARM			23				
\$SS	1 3						
LABEL	1234567890123456 89	1234					6789012
PRTPARM	1234567890123456 89	1234					6789012
LABEL	1234567890123456 89	1234					6789012
PRTPARM	1234567890123456 89	1234					6789012
LABEL	1234567890123456 89	1234					6789012
PRTPARM	1234567890123456 89	1234					6789012
LABEL	1234567890123456 89	1234					6789012
END	1234567890123456 89	1234					6789012

RIGID FORMAT RESTART TABLES

7.11.4 Rigid Format Change Restart Table

DMAPI	Bit Position			DMAPI	Bit Position		
Inst.	63	70	80	Inst.	63	70	80
BEGIN		345678901	34567	345	SAVE		
FILE		345678901	34567	345	COND		
GP1				OFF			
SAVE				LABEL			
CHKPNT				EQUIV			
GP2				CHKPNT			
CHKPNT				COND			
PARAML				MCE1			
PURGE				CHKPNT			
COND				MCE2			
PLTSET				CHKPNT			
SAVE				LABEL			
PRTMSG				EQUIV			
PARAM				CHKPNT			
PARAM				COND			
COND				SCE1			
PLOT				CHKPNT			
SAVE				LABEL			
PRTMSG				EQUIV			
LABEL				EQUIV			
CHKPNT				CHKPNT			
GP3				COND			
CHKPNT				SMP1			
TA1				CHKPNT			
SAVE				SMP2			
COND				CHKPNT			
PURGE				LABEL			
CHKPNT				COND			
PARAM				RBMG1			
PARAM	3	678		CHKPNT			
EMG	3	678		RBMG2			
SAVE	3	678		CHKPNT			
CHKPNT	3	678		RBMG3			
COND				CHKPNT			
EMA				RBMG4			
CHKPNT				CHKPNT			
LABEL				LABEL			
COND	3	678		DPD			
EMA	3	678		SAVE			
CHKPNT	3	678		COND	345678901	34567	345
COND				EQUIV			
GPWG				CHKPNT			
OFF				PARAM			
LABEL				READ			
EQUIV				SAVE			
CHKPNT				CHKPNT			
COND				PARAM			
SMA3				OFF			
CHKPNT				SAVE			
LABEL				COND	345678901	34567	345
PARAM				OFF			
GP4				SAVE			
SAVE				PARAM			
PAFAM				PARAM	345678901	34567	345
PURGE				JUMP	345678901	34567	345
CHKPNT				LABEL	345678901	34567	345
COND				PURGE			
GPSP				CASE	345678901	34567	345

MODAL COMPLEX EIGENVALUE ANALYSIS

DMAP	Bit Position		
Inst.	63	70	80
SAVE	345678901	34567	345
CHKPNT	345678901	34567	345
MTRXIN			
SAVE			
PURGE			
EQUIV			
CHKPNT			
GKAD			
CHKPNT			
GKAM			
SAVE			
CHKPNT			
CEAD			
SAVE			
CHKPNT			
OFF			
SAVE			
COND			
VDR			
SAVE			
COND			
OFF			
SAVE			
LABEL			
COND			
DDR1			
CHKPNT			
EQUIV	345678901	34567	345
COND	345678901	34567	345
SDR1	345678901	34567	345
LABEL	345678901	34567	345
CHKPNT	345678901	34567	345
SDR2			
OFF			
SAVE			
LABEL			
COND	345678901	34567	345
REPT	345678901	34567	345
JUMP	345678901	34567	345
JUMP	345678901	34567	345
LABEL	345678901	34567	345
PRTPARM	345678901	34567	345
LABEL	345678901	34567	345
PRTPARM	345678901	34567	345
LABEL	345678901	34567	345
PRTPARM	345678901	34567	345
LABEL	345678901	34567	345
PRTPARM	345678901	34567	345
LABEL	345678901	34567	345
END	345678901	34567	345

RIGID FORMAT RESTART TABLES

7.11.5 File Name Restart Table

DMAP		Bit Position			DMAP		Bit Position		
Inst.	94	100	110	120	Inst.	94	100	110	120
BEGIN					SAVE		2		
FILE					COND		2		
GP1	4				OFF		2		
SAVE	4				LABEL		2		
CHKPNT	4				EQUIV		4		
GP2	5				CHKPNT		4		
CHKPNT	5				COND		34		
PARAML				2	MCE1		3		
PURGE				2	CHKPNT		3		
COND				2	MCE2		4		
PLTSET				2	CHKPNT		4		
SAVE				2	LABEL		34		
PRTMSG				2	EQUIV		5		
PARAM				2	CHKPNT		5		
PARAM				2	COND		5		
COND					SCE1		5		
PLOT					CHKPNT		5		
SAVE					LABEL		5		
PRTMSG					EQUIV		6		3
LABEL				2	EQUIV		6		3
CHKPNT					CHKPNT		6		3
GP3	6				COND		6		3
CHKPNT	6				SMP1		6		
TA1	7				CHKPNT		6		
SAVE	7				SMP2				3
COND	7				CHKPNT				3
PURGE	7	2			LABEL		6		3
CHKPNT	7				COND		7890		
PAPAM	8				RBMG1		7		
PARAM	9				CHKPNT	4	7		
EMG					RBMG2	4	8		
SAVE					CHKPNT	4	8		
CHKPNT					RBMG3		9		
COND	8				CHKPNT		9		
EMA	8				RBMG4		0		
CHKPNT	8				CHKPNT		0		
LABEL	8				LABEL		7890		
COND	9				OPD		1		
EMA	9				SAVE		1		
CHKPNT	9				COND		1		
COND					EQUIV				5
GPWG					CHKPNT		1		
OFF					PARAM		2		
LABEL					READ		2		
EQUIV	0				SAVE		2		
CHKPNT	0				CHKPNT		2		
COND	0				PARAM				
SMA3	0				OFF		2		
CHKPNT	0				SAVE		2		
LABEL	0				COND				
PARAM	1				OFF		2		
GP4	1				SAVE		2		
SAVE	1				PARAM				
PAPAM	1				PARAM				3
PURGE	1	3	567 90	5	JUMP				3
CHKPNT	1	3	567 90	5	LABEL				3
COND	2				PURGE				
GPSP	2								

MODAL COMPLEX EIGENVALUE ANALYSIS

DMAP		Bit Position		
Inst.	94	100	110	120
CASE			3	
SAVE			3	
CHKPNT			3	
MTRXIN			4	
SAVE			4	
PURGE			4	
EQUIV			4	
CHKPNT			4	
GKAD			5	
CHKPNT			5	
GKAM			6	
SAVE			6	
CHKPNT			6	
CEAD			7	
SAVE			7	
CHKPNT			7	
OFF			7	
SAVE			7	
COND			8	
VDR			8	
SAVE			8	
COND			8	
OFF			8	
SAVE			8	
LABEL			8	
COND			9	
DDF1			9	
CHKPNT			9	
EQUIV			0	
COND			0	
SDR1			0	
LABEL			0	
CHKPNT			0	
SDR2			1	
OFF			1	
SAVE			1	
LABEL				
COND				
REPT				
JUMP				
JUMP				
LABEL				
PRTPARM				
LABEL				
PRTPARM				
LABEL				
PRTPARM				
LABEL				
PRTPARM				
LABEL				
END				

MODAL FREQUENCY AND RANDOM RESPONSE

7.12 RESTART TABLES FOR MODAL FREQUENCY AND RANDOM RESPONSE

7.12.1 Bit Positions for Card Name Restart Table

Card Name	Bit Pos.	Card Name	Bit Pos.	Card Name	Bit Pos.
AXIC	1	CQUAD1	2	PTWIST	3
AXIF	1	CQUAD2	2	GENEL	4
CELAS1	1	CQUADTS	2	CONM1	5
CELAS2	1	CROD	2	CONM2	5
CELAS3	1	CSHEAR	2	PELAS	6
CELAS4	1	CTETRA	2	PMASS	7
CMASS1	1	CTORDRG	2	MAT1	8
CMASS2	1	CTRAPAX	2	MAT2	8
CMASS3	1	CTRAPRG	2	MAT3	8
CMASS4	1	CTRBSC	2	MATT1	8
CORD1C	1	CTRIA1	2	MATT2	8
CORD1R	1	CTRIA2	2	MATT3	8
CORD1S	1	CTRIAAX	2	TABLEM1	8
CORD2C	1	CTRIARG	2	TABLEM2	8
CORD2R	1	CTRIATS	2	TABLEM3	8
CORD2S	1	CTRIM6	2	TABLEM4	8
GRDSET	1	CTRMEM	2	TEMPMT\$	8
GRID	1	CTRPLT	2	TEMPMX\$	8
GRIDB	1	CTRPLT1	2	AXISYM\$	9
POINTAX	1	CTRSHL	2	CRIGD1	9
RINGAX	1	CTUBE	2	CRIGD2	9
RINGFL	1	CTWIST	2	CRIGD3	9
SECTAX	1	CHEDGE	2	CRIGDR	9
SEQGP	1	PBAR	3	MPC	9
SPOINT	1	PCONEAX	3	MPCADD	9
ADUM1	2	PDUM1	3	MPCAX	9
ADUM2	2	PDUM2	3	MPC\$	9
ADUM3	2	PDUM3	3	SPC	10
ADUM4	2	PDUM4	3	SPC1	10
ADUM5	2	PDUM5	3	SPCADD	10
ADUM6	2	PDUM6	3	SPCAX	10
ADUM7	2	PDUM7	3	SPC\$	10
ADUM8	2	PDUM8	3	ASET	11
ADUM9	2	PDUM9	3	ASET1	11
BAROR	2	PIHEX	3	OMIT	11
CBAR	2	PQDMEM	3	OMIT1	11
CCONEAX	2	PQDMEM1	3	OMITAX	11
CDUM1	2	PQDMEM2	3	SUPAX	12
CDUM2	2	PQDMEM3	3	SUPORT	12
CDUM3	2	PQDPLT	3	TEMP	13
CDUM4	2	PQUAD1	3	TEMPAX	13
CDUM5	2	PQUAD2	3	TEMPD	13
CDUM6	2	PQUADTS	3	TEMPP1	13
CDUM7	2	PROD	3	TEMPP2	13
CDUM8	2	PSHEAR	3	TEMPP3	13
CDUM9	2	PTORDRG	3	TEMPRB	13
CHEXA1	2	PTRAPAX	3	WTMASS	14
CHEXA2	2	PTRBSC	3	GRDPNT	15
CIHEX1	2	PTRIA1	3	PLOTEL	16
CIHEX2	2	PTRIA2	3	PLOTS	18
CIHEX3	2	PTRIAAX	3	POUT\$	19
CONROD	2	PTRIATS	3	XYOUT\$	20
CQDMEM	2	PTRIM6	3	AOUT\$	21
CQDMEM1	2	PTRMEM	3	LOOP\$	22
CQDMEM2	2	PTRPLT	3	LOOP1\$	23
CQDMEM3	2	PTRPLT1	3	COUPMASS	24
CQDPLT	2	PTRSHL	3	CPBAR	24
		PTUBE	3	CPQDPLT	24

RIGID FORMAT RESTART TABLES

<u>Card Name</u>	<u>Bit Pos.</u>
CPQUAD1	24
CPQUAD2	24
CPROD	24
CPTRBSC	24
CPTRIA1	24
CPTRIA2	24
CPTRPLT	24
CPTUBE	24
NOLLOOP\$	25
RANDOM\$	26
AXYOUT\$	27
MODACC	53
TABRND1	54
TABRND2	54
TABRND3	54
TABRND4	54
RANDPS	55
RANDT1	55
RANDT2	55
EPOINT	56
SEQEP	56
TF	56
DMIAX	57
DMIG	57
B2PP\$	57
K2PP\$	57
M2PP\$	57
TF\$	57
DAREA	58
DELAY	58
DLOAD	58
DPHASE	58
FREQ	58
FREQ1	58
FREQ2	58
RLOAD1	58
RLOAD2	58
TABLED1	58
TABLED2	58
TABLED3	58
TABLED4	58
EIGR	59
METHOD\$	60
DECOMOPT	61
DLOAD\$	61
FREQ\$	61
HFREQ	62
LFREQ	62
LMODES	62
TABDMP1	62
SDAMP\$	62

MODAL FREQUENCY AND RANDOM RESPONSE

7.12.2 Bit Positions for File Name Restart Table

<u>File Name</u>	<u>Bit Pos.</u>	<u>File Name</u>	<u>Bit Pos.</u>
BGPD	94	BHH	116
CSTM	94	KHH	116
EQEXIN	94	MHH	116
GPD	94	PHIDH	116
GPL	94	PDF	117
SIL	94	PPF	117
ECT	95	PSF	117
GPTT	96	UHV	117
EST	97	OUHVC1	118
GEI	97	OUHVC2	119
GPECT	97	PAF	120
GPST	98	UDV2F	120
KGGX	98	UEVF	120
MGG	99	QPC	121
KGG	100	UPVC	121
ASET	101	OEFC1	122
RG	101	OESC1	122
USET	101	OPPC1	122
OGPST	102	OQPC1	122
GM	103	OUPVC1	122
KNN	104	OEFC2	123
MNN	104	OESC2	123
KFF	105	OPPC2	123
KFS	105	OQPC2	123
MFF	105	OUPVC2	123
GO	106	UDV1F	124
KAA	106	AUTO	125
KLL	107	PSDF	125
KLR	107	ELSETS	126
KRR	107	GPSETS	126
MLL	107	PLTPAR	126
MLR	107	PLTSETX	126
MRR	107	MAA	127
LLL	108	KDICT	128
DM	109	KELM	128
MR	110	MDICT	128
DLT	111	MELM	128
EED	111	PHIPH	129
EQDYN	111	QPH	129
FRL	111	IEF1	130
GPLD	111	IES1	130
PSDL	111	IPHIP1	130
SILD	111	IQP1	130
TFPOOL	111	OPPCA	131
USETD	111	IEF2	132
LAMA	112	IES2	132
MI	112	IPHIP2	132
OEIGS	112	OPPCB	132
PHIA	112	IQP2	132
CASEXX	113	ZEFC2	133
B2PP	114	ZESC2	133
K2PP	114	ZQPC2	133
M2PP	114	ZUPVC2	133
B2DD	115	ZEFC1	134
GMD	115	ZESC1	134
GOD	115	ZQPC1	134
K2DD	115	ZUPVC1	134
M2DD	115		
MDD	115		

RIGID FORMAT RESTART TABLES

7.12.3 Card Name Restart Table

DMAP	Bit Position																	
Inst.	1	10				20	30				40		50	60				
BEGIN	1234567890123456	8901234				8901234								3456789012				
FILE	1234567890123456	8901234				8901234								3456789012				
GP1	1																	
SAVE	1																	
CHKPNT	1																	
\$\$\$		6																
GP2	12 45					6												
CHKPNT	12 45					6												
\$\$\$		6																
PARAML						8												
\$\$\$		7																
PURGE						8												
\$\$\$		7																
COND						8												
\$\$\$		7																
PLTSET						8												
\$\$\$		7																
SAVE						8												
\$\$\$		7																
PRTMSG						8												
\$\$\$		7																
PARAM						8												
\$\$\$		7																
PARAM						8												
\$\$\$		7																
COND						8												
\$\$\$		7																
PLOT						8												
\$\$\$		7																
SAVE						8												
\$\$\$		7																
PRTMSG						8												
\$\$\$		7																
LABEL						8												
\$\$\$		7																
CHKPNT						8												
\$\$\$		67																
GP3	1					3												
CHKPNT	1					3												
\$\$\$		6																
TA1	1234567					3												
SAVE	1234567					3												
COND	12345678					34					4							
PURGE	1234567					3												
CHKPNT	1234567					3												
\$\$\$		6																
PARAM	123 6 8																	
PARAM	123 5 7 8					4					4							
EMG	123 5678					4					4							
SAVE	123 5678					4					4							
CHKPNT	123 5678					4					4							
\$\$\$		6																
COND	123 6 8																	
EMA	123 6 8																	
CHKPNT	123 6 8																	
\$\$\$		6																
LABEL	123 6 8																	
COND	123 5 7 8					4					4							
EMA	123 5 7 8					4					4							

MODAL FREQUENCY AND RANDOM RESPONSE

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
CHKPNT	123 5 7 8	4	4				
\$SS	6						
COND	123 5 7 8	4 5	4				
\$SS	8						
GPWG	123 5 7 8	4 5	4				
\$SS	8						
QFP	123 5 7 8	4 5	4				
\$SS	8						
LABEL	123 5 7 8	4 5	4				
\$SS	8						
EQUIV	1234 6 8						
CHKPNT	1234 6 8						
\$SS	6						
COND	1234 6 8						
SMA3	1234 6 8						
CHKPNT	1234 6 8						
\$SS	6						
LABEL	1234 6 8						
PARAM	1	9012					
GP4	1	9012					
SAVE	1	9012					
PARAM	1	9012					
PURGE	1	9012					
CHKPNT	123456789012	4	4				
\$SS	6						
COND	123 6 8 90						
GPSP	123 6 8 90						
SAVE	123 6 8 90						
COND	123 6 8 90						
QFP	123 6 8 90						
LABEL	123 6 8 90						
EQUIV	123456789	4	4				
CHKPNT	123456789	4	4				
\$SS	6						
COND	123456789	4	4				
MCE1	1 9						
CHKPNT	1 9						
\$SS	6						
MCE2	123456789	4	4				
CHKPNT	123456789	4	4				
\$SS	6						
LABEL	123456789	4	4				
EQUIV	1234567890	4	4				
CHKPNT	1234567890	4	4				
\$SS	6						
COND	1234567890	4	4				
SCE1	1234567890	4	4				
CHKPNT	1234567890	4	4				
\$SS	6						
LABEL	1234567890	4	4				
EQUIV	1234 6 8 901						
EQUIV	12345 7 8 901	4	4				
CHKPNT	12345678901	4	4				
\$SS	6						
COND	12345678901	4	4				
SMP1	1234 6 8 901						
CHKPNT	1234 6 8 901						
\$SS	6						
SMP2	12345678901	4	4				

RIGID FORMAT RESTART TABLES

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
CHKPNT	12345678901	4	4				
\$\$\$	6						
LABEL	12345678901	4	4				
EQUIV	123456789012	4	4				
CHKPNT	123456789012	4	4				
\$\$\$	6						
COND	123456789012	4	4				
RBMG1	123456789012	4	4				
CHKPNT	123456789012	4	4				
\$\$\$	6						
JUMP	1234 6 89012						
LABEL	1234 6 89012						
COND	1234 6 89012						
LABEL	1234 6 89012						
RBMG2	1234 6 89012						
CHKPNT	1234 6 89012						
\$\$\$	6						
COND	1234 6 89012						
RBMG3	1234 6 89012						
CHKPNT	1234 6 89012						
\$\$\$	6						
RBMG4	123456789012	4	4				
CHKPNT	123456789012	4	4				
\$\$\$	6						
LABEL	123456789012	4	4				
OPD	1 9012					56 89	
SAVE	1 9012					56 89	
COND	1 9012					56 89	
PURGE	1 9012					56 89	
EQUIV	123456789012	4	234			3 67 90	
CHKPNT	1 9012					56 89	
\$\$\$	6						
PARAM	123456789012	4	4				
READ	123456789012	4	4				90
SAVE	123456789012	4	4				90
CHKPNT	123456789012	4	4				90
\$\$\$	6						
PARAM			9 1				
OFF	123456789012	4	4				90
SAVE	123456789012	4	4				90
COND	123456789012	4	4				90
OFF	123456789012	4	4				90
SAVE	123456789012	4	4				90
PARAM			23				
PARAM	12345678901234	6	90123	7		3456789012	
JUMP			23				
\$\$\$	1 3						
LABEL	1234567890123456	890123				3456789012	
\$\$\$	1 3						
PURGE			90123	7			
CASE	12345678901234	6	90123	5 7		3456789012	
SAVE	12345678901234	6	90123	5 7		3456789012	
CHKPNT	12345678901234	6	90123	5 7		3456789012	
\$\$\$	6						
MTRXIN	1		23			67	
SAVE	1		23			67	
PURGE	12345678901		23			67	
PARAM	12345678901		23			67	
EQUIV	12345678901		23			67	

MODAL FREQUENCY AND RANDOM RESPONSE

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
CHKPNT	12345678901		23				67
\$\$\$	6						
GKAD	12345678901	4	234			3	67 0
CHKPNT	12345678901	4	234			3	67 0
\$\$\$	6						
GKAM	123456789012	4	234				67 90 2
SAVE	123456789012	4	234				67 90 2
CHKPNT	123456789012	4	234				67 90 2
\$\$\$	6						
COND	123456789012	4	234				6789012
COND	123456789012	4	234				6789012
FRPD	123456789012	4	234				6789012
SAVE	123456789012	4	234				6789012
EQUIV	123456789012	4	234				6789012
CHKPNT	123456789012	4	234				6789012
\$\$\$	6						
VDR			901	7			
SAVE			901	7			
COND			1	7			
COND			1	7			
CHKPNT			1	7			
\$\$\$	6						
SDR3			1	7			
OFF			1				
SAVE			1				
CHKPNT			1	7			
\$\$\$	6						
XYTRAN				7			
\$\$\$	7						
SAVE				7			
\$\$\$	7						
XYPLOT				7			
\$\$\$	7						
JUMP			1	7			
LABEL			1	7			
OFF			1				
SAVE			1				
LABEL			01	7			
COND	123456789012	4	234			3	6789012
PAPAM	123456789012	4	234			3	6789012
COND	123456789012	4	234			3	6789012
DDR1	123456789012	4	234			3	6789012
CHKPNT	123456789012	4	234			3	6789012
\$\$\$	6						
DDR2	123456789012	4	234			3	6789012
\$\$\$	23						
CHKPNT	123456789012	4	234			3	6789012
\$\$\$	23 6						
EQUIV	123456789012	4	234			3	6789012
\$\$\$	23						
CHKPNT	123456789012	4	234			3	6789012
\$\$\$	23 6						
EQUIV	123456789012	4	234			3	6789012
COND	123456789012	4	234			3	6789012
SDR1	123456789012	4	234			3	6789012
LABEL	123456789012	4	234			3	6789012
CHKPNT	123456789012	4	234			3	6789012
\$\$\$	6						
SDR2			90				

RIGID FORMAT RESTART TABLES

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
SAVE			90				
COND			90				
SDR3			90				
JUMP			90				
LABEL	123456789012 4		234			3 6789012	
SDR1	123456789012 4		234			3 6789012	
SDR2			90				
SAVE			90				
SDR2			90				
EQUIV			90				
COND			90				
SDR3			90				
EQUIV			90				
DDRMM			90				
EQUIV			90				
JUMP			90				
LABEL			90				
DDRMM			90				
EQUIV			90				
JUMP			90				
LABEL			90				
CHKPNT			90				
\$\$\$	6						
OFF			9				
SAVE			9				
XYTRAN			0				
\$\$\$	7						
SAVE			0				
\$\$\$	7						
XYPLOT			0				
\$\$\$	7						
COND			0				
\$\$\$	7						
PLOT			0				
\$\$\$	7						
SAVE			0				
\$\$\$	7						
LABEL			0				
\$\$\$	7						
COND			0	6		45	
RANDOM				6		45	
SAVE				6		45	
CHKPNT				6		45	
\$\$\$	6						
COND				6		45	
XYTRAN			0				
\$\$\$	7						
SAVE			0				
\$\$\$	7						
XYPLOT			0				
\$\$\$	7						
JUMP			0				
LABEL			9				
OFF			9				
SAVE			9				
LABEL			0			45	
COND			23				

MODAL FREQUENCY AND RANDOM RESPONSE

DMAP	<u>Bit Position</u>						
Inst.	1	10	20	30	40	50	60
\$\$\$	1 3						
REPT			23				
\$\$\$	1 3						
JUMP			23				
\$\$\$	1 3						
JUMP	1234567890123456	8901234				3456789012	
LABEL			23				
\$\$\$	1 3						
PRTPARM			23				
\$\$\$	1 3						
LABEL	1234567890123456	8901234				3456789012	
PRTPARM	1234567890123456	8901234				3456789012	
LABEL	1234567890123456	8901234				3456789012	
PRTPARM	1234567890123456	8901234				3456789012	
LABEL	1234567890123456	8901234				3456789012	
PRTPARM	1234567890123456	8901234				3456789012	
LABEL	1234567890123456	8901234				3456789012	
PRTPARM	1234567890123456	8901234				3456789012	
LABEL	1234567890123456	8901234				3456789012	
PRTPARM	1234567890123456	8901234				3456789012	
LABEL	1234567890123456	8901234				3456789012	
PRTPARM	1234567890123456	8901234				3456789012	
LABEL	1234567890123456	8901234				3456789012	
END	1234567890123456	8901234				3456789012	

RIGID FORMAT RESTART TABLES

7.12.4 Rigid Format Change Restart Table

DMAP	Bit Position			DMAP	Bit Position		
Inst.	63	70	80	Inst.	63	70	80
BEGIN		3456789012	4567	345	SAVE		
FILE		3456789012	4567	345	COND		
GP1				OFF			
SAVE				LABEL			
CHKPNT				EQUIV			
GP2				CHKPNT			
CHKPNT				COND			
PARAML				MCE1			
PURGE				CHKPNT			
COND				MCE2			
PLTSET				CHKPNT			
SAVE				LABEL			
PRTMSG				EQUIV			
PARAM				CHKPNT			
PARAM				COND			
COND				SCE1			
PLOT				CHKPNT			
SAVE				LABEL			
PRTMSG				EQUIV			
LABEL				EQUIV			
CHKPNT				CHKPNT			
GP3				COND			
CHKPNT				SMP1			
TA1				CHKPNT			
SAVE				SMP2			
COND				CHKPNT			
PURGE				LABEL			
CHKPNT				EQUIV			
PARAM				CHKPNT			
PARAM	3	678		COND			
EMG	3	678		RBMG1			
SAVE	3	678		CHKPNT			
CHKPNT	3	678		JUMP			
COND				LABEL			
EMA				COND			
CHKPNT				LABEL			
LABEL				RBMG2			
COND	3	678		CHKPNT			
EMA	3	678		COND			
CHKPNT	3	678		RBMG3			
COND				CHKPNT			
GPWG				RBMG4			
OFF				CHKPNT			
LABEL				LABEL			
EQUIV				DPD			
CHKPNT				SAVE			
COND				COND	3456789012	4567	345
SMA3				PURGE			
CHKPNT				EQUIV			
LABEL				CHKPNT			
PARAM				PARAM			
GP4				READ			
SAVE				SAVE			
PARAM				CHKPNT			
PURGE				PARAM			
CHKPNT				OFF			
COND				SAVE			
GPSP				COND	3456789012	4567	345

MODAL FREQUENCY AND RANDOM RESPONSE

DMAP Inst.	63	Bit Position 70	80	DMAP Inst.	63	Bit Position 70	80
OFF				COND			
SAVE				SDR3			
PARAM				JUMP			
PARAM		3456789012	4567	LABEL			
JUMP		3456789012	4567	SDR1		3456789012	4567
LABEL		3456789012	4567	SDR2			
PURGE				SAVE			
CASE		3456789012	4567	SDR2			
SAVE		3456789012	4567	EQUIV			
CHKPNT		3456789012	4567	COND			
MTRXIN				SDR3			
SAVE				EQUIV			
PURGE				DDRMH			
PARAM				EQUIV			
EQUIV				JUMP			
CHKPNT				LABEL			
GKAO				DDRMH			
CHKPNT				EQUIV			
GKAM				JUMP			
SAVE				LABEL			
CHKPNT				CHKPNT			
COND		3456789012	4567	OFF			
COND		3456789012	4567	SAVE			
FRRD				XYTRAN			
SAVE				SAVE			
EQUIV				XYPLOT			
CHKPNT				COND			
VDR				PLOT			
SAVE				SAVE			
COND				LABEL			
COND				COND		3456789012	4567
CHKPNT				RANDOM		3456789012	4567
SDR3				SAVE		3456789012	4567
OFF				CHKPNT		3456789012	4567
SAVE				COND			
CHKPNT				XYTRAN			
XYTRAN				SAVE			
SAVE				XYPLOT			
XYPLOT				JUMP			
JUMP				LABEL			
LABEL				OFF			
OFF				SAVE			
SAVE				LABEL			
LABEL				COND		3456789012	4567
CONO				REPT		3456789012	4567
PAFAM				JUMP		3456789012	4567
COND				JUMP		3456789012	4567
DDR1				LABEL		3456789012	4567
CHKPNT				PRTPARM		3456789012	4567
DDR2				LABEL		3456789012	4567
CHKPNT				PRTPARM		3456789012	4567
EQUIV				LABEL		3456789012	4567
CHKPNT				PRTPARM		3456789012	4567
EQUIV		3456789012	4567	LABEL		3456789012	4567
COND		3456789012	4567	PRTPARM		3456789012	4567
SDR1		3456789012	4567	LABEL		3456789012	4567
LABEL		3456789012	4567	PRTPARM		3456789012	4567
CHKPNT		3456789012	4567	LABEL		3456789012	4567
SDR2				PRTPARM		3456789012	4567
SAVE				LABEL		3456789012	4567
				END		3456789012	4567

RIGID FORMAT RESTART TABLES

7.12.5 File Name Restart Table

DMAP		Bit Position				
Inst.	94	100	110	120		
BEGIN						
FILE						
GP1	4					
SAVE	4					
CHKPNT	4					
GP2	5					
CHKPNT	5					
PARAML				6		
PURGE				6		
COND				6		
PLTSET				6		
SAVE				6		
PRTMSG				6		
PARAM				6		
PARAM				6		
COND				6		
PLOT						
SAVE						
PRTMSG						
LABEL						
CHKPNT				6		
GP3	6					
CHKPNT	6					
TA1	7					
SAVE	7					
COND	7					
PURGE	7	2				
CHKPNT	7					
PARAM	8					
PARAM	9					
EMG				8		
SAVE				8		
CHKPNT				8		
COND	8					
EMA	8					
CHKPNT	8					
LABEL	8					
COND	9					
EMA	9					
CHKPNT	9					
COND						
GPWG						
OFF						
LABEL						
EQUIV	0					
CHKPNT	0					
COND	0					
SMA3	0					
CHKPNT	0					
LABEL	0					
PARAM	1					
GP4	1					
SAVE	1					
PARAM	1					
PURGE	1	3	567 90	5	1	
CHKPNT	1	3	567 90	5	1	
COND	2					
GPSP	2					

MODAL FREQUENCY AND RANDOM RESPONSE

DMAP	64	Bit Position		120
Inst.		100	110	
SAVE		2		
COND		2		
OFFP		2		
LABEL		2		
EQUIV		4		
CHKPNT		4		
COND		34		
MCE1		3		
CHKPNT		3		
MCE2		4		
CHKPNT		4		
LABEL		34		
EQUIV		5		
CHKPNT		5		
COND		5		
SCE1		5		
CHKPNT		5		
LABEL		5		
EQUIV		6		7
EQUIV				7
CHKPNT		6		7
COND		6		7
SMP1		6		
CHKPNT		6		
SMP2				7
CHKPNT				7
LABEL		6		7
EQUIV		7		
CHKPNT		7		
COND		7		
RBMG1		7		
CHKPNT		7		
JUMP				
LABEL		7		
COND		890		
LABEL		8		
RBMG2		8		
CHKPNT		8		
COND		9		
RBMG3		9		
CHKPNT		9		
RBMG4		0		
CHKPNT		0		
LABEL		890		
DPD		1		
SAVE		1		
COND		1		
PURGE		1		
EQUIV				5
CHKPNT		1		
PARAM		2		
READ		2		
SAVE		2		
CHKPNT		2		
PARAM				
OFFP		2		
SAVE		2		
COND		2		

RIGID FORMAT RESTART TABLES

DMAP Inst.	64	Bit Position		120
		100	110	
OFF			2	
SAVE			2	
PARAM				
PARAM			3	
JUMP			3	
LABEL			3	
PURGE				
CASE			3	
SAVE			3	
CHKPNT			3	
MTRXIN			4	
SAVE			4	
PURGE			4	
PARAM			4	
EQUIV			4	
CHKPNT			4	
GKAD			5	
CHKPNT			5	
GKAM			6	
SAVE			6	
CHKPNT			6	
COND			7	
COND			7	
FRRD			7	
SAVE			7	
EQUIV			7	
CHKPNT			7	
VDR			8	
SAVE			8	
COND			8	
COND			8	
CHKPNT			8	
SDR3			9	
OFF				
SAVE				
CHKPNT			9	
XYTRAN				
SAVE				
XYPLOT				
JUMP				
LABEL				
OFF				
SAVE				
LABEL				
COND			0	4
PARAM				
COND				
DDR1				4
CHKPNT				4
DDR2			0	
CHKPNT			0	
EQUIV			0	
CHKPNT			0	
EQUIV			1	
COND			1	
SDR1			1	
LABEL			1	
CHKPNT			1	
SDR2			2	

MODAL FREQUENCY AND RANDOM RESPONSE

DMAP	Bit Position			
Inst.	64	100	110	120
SAVE				2
COND				3
SDR3				3
JUMP				
LABEL				
SDR1				9
SDR2				0
SAVE				0
SDR2				1
EQUIV				1
COND				1
SDR3				2
EQUIV				2
DDRRM				3
EQUIV				
JUMP		3		
LABEL				1
DDRRM				4
EQUIV				
JUMP				
LABEL				1
CHKPNT				
OFF			3	
SAVE				
XYTRAN				
SAVE				
XYPLOT				
COND				
PLOT				
SAVE				
LABEL				
COND				5
RANDOM				5
SAVE				5
CHKPNT				5
COND				
XYTRAN				
SAVE				
XYPLOT				
JUMP				
LABEL				
OFF				
SAVE				
LABEL				
COND				
REPT				
JUMP				
JUMP				
LABEL				
PRTPARM				
LABEL				
PRTPARM				
LABEL				
PRTPARM				
LABEL				
PRTPARM				
LABEL				
PRTPARM				
LABEL				
PRTPARM				
LABEL				
END				

MODAL TRANSIENT RESPONSE

7.13 RESTART TABLES FOR MODAL TRANSIENT RESPONSE

7.13.1 Bit Positions for Card Name Restart Table

Card Name	Bit Pos.	Card Name	Bit Pos.	Card Name	Bit Pos.
AXIC	1	CQUAD1	2	PTRSHL	3
AXIF	1	CQUAD2	2	PTUBE	3
CELAS1	1	CQUADTS	2	PTWIST	3
CELAS2	1	CROD	2	GENEL	4
CELAS3	1	CSHEAR	2	CONM1	5
CELAS4	1	CTETRA	2	CONM2	5
CMASS1	1	CTORDRG	2	PELAS	6
CMASS2	1	CTRAPAX	2	PMASS	7
CMASS3	1	CTRAPRG	2	MAT1	8
CMASS4	1	CTRBSC	2	MAT2	8
CORD1C	1	CTRIA1	2	MAT3	8
CORD1R	1	CTRIA2	2	MATT1	8
CORD1S	1	CTRIAAX	2	MATT2	8
CORD2C	1	CTRIARG	2	MATT3	8
CORD2R	1	CTRIATS	2	TABLEM1	8
CORD2S	1	CTRM6	2	TABLEM2	8
GRDSET	1	CTRMEM	2	TABLEM3	8
GRID	1	CTRPLT	2	TABLEM4	8
GRIDB	1	CTRPLT1	2	TEMPMT\$	8
POINTAX	1	CTRSHL	2	TEMPMX\$	8
RINGAX	1	CTUBE	2	AXISYM\$	9
RINGFL	1	CTWIST	2	CRIGD1	9
SECTAX	1	CWEDGE	2	CRIGD2	9
SEQGP	1	PBAR	3	CRIGD3	9
SPOINT	1	PCONEAX	3	CRIGDR	9
ADUM1	2	PDUM1	3	CRIGD3	9
ADUM2	2	PDUM2	3	CRIGDR	9
ADUM3	2	PDUM3	3	MPC	9
ADUM4	2	PDUM4	3	MPCADD	9
ADUM5	2	PDUM5	3	MPCAX	9
ADUM6	2	PDUM6	3	MPC\$	9
ADUM7	2	PDUM7	3	SPC	10
ADUM8	2	PDUM8	3	SPC1	10
ADUM9	2	PDUM9	3	SPCADD	10
BAROR	2	PIHEX	3	SPCAX	10
CBAR	2	PQDMEM	3	SPC\$	10
CCONEAX	2	PQDMEM1	3	ASET	11
CDUM1	2	PQDMEM2	3	ASET1	11
CDUM2	2	PQDMEM3	3	OMIT	11
CDUM3	2	PQDPLT	3	OMIT1	11
CDUM4	2	PQUAD1	3	OMITAX	11
CDUM5	2	PQUAD2	3	SUPAX	12
CDUM6	2	PQUADTS	3	SUPORT	12
CDUM7	2	PROD	3	TEMP	13
CDUM8	2	PSHEAR	3	TEMPAX	13
CDUM9	2	PTORDRG	3	TEMPO	13
CHEXA1	2	PTRAPAX	3	TEMPP1	13
CHEXA2	2	PTRBSC	3	TEMPP2	13
CIHEX1	2	PTRIA1	3	TEMPP3	13
CIHEX2	2	PTRIA2	3	TEMPRB	13
CIHEX3	2	PTRIAAX	3	WTHASS	14
CONROD	2	PTRIATS	3	GRDPNT	15
CQDMEM	2	PTRIM6	3	PLOTEL	16
CQDMEM1	2	PTRIM6	3	PLOTS	18
CQDMEM2	2	PTRMEM	3	POUT\$	19
CQDMEM3	2	PTRPLT	3	XYOUT\$	20
CQDPLT	2	PTRPLT1	3	AOUT\$	21

RIGID FORMAT RESTART TABLES

Card Name	Bit Pos.
LOOPS	22
LOOP1\$	23
COUPMASS	24
CPBAR	24
CPQOPLT	24
CPQUAD1	24
CPQUAD2	24
CPROD	24
CPTRBSC	24
CPTRIA1	24
CPTRIA2	24
CPTRPLT	24
CPTUBE	24
NOLoops	25
AXYOUT\$	27
MODACC	55
EPOINT	56
SEQEP	56
TF	56
DMIAx	57
DMIG	57
B2PP\$	57
K2PP\$	57
M2PP\$	57
TFS	57
DAREA	58
DELAY	58
DLOAD	58
FORCE	58
FORCE1	58
FORCE2	58
GRAV	58
MOMENT	58
MOMENT1	58
MOMENT2	58
NOLIN1	58
NOLIN2	58
NOLIN3	58
NOLIN4	58
PLOAD	58
PLOAD1	58
PLOAD2	58
SLOAD	58
TABLED1	58
TABLED2	58
TABLED3	58
TABLED4	58
TLOAD1	58
TLOAD2	58
TSTEP	58
EIGR	59
METHOD\$	60
OLOAD\$	61
NLFORCE	61
TSTEP\$	61
HFREQ	62
LFREQ	62
LMOOES	62
TABDMP1	62
SDAMPS	62

MODAL TRANSIENT RESPONSE

7.13.2 Bit Positions for File Name Restart Table

<u>File Name</u>	<u>Bit Pos.</u>	<u>File Name</u>	<u>Bit Pos.</u>
BGPD	94	KHH	115
CSTM	94	MHH	115
EQEXIN	94	PHIDH	115
GPOT	94	CASEXX	116
GPL	94	PD	117
SIL	94	POT	117
ECT	95	PH	117
GPTT	96	PPT	117
SLT	96	PST	117
EST	97	TOL	117
GEI	97	OPNL1	118
GPECT	97	OUHV1	118
GPST	98	OPNL2	119
KGGX	98	OUHV2	119
MGG	99	PAF	120
KGG	100	UDV2T	120
ASET	101	UEVT	120
RG	101	QP	121
USET	101	UPV	121
OGPST	102	OEF1	122
GM	103	OES1	122
KNN	104	OPP1	122
MNN	104	OQP1	122
KFF	105	OUPV1	122
KFS	105	PUGV	122
MFF	105	OEF2	123
GO	106	OES2	123
KAA	106	OPP2	123
KLL	107	OQP2	123
KLR	107	OUPV2	123
KRR	107	UDV1T	124
MLL	107	ELSETS	125
MLR	107	GPSETS	125
MRR	107	PLTPAR	125
LLL	108	PLTSETX	125
DM	109	MAA	126
MR	110	KDICT	127
OLT	111	KELM	127
EED	111	MDICT	127
EQDYN	111	MELM	127
GPLD	111	PNLH	128
NLFT	111	UHV1	128
SILD	111	PHIPH	129
TFPOOL	111	QPH	129
TRL	111	IEF1	130
USETD	111	IES1	130
LAMA	112	IPHIP1	130
MI	112	IQP1	130
OEIGS	112	IEF2	131
PHIA	112	IES2	131
B2PP	113	IPHIP2	131
K2PP	113	IQP2	131
M2PP	113	OPPB	131
B2DD	114	ZEF2	132
GMD	114	ZES2	132
GOD	114	ZQP2	132
K2DD	114	ZUPV2	132
M2DD	114		
MDD	114		
BHH	115		

RIGID FORMAT RESTART TABLES

7.13.3 Card Name Restart Table

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
BEGIN	1234567890123456	8901234					
FILE	1234567890123456	8901234					56789012
GP1	1						56789012
SAVE	1						
CHKPNT	1						
\$\$\$		6					
GP2	12 45		6				
CHKPNT	12 45		6				
\$\$\$		6					
PARAML				8			
\$\$\$		7					
PURGE				8			
\$\$\$		7					
COND				8			
\$\$\$		7					
PLTSET				8			
\$\$\$		7					
SAVE				8			
\$\$\$		7					
PRTMSG				8			
\$\$\$		7					
PARAM				8			
\$\$\$		7					
PARAM				8			
\$\$\$		7					
COND				8			
\$\$\$		7					
PLOT				8			
\$\$\$		7					
SAVE				8			
\$\$\$		7					
PRTMSG				8			
\$\$\$		7					
LABEL				8			
\$\$\$		7					
CHKPNT				8			
\$\$\$		67					
GP3	12		3				
CHKPNT	12		3				1
\$\$\$		6					1
TA1	1234567		3				
SAVE	1234567		3				
COND	12345678		34		4		
PURGE	1234567		3				
CHKPNT	1234567		3				
\$\$\$		6					
PARAM	123 6 8						
PARAM	123 5 7 8		4		4		
EMG	123 567 8		4		4		
SAVE	123 567 8		4		4		
CHKPNT	123 567 8		4		4		
\$\$\$		6					
COND	123 6 8						
EMA	123 6 8						
CHKPNT	123 6 8						
\$\$\$		6					
LABEL	123 6 8						
COND	123 5 7 8		4		4		
EMA	123 5 7 8		4		4		

MODAL TRANSIENT RESPONSE

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
CHKPNT	123 5 7 8	4	4				
\$\$\$	6						
COND	123 5 7 8	4 5	4				
\$\$\$	8						
GPWG	123 5 7 8	4 5	4				
\$\$\$	8						
OFF	123 5 7 8	4 5	4				
\$\$\$	8						
LABEL	123 5 7 8	4 5	4				
\$\$\$	8						
EQUIV	1234 8						
CHKPNT	1234 6 8						
\$\$\$	6						
COND	1234 6 8						
SMA3	1234 6 8						
CHKPNT	1234 6 8						
\$\$\$	6						
LABEL	1234 6 8						
PARAM	1 9012						
GP4	1 9012						
SAVE	1 9012						
PARAM	1 9012						
PURGE	1 9012						
CHKPNT	123456789012 4		4				
\$\$\$	6						
COND	123 6 8 90						
GPSP	123 6 8 90						
SAVE	123 6 8 90						
COND	123 6 8 90						
OFF	123 6 8 90						
LABEL	123 6 8 90						
EQUIV	123456789	4	4				
CHKPNT	123456789	4	4				
\$\$\$	6						
COND	123456789	4	4				
MCE1	1 9						
CHKPNT	1 9						
\$\$\$	6						
MCE2	123456789	4	4				
CHKPNT	123456789	4	4				
\$\$\$	6						
LABEL	123456789	4	4				
EQUIV	1234567890	4	4				
CHKPNT	1234567890	4	4				
\$\$\$	6						
COND	1234567890	4	4				
SCE1	1234567890	4	4				
CHKPNT	1234567890	4	4				
\$\$\$	6						
LABEL	1234567890	4	4				
EQUIV	1234 6 8 901						
EQUIV	12345 7 8 901	4	4				
CHKPNT	12345678901	4	4				
\$\$\$	6						
COND	12345678901	4	4				
SMP1	1234 6 8 901						
CHKPNT	1234 6 8 901						
\$\$\$	6						
SMP2	12345678901	4	4				

RIGID FORMAT RESTART TABLES

DMAP	Bit Position						
Inst.	1	10	20	30	40	50	60
CHKPNT	12345678901	4		4			
SSS	6						
LABEL	12345678901	4		4			
EQUIV	123456789012	4		4			
CHKPNT	123456789012	4		4			
SSS	6						
COND	123456789012	4		4			
RBMG1	123456789012	4		4			
CHKPNT	123456789012	4		4			
SSS	6						
JUMP	1234 6 89012						
LABEL	1234 6 89012						
COND	1234 6 89012						
LABEL	1234 6 89012						
RBMG2	1234 6 89012						
CHKPNT	1234 6 89012						
SSS	6						
COND	1234 6 89012						
RBMG3	1234 6 89012						
CHKPNT	1234 6 89012						
SSS	6						
RBMG4	123456789012	4		4			
CHKPNT	123456789012	4		4			
SSS	6						
LABEL	123456789012	4		4			
DPD	1 9012					6 89	
SAVE	1 9012					6 89	
COND	1 9012					6 89	
PURGE	1 9012					6 89	
EQUIV	123456789012	4		4		567 90	
CHKPNT	1 9012					6 89	
SSS	6						
PARAM	123456789012	4		4			
READ	123456789012	4		4			0
SAVE	123456789012	4		4			0
CHKPNT	123456789012	4		4			0
SSS	6						
PARAM			9 1				
OFF	123456789012	4		4			0
SAVE	123456789012	4		4			0
COND	123456789012	4		4			0
OFF	123456789012	4		4			0
SAVE	123456789012	4		4			0
MTRXIN	1					67	
SAVE	1					67	
PURGE	12345678901					67	
PARAM	12345678901					67	
EQUIV	12345678901					67	
CHKPNT	12345678901					67	
SSS	6						
GKAD	12345678901	4		4		567	0
CHKPNT	12345678901	4		4		567	0
SSS	6						
GKAM	123456789012	4		4		67 90	2
SAVE	123456789012	4		4		67 90	2
CHKPNT	123456789012	4		4		67 90	2
SSS	6						
COND	123456789012	4		4		67 90	2
PARAM			23				
PARAM	12345678901234 6	901234	7			56789012	

MODAL TRANSIENT RESPONSE

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
JUMP		4	234				
\$\$\$	1 3						
LABEL	1234567890123456	8901234				56789012	
\$\$\$	1 3						
PURGE			90123	7			
CASE	12345678901234	6	9012345	7		56789012	
SAVE	12345678901234	6	9012345	7		56789012	
CHKPNT	12345678901234	6	9012345	7		56789012	
\$\$\$	6						
PAPAM	123456789012	4	4			67890	
\$\$\$	4						
TRLG	123456789012	4	234			6789012	
SAVE	123456789012	4	234			6789012	
CHKPNT	123456789012	4	234			6789012	
\$\$\$	6						
EQUIV	123456789012	4	234			6789012	
CHKPNT	123456789012	4	234			6789012	
\$\$\$	6						
TRD	123456789012	4	234			6789012	
SAVE	123456789012	4	234			6789012	
CHKPNT	123456789012	4	234			6789012	
\$\$\$	6						
VDR			901	7			
SAVE			901	7			
CHKPNT			1	7			
\$\$\$	6						
COND			1	7			
SDR3			1	7			
OFF			1				
SAVE			1				
CHKPNT			1				
\$\$\$	6						
XYTRAN				7			
\$\$\$	7						
SAVE				7			
\$\$\$	7						
XYPLOT				7			
\$\$\$	7						
LABEL			1				
PARAM	123456789012	4	234			56789012	
COND	123456789012	4	234			56789012	
PARAM	123456789012	4	234			56789012	
PARAM	123456789012	4	234			56789012	
COND	123456789012	4	234			56789012	
DDR1	123456789012	4	234			6789012	
CHKPNT	123456789012	4	234			6789012	
\$\$\$	6						
COND	123456789012	4	234			56789012	
\$\$\$	23						
DDR2	123456789012	4	234			56789012	
\$\$\$	23						
CHKPNT	123456789012	4	234			56789012	
\$\$\$	23 6						
EQUIV	123456789012	4	234			56789012	
\$\$\$	23						
CHKPNT	123456789012	4	234			56789012	
\$\$\$	23 6						
LABEL	123456789012	4	234			56789012	
\$\$\$	23						

RIGID FORMAT RESTART TABLES

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
EQUIV	123456789012	4	234				56789012
COND	123456789012	4	234				56789012
SDR1	123456789012	4	234				56789012
LABEL	123456789012	4	234				56789012
CHKPNT	123456789012	4	234				56789012
\$\$\$	6						
SDR2			890				
SDR3			890				
JUMP			890				
LABEL	123456789012	4	234				56789012
SDR1	123456789012	4	234				56789012
SDR2			890				
SDR2			890				
SDR3			890				
EQUIV			890				
DDRMH			890				
EQUIV			890				
LABEL			890				
CHKPNT			890				
\$\$\$	6						
OFF			9				
SAVE			9				
COND			8				
\$\$\$	7						
PLOT			8				
\$\$\$	7						
SAVE			8				
\$\$\$	7						
PRTMSG			8				
\$\$\$	7						
LABEL			8				
\$\$\$	7						
XYTRAN			0				
\$\$\$	7						
SAVE			0				
\$\$\$	7						
XYPLOT			0				
\$\$\$	7						
LABEL			0				
COND			23				
\$\$\$	1 3						
REPT			23				
\$\$\$	1 3						
JUMP			23				
\$\$\$	1 3						
JUMP	1234567890123456	8901234					56789012
LABEL			23				
\$\$\$	1 3						
PRTPARM			23				
\$\$\$	1 3						
LABEL	1234567890123456	8901234					56789012
PRTPARM	1234567890123456	8901234					56789012
LABEL	1234567890123456	8901234					56789012
PRTPARM	1234567890123456	8901234					56789012
LABEL	1234567890123456	8901234					56789012
PRTPARM	1234567890123456	8901234					56789012
LABEL	1234567890123456	8901234					56789012
PRTPARM	1234567890123456	8901234					56789012
LABEL	1234567890123456	8901234					56789012
END	1234567890123456	8901234					56789012

MODAL TRANSIENT RESPONSE

7.13.4 Rigid Format Change Restart Table

DMAP Inst.	63	Bit Position 70	80	DMAP Inst.	63	Bit Position 70	80
BEGIN		34567890123	567 345	COND			
FILE		34567890123	567 345	OFF			
GP1				LABEL			
SAVE				EQUIV			
CHKPNT				CHKPNT			
GP2				COND			
CHKPNT				MCE1			
PARAML				CHKPNT			
PURGE				MCE2			
COND				CHKPNT			
PLTSET				LABEL			
SAVE				EQUIV			
PRTMSG				CHKPNT			
PARAM				COND			
PARAM				SCE1			
COND				CHKPNT			
PLOT				LABEL			
SAVE				EQUIV			
PRTMSG				EQUIV			
LABEL				CHKPNT			
CHKPNT				COND			
GP3				SMP1			
CHKPNT				CHKPNT			
TA1				SMP2			
SAVE				CHKPNT			
COND				LABEL			
PURGE				EQUIV			
CHKPNT				CHKPNT			
PARAM				COND			
PARAM	3	678		RBMG1			
EMG	3	678		CHKPNT			
SAVE	3	678		JUMP			
CHKPNT	3	678		LABEL			
COND				COND			
EMA				LABEL			
CHKPNT				RBMG2			
LABEL				CHKPNT			
COND	3	678		COND			
EMA	3	678		REMG3			
CHKPNT	3	678		CHKPNT			
COND				RBMG4			
GPWG				CHKPNT			
OFF				LABEL			
LABEL				DPD			
EQUIV				SAVE			
CHKPNT				COND	34567890123	567	345
COND				PURGE			
SMA3				EQUIV			
CHKPNT				CHKPNT			
LABEL				PARAM			
PARAM				READ			
GP4				SAVE			
SAVE				CHKPNT			
PARAM				PARAM			
PURGE				OFF			
CHKPNT				SAVE			
COND				COND	34567890123	567	345
GPSP				OFF			
SAVE				SAVE			

RIGID FORMAT RESTART TABLES

DMAP Inst.	63	Bit Position 70	80	DMAP Inst.	63	Bit Position 70	80
MTRXIN				SDR3			
SAVE				JUMP			
PURGE				LABEL			
PARAM				SDR1	34567890123	567	345
EQUIV				SDR2			
CHKPNT				SDR2			
GKAD				SDR3			
CHKPNT				EQUIV			
GKAM				DDRMH			
SAVE				EQUIV			
CHKPNT				LABEL			
COND	34567890123	567	345	CHKPNT			
PARAM				OFF			
PARAM	34567890123	567	345	SAVE			
JUMP	34567890123	567	345	COND			
LABEL	34567890123	567	345	PLOT			
PURGE				SAVE			
CASE	34567890123	567	345	PRMSG			
SAVE	34567890123	567	345	LABEL			
CHKPNT	34567890123	567	345	XYTRAN			
PARAM				SAVE			
TRLG				XYPLOT			
SAVE				LABEL			
CHKPNT				COND	34567890123	567	345
EQUIV				REPT	34567890123	567	345
CHKPNT				JUMP	34567890123	567	345
TRD				JUMP	34567890123	567	345
SAVE				LABEL	34567890123	567	345
CHKPNT				PRTPARM	34567890123	567	345
VDR				LABEL	34567890123	567	345
SAVE				PRTPARM	34567890123	567	345
CHKPNT				LABEL	34567890123	567	345
COND				PRTPARM	34567890123	567	345
SDR3				LABEL	34567890123	567	345
OFF				LABEL	34567890123	567	345
SAVE				PRTPARM	34567890123	567	345
CHKPNT				LABEL	34567890123	567	345
XYTRAN				PRTPARM	34567890123	567	345
SAVE				LABEL	34567890123	567	345
XYPLOT				END	34567890123	567	345
LABEL							
PARAM							
COND							
PARAM							
PARAM							
COND							
DDR1							
CHKPNT							
COND							
DDR2							
CHKPNT							
EQUIV							
CHKPNT							
LABEL							
EQUIV							
COND	34567890123	567	345				
SDR1	34567890123	567	345				
LABEL	34567890123	567	345				
CHKPNT	34567890123	567	345				
SDR2							

MODAL TRANSIENT RESPONSE

7.13.5 File Name Restart Table

DMAP	Bit Position			
Inst.	64	100	110	120
BEGIN				
FILE				
GP1	4			
SAVE	4			
CHKPNT	4			
GP2	5			
CHKPNT	5			
PARAML				5
PURGE				5
COND				5
PLTSET				5
SAVE				5
PRTMSG				5
PARAM				5
PARAM				5
COND				
PLOT				
SAVE				
PRTMSG				
LABEL				
CHKPNT				5
GP3	6			
CHKPNT	6			
TA1	7			
SAVE	7			
COND	7			
PURGE	7	2		
CHKPNT	7			
PARAM	8			
PARAM	9			
EMG				7
SAVE				7
CHKPNT				7
COND	8			
EMA	8			
CHKPNT	8			
LABEL	8			
COND	9			
EMA	9			
CHKPNT	9			
COND				
GPWG				
OFF				
LABEL				
EQUIV	0			
CHKPNT	0			
COND	0			
SMA3	0			
CHKPNT	0			
LABEL	0			
PARAM	1			
GP4	1			
SAVE	1			
PARAM	1			
PURGE	1	3	567 90	4
CHKPNT	1	3	567 90	4
COND	2			
GPSP	2			

RIGID FORMAT RESTART TABLES

DMAP		Bit Position		
Inst.	64	100	110	120
SAVE		2		
COND		2		
OFF		2		
LABEL		2		
EQUIV		4		
CHKPNT		4		
COND		34		
MCE1		3		
CHKPNT		3		
MCE2		4		
CHKPNT		4		
LABEL		34		
EQUIV		5		
CHKPNT		5		
COND		5		
SCE1		5		
CHKPNT		5		
LABEL		5		
EQUIV		6		6
EQUIV				6
CHKPNT		6		6
COND		6		6
SMP1		6		
CHKPNT		6		
SMP2				6
CHKPNT				6
LABEL		6		6
EQUIV		7		
CHKPNT		7		
COND		7		
RBMG1		7		
CHKPNT		7		
JUMP				
LABEL		7		
COND		890		
LABEL		8		
RBMG2		8		
CHKPNT		8		
COND		9		
RBMG3		9		
CHKPNT		9		
RBMG4		0		
CHKPNT		0		
LABEL		890		
OPD		1		
SAVE		1		
COND		1		
PURGE		1		
EQUIV			4	
CHKPNT		1		
PARAM		2		
READ		2		
SAVE		2		
CHKPNT		2		
PARAM				
OFF		2		
SAVE		2		
COND		2		
OFF		2		

MODAL TRANSIENT RESPONSE

DMAP	64	Bit Position		120
Inst.	100	110		
SAVE		2		
MTRXIN		3		
SAVE		3		
PURGE		3		
PARAM		3		
EQUIV		3		
CHKPNT		3		
GKAD		4		
CHKPNT		4		
GKAM		5		
SAVE		5		
CHKPNT		5		
COND		7		
PARAM				
PARAM				
JUMP				
LABEL				
PURGE				
CASE		6		
SAVE		6		
CHKPNT		6		
PARAM				
TRLG		7		
SAVE		7		
CHKPNT		7		
EQUIV		7		
CHKPNT		7		
TRD			8	
SAVE			8	
CHKPNT			8	
VDR		8		
SAVE		8		
CHKPNT		8		
COND		9		
SDR3		9		
OFF				
SAVE				
CHKPNT		9		
XYTRAN				
SAVE				
XYPLOT				
LABEL		9		
PARAM		0		
COND		0		
PARAM				
PARAM				
COND				
DDR1			4	
CHKPNT			4	
COND		0		
DDR2		0		
CHKPNT		0		
EQUIV		0		
CHKPNT		0		
LABEL		0		
EQUIV		1		
COND		1		
SDR1		1		
LABEL		1		

7.13-14 (12/29/78)

DMAP	Inst.	64	100	Bit Position	110	120
CHKPNT						1
SDR2						2
SDR3						3
JUMP						
LABEL						
SDR1						9
SDR2						0
SDR2						1
SDR3						2
EQUIV						2
DDRMH						3
EQUIV						1 3
LABEL						
CHKPNT						3
OFF						
SAVE						
COND						
PLOT						
SAVE						
PRTMSG						
LABEL						
XYTRAN						
SAVE						
XYPLOT						
LABEL						
COND						
REPT						
JUMP						
JUMP						
LABEL						
PRTPARM						
LABEL						
PRTPARM						
LABEL						
PRTPARM						
LABEL						
PRTPARM						
LABEL						
PRTPARM						
LABEL						
END						

NORMAL MODES WITH DIFFERENTIAL STIFFNESS

7.14 RESTART TABLES FOR NORMAL MODES WITH DIFFERENTIAL STIFFNESS

7.14.1 Bit Positions for Card Name Restart Table

<u>Card Name</u>	<u>Bit Pos.</u>	<u>Card Name</u>	<u>Bit Pos.</u>	<u>Card Name</u>	<u>Bit Pos.</u>
AXIC	1	CIHEX2	2	PTRAPAX	3
AXIF	1	CIHEX3	2	PTRBSC	3
CELAS1	1	CONROD	2	PTRIA1	3
CELAS2	1	CQDMEM	2	PTRIA2	3
CELAS3	1	CQDMEM1	2	PTRIAAX	3
CELAS4	1	CQDMEM2	2	PTRIATS	3
CMASS1	1	CQDMEM3	2	PTRIM6	3
CMASS2	1	CQDPLT	2	PTRMEM	3
CMASS3	1	CQUAD1	2	PTRPLT	3
CMASS4	1	CQUAD2	2	PTRPLT1	3
CORD1C	1	CQUADTS	2	PTRSHL	3
CORD1R	1	CROD	2	PTUBE	3
CORD1S	1	CSHEAR	2	PTWIST	3
CORD2C	1	CTETRA	2	GENEL	4
CORD2R	1	CTORDRG	2	CONM1	5
CORD2S	1	CTRAPAX	2	CONM2	5
FREETPT	1	CTRAPRG	2	FSLIST	5
GRDSET	1	CTRBSC	2	PELAS	6
GRID	1	CTRIA1	2	PMASS	7
GRIDB	1	CTRIA2	2	MAT1	8
POINTAX	1	CTRIAAX	2	MAT2	8
PRESPT	1	CTRIARG	2	MAT3	8
RINGAX	1	CTRIATS	2	MATT1	8
RINGFL	1	CTRIM6	2	MATT2	8
SECTAX	1	CTRMEM	2	MATT3	8
SEQGP	1	CTRPLT	2	TABLEM1	8
SPOINT	1	CTRPLT1	2	TABLEM2	8
ADUM1	2	CTRSHL	2	TABLEM3	8
ADUM2	2	CTUBE	2	TABLEM4	8
ADUM3	2	CTWIST	2	TEMPMTS	8
ADUM4	2	CHEDGE	2	TEMPMXS	8
ADUM5	2	PBAR	3	AXISYMS	9
ADUM6	2	PCONEAX	3	CRIGD1	9
ADUM7	2	PDUM1	3	CRIGD2	9
ADUM8	2	PDUM2	3	CRIGD3	9
ADUM9	2	PDUM3	3	CRIGDR	9
BAROR	2	PDUM4	3	MPC	9
CBAR	2	PDUM5	3	MPCADD	9
CCONEAX	2	PDUM6	3	MPCAX	9
CDUM1	2	PDUM7	3	MPCS	9
CDUM2	2	PDUM8	3	SPC	10
CDUM3	2	PDUM9	3	SPC1	10
CDUM4	2	PIHEX	3	SPCADD	10
CDUM5	2	PQDMEM	3	SPCAX	10
CDUM6	2	PQDMEM1	3	SPCS	10
CDUM7	2	PQDMEM2	3	ASET	11
CDUM8	2	PQDMEM3	3	ASET1	11
CDUM9	2	PQDPLT	3	OMIT	11
CFLUID2	2	PQUAD1	3	OMIT1	11
CFLUID3	2	PQUAD2	3	OMITAX	11
CFLUID4	2	PQUADTS	3	TEMP	13
CHEXA1	2	PROD	3	TEMPAX	13
CHEXA2	2	PSHEAR	3	TEMPO	13
CIHEX1	2	PTORDRG	3	TEMPP1	13

RIGID FORMAT RESTART TABLES

<u>Card Name</u>	<u>Bit Pos.</u>
TEMPP2	13
TEMPP3	13
TEMPRB	13
WTMASS	14
GRDPNT	15
PLOTEL	16
IRES	17
PLOTS	18
POUTS	19
DSFACT	21
DSCOS	21
COUPMASS	24
CPBAR	24
CPQDPLT	24
CPQUAD1	24
CPQUAD2	24
CPR00	24
CPTRBSC	24
CPTRIA1	24
CPTRIA2	24
CPTRPLT	24
CPTUBE	24
GRAV	57
RFORCE	57
TEMPLDS	58
DEFORM	59
DEFORMS	59
LOADS	59
RFORCES	59
SPCD	59
FORCE	60
FORCE1	60
FORCE2	60
FORCEAX	60
LOAD	60
MOMAX	60
MOMENT	60
MOMENT1	60
MOMENT2	60
PLOAD	60
PLOAD1	60
PLOAD2	60
PLOAD3	60
PRESAX	60
SLOAD	60
EIGR	61
METHODS	62

NORMAL MODES WITH DIFFERENTIAL STIFFNESS

7.14.2 Bit Positions for File Name Restart Table

<u>File Name</u>	<u>Bit Pos.</u>	<u>File Name</u>	<u>Bit Pos.</u>
BGPD T	94	KDGG	113
CSTM	94	KONN	114
EQEXIN	94	KOFF	115
GPD T	94	KDFS	115
GPL	94	KDSS	115
SIL	94	KDAA	116
ECT	95	EED	117
GPTT	96	EQDYN	117
SLT	96	GPLD	117
EST	97	SILD	117
GEI	97	USETD	117
GPECT	97	LAMA	118
GPST	98	OEIGS	118
KGGX	98	PHIA	118
MGG	99	QG	119
KGG	100	PHIG	119
ASET	101	OB EF1	120
RG	101	OBES1	120
USET	101	OBQG1	120
YS	101	OPHIG	120
OGPST	102	PPHIG	120
GM	103	ELSETS	121
KNN	104	GPSETS	121
MNN	104	PLTPAR	121
KFF	105	PLTSETX	121
KFS	105	MAA	122
KSS	105	KDICT	123
MFF	105	KELM	123
GO	106	MDICT	123
KAA	106	MELM	123
KOO	106	PBL	125
LOO	106	PBS	125
LLL	107	UBOOV	125
PG	108	YBS	125
PL	109	KBLL	126
PO	109	K9FS	126
PS	109	KBSS	126
RULV	110	LBLL	127
RUOV	110	UBLV	128
ULV	110	RUBLV	128
UOOV	110	UBGV	129
PGG	111	QBG	129
QG	111	QBG1	130
UGV	111	OUBGV1	130
OE F1	112	OESB1	130
OES1	112	OE FB1	130
OPG1	112	PUBGV1	130
OQG1	112		
OUGV1	112		
PUGV1	112		

RIGID FORMAT RESTART TABLES

7.14.3 Card Name Restart Table

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
BEGIN	12345678901	3456789	4				789012
FILE	12345678901	3456789	4				789012
GP1	1						
SAVE	1						
CHKPNT	1						
SSS		6					
GP2	12 45		6				
CHKPNT	12 45		6				
SSS		6					
PARAML			8				
SSS		7					
PURGE			8				
SSS		7					
COND			8				
SSS		7					
PLTSET			8				
SSS		7					
SAVE			8				
SSS		7					
PRTMSG			8				
SSS		7					
SETVAL			8				
SSS		7					
SAVE			8				
SSS		7					
COND			8				
SSS		7					
PLOT			8				
SSS		7					
SAVE			8				
SSS		7					
PRTMSG			8				
SSS		7					
LABEL			8				
SSS		7					
CHKPNT			8				
SSS		67					
GP3	12		3				7 0
CHKPNT	12		3				7 0
SSS		6					
TA1	1234567		3				
SAVE	1234567		3				
COND	12345678		3				
PURGE	1234567		3				
CHKPNT	1234567		3				
SSS		6					
PARAM	123 6 8						
PARAM	123 6 8						
EMG	123 5678		45				7
SAVE	123 5678		45				7
CHKPNT	123 5678		45				7
SSS		6					
COND	123 6 8						
EMA	123 6 8						
CHKPNT	123 6 8						
SSS		6					
LABEL	123 6 8						
COND	123 5 7 8		45	4			7
EMA	123 5 7 8		45				7

NORMAL MODES WITH DIFFERENTIAL STIFFNESS

DMAP Inst.	Bit Position						
	1	10	20	30	40	50	60
CHKPNT	123 5 7 8		45				7
SSS		6					
COND	123 5 7 8		45	4			
SSS		8					
GPWG	123 5 7 8		45	4			
SSS		8					
OFP	123 5 7 8		45	4			
SSS		8					
LABEL	123 5 7 8		45				7
EQUIV	1234 6 8						
CHKPNT	1234 6 8						
SSS		6					
COND	1234 6 8						
SMA3	1234 6 8						
CHKPNT	1234 6 8						
SSS		6					
LABEL	1234 6 8						
PARAM	1	901					
GP4	1	901					9
SAVE	1	901					9
COND	1	901					9
PARAM	1	901					9
PURGE	1	901					9
CHKPNT	1234 6 8 9 0 1						9
SSS		6					
COND	1						
JUMP	1						
LABEL	1						
COND	123 6 8 9 0						
GPSP	123 6 8 9 0						
SAVE	123 6 8 9 0						
COND	123 6 8 9 0						
OFP	123 6 8 9 0						
LABEL	123 6 8 9 0						
EQUIV	1234 6 8 9						
CHKPNT	1234 6 8 9						
SSS		6					
COND	1234 6 8 9						
MCE1	1	9					
CHKPNT	1	9					
SSS		6					
MCE2	1234 6 8 9						
CHKPNT	1234 6 8 9						
SSS		6					
LABEL	1234 6 8 9						
EQUIV	1234 6 8 9 0						
CHKPNT	1234 6 8 9 0						
SSS		6					
COND	1234 6 8 9 0						
SCE1	1234 6 8 9 0						
CHKPNT	1234 6 8 9 0						
SSS		6					
LABEL	1234 6 8 9 0						
EQUIV	1234 6 8 9 0 1						
CHKPNT	1234 6 8 9 0 1						
SSS		6					
COND	1234 6 8 9 0 1						
SMP1	1234 6 8 9 0 1						

RIGID FORMAT RESTART TABLES

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
CHKPNT	1234	6 8901					
SSS		6					
LABEL	1234	6 8901					
RBMG2	1234	6 8901					
CHKPNT	1234	6 8901					
SSS		6					
SSG1	123	5678					7890
CHKPNT	123	5678					7890
SSS		6					
EQUIV	123	5678901					7890
CHKPNT	123	5678901					7890
SSS		6					
COND	123	5678901					7890
SSG2	123	5678901					7890
CHKPNT	123	5678901					7890
SSS		6					
LABEL	123	5678901					7890
SSG3	1234	5678901					7890
SAVE	1234	5678901					7890
CHKPNT	1234	5678901					7890
SSS		6					
COND	1234	5678901	7				7890
MATGPR	1234	5678901	7				7890
MATGPR	1234	5678901	7				7890
LABEL	1234	5678901	7				7890
SDR1	1234	5678901					7890
CHKPNT	1234	5678901					7890
SSS		6					
SDR2			9				
PARAM			9				
OFF			9				
SAVE			9				
COND			8				
SSS		7					
PLOT			8				
SSS		7					
SAVE			8				
SSS		7					
PRTHSG			8				
SSS		7					
LABEL			8				
SSS		7					
TA1	1234	567890					7890
DSMG1	1234	5678901					7890
SAVE	1234	5678901					7890
CHKPNT	1234	5678901					7890
EQUIV	1234	5678901					7890
CHKPNT	1234	5678901					7890
SSS		6					
COND	1234	5678901					7890
MCE2	1234	5678901					7890
CHKPNT	1234	5678901					7890
SSS		6					
LABEL	1234	5678901					7890
EQUIV	1234	5678901					7890
CHKPNT	1234	5678901					7890
SSS		6					
COND	1234	5678901					7890

NORMAL MODES WITH DIFFERENTIAL STIFFNESS

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
SCE1	12345678901						7890
CHKPNT	12345678901						7890
\$\$\$	6						
LABEL	12345678901						7890
EQUIV	12345678901						7890
CHKPNT	12345678901						7890
\$\$\$	6						
COND	12345678901						7890
SMP2	12345678901						7890
SMP2	12345678901						7890
CHKPNT	12345678901						7890
\$\$\$	6						
LABEL	12345678901						7890
PARAM	12345678901						7890
EQUIV	12345678901						7890
CHKPNT	12345678901						7890
\$\$\$	6						
PARAM	12345678901		1				7890
DSMG2	12345678901		1				7890
SAVE	12345678901		1				7890
CHKPNT	12345678901		1				7890
\$\$\$	6						
RBMG2	12345678901		1				7890
SAVE	12345678901		1				7890
CHKPNT	12345678901		1				7890
\$\$\$	6						
PRTPARM	12345678901		1				7890
PRTPARM	12345678901		1				7890
SSG3	12345678901		1				7890
SAVE	12345678901		1				7890
CHKPNT	12345678901		1				7890
\$\$\$	6						
COND	12345678901		1				7890
MATGPR	12345678901		1				7890
LABEL	12345678901		1				7890
SDR1	12345678901		1				7890
CHKPNT	12345678901		1				7890
\$\$\$	6						
SDR2	12345678901		1				7890
OFF	12345678901		1				7890
OPD	12345678901		1				7890
SAVE	12345678901		1				7890
COND	12345678901		1				7890
CHKPNT	12345678901		1				7890
\$\$\$	6						
PARAM	12345678901		1				789012
READ	12345678901		1				789012
SAVE	12345678901		1				789012
CHKPNT	12345678901		1				789012
\$\$\$	6						
OFF	12345678901		1				789012
SAVE	12345678901		1				789012
COND	12345678901		1				789012
OFF	12345678901		1				789012
SAVE	12345678901		1				789012
SDR1	12345678901		1				789012
CHKPNT	12345678901		1				789012

RIGID FORMAT RESTART TABLES

DMAP Inst.	1	10	20	Bit Position <u>30</u>	40	50	60
\$\$\$	6						
CASE			89				
SDR2			89				
OFF			9				
SAVE			9				
COND			8				
\$\$\$	7						
PLOT			8				
\$\$\$	7						
SAVE			8				
\$\$\$	7						
PRTMSG			8				
\$\$\$	7						
LABEL			8				
\$\$\$	7						
JUMP	12345678901	3456789	4			789012	
LABEL	12345678901	3456789	4			789012	
PRTPARM	12345678901	3456789	4			789012	
LABEL	12345678901	3456789	4			789012	
PRTPARM	12345678901	3456789	4			789012	
LABEL	12345678901	3456789	4			789012	
PRTPARM	12345678901	3456789	4			789012	
LABEL	12345678901	3456789	4			789012	
PRTPARM	12345678901	3456789	4			789012	
LABEL	12345678901	3456789	4			789012	
\$\$\$	8						
PRTPARM	12345678901	3456789	4			789012	
\$\$\$	8						
LABEL	12345678901	3456789	4			789012	
PRTPARM	12345678901	3456789	4			789012	
LABEL	12345678901	3456789	4			789012	
END	12345678901	3456789	4			789012	

NORMAL MODES WITH DIFFERENTIAL STIFFNESS

7.14.4 Rigid Format Change Restart Table

DMAP Inst.	Bit Position			DMAP Inst.	Bit Position		
	63	70	80		63	70	80
BEGIN	345678901234	67	345	JUMP	345	901234	
FILE	345678901234	67	345	LABEL	345	901234	
GP1				COND			
SAVE				GPSP			
CHKPNT				SAVE			
GP2				COND			
CHKPNT				OFF			
PAPAML				LABEL			
PURGE				EQUIV			
COND				CHKPNT			
PLTSET				COND			
SAVE				MCE1			
PRTMSG				CHKPNT			
SETVAL				MCE2			
SAVE				CHKPNT			
COND				LABEL			
PLOT				EQUIV			
SAVE				CHKPNT			
PRTMSG				COND			
LABEL				SCE1			
CHKPNT				CHKPNT			
GP3				LABEL			
CHKPNT				EQUIV			
TA1				CHKPNT			
SAVE				COND			
COND	345678901234	67	345	SMP1			
PURGE				CHKPNT			
CHKPNT				LABEL			
PARAM				RBMG2			
PARAM				CHKPNT			
EMG				SSG1			
SAVE				CHKPNT			
CHKPNT				EQUIV			
COND				CHKPNT			
EMA				COND			
CHKPNT				SSG2			
LABEL				CHKPNT			
COND				LABEL			
EMA				SSG3	4		
CHKPNT				SAVE	4		
COND				CHKPNT	4		
GPWG				COND	45	8901234	
OFF				MATGPR	45	8901234	
LABEL				MATGPR	45	8901234	
EQUIV				LABEL	45	8901234	
CHKPNT				SDR1			
COND				CHKPNT			
SMA3				SDR2			
CHKPNT				PARAM			
LABEL				OFF			
PARAM				SAVE			
GP4				COND			
SAVE				PLOT			
COND	345678901234	67	345	SAVE			
PARAM				PRTMSG			
PURGE				LABEL			
CHKPNT				TA1			
COND	345	901234		DSMG1			

RIGID FORMAT RESTART TABLES

DMAP Inst.	63	<u>Bit Position</u> 70	80	DMAP Inst.	63	<u>Bit Position</u> 70	80
SAVE				CASE			
CHKPNT				SDR2			
EQUIV				OFF			
CHKPNT				SAVE			
COND				COND			
MCE2				PLOT			
CHKPNT				SAVE			
LABEL				PRTMSG			
EQUIV				LABEL			
CHKPNT				JUMP	345	78901234 67	345
COND				LABEL	345	78901234 67	345
SCE1				PRTPARM	345	78901234 67	345
CHKPNT				LABEL	345	78901234 67	345
LABEL				PRTPARM	345	78901234 67	345
EQUIV				LABEL	345	78901234 67	345
CHKPNT				PRTPARM	345	78901234 67	345
COND				LABEL	345	78901234 67	345
SMP2				PRTPARM	345	78901234 67	345
SMP2				LABEL	345	78901234 67	345
CHKPNT				PRTPARM	345	78901234 67	345
LABEL				LABEL	345	78901234 67	345
PARAM				PRTPARM	345	78901234 67	345
EQUIV				LABEL	345	78901234 67	345
CHKPNT				END	345	78901234 67	345
PARAM							
OSMG2							
SAVE							
CHKPNT							
RBMG2							
SAVE							
CHKPNT							
PRTPARM							
PRTPARM							
SSG3							
SAVE							
CHKPNT							
COND							
MATGPR							
LABEL							
SDR1							
CHKPNT							
SDR2							
OFF							
DPD							
SAVE							
COND	345	78901234 67	345				
CHKPNT							
PARAM							
READ							
SAVE							
CHKPNT							
OFF							
SAVE							
COND							
OFF							
SAVE							
SDR1	345	78901234 67	345				
CHKPNT	345	78901234 67	345				

NORMAL MODES WITH DIFFERENTIAL STIFFNESS

7.14.5 File Name Restart Table

DMAP				DMAP			
Inst.	94	Bit Position		Inst.	94	Bit Position	
		100	110			100	110
BEGIN				CHKPNT		1 3 56	901
FILE				COND			
GP1	4			JUMP			
SAVE	4			LABEL			
CHKPNT	4			COND		2	
GP2	5			GPSP		2	
CHKPNT	5			SAVE		2	
PARAML			1	COND		2	
PURGE			1	OFF		2	
COND			1	LABEL		2	
PLTSET			1	EQUIV			4
SAVE			1	CHKPNT			4
PRTMSG			1	COND		34	
SETVAL			1	MCE1		3	
SAVE			1	CHKPNT		3	
COND				MCE2			4
PLOT				CHKPNT			4
SAVE				LABEL		34	
PPTMSG				EQUIV			5
LABEL				CHKPNT			5
CHKPNT			1	COND			5
GP3	6			SCE1			5
CHKPNT	6			CHKPNT			5
TA1	7			LABEL			5
SAVE	7			EQUIV			6
COND	7 9			CHKPNT			6
PURGE	7	2		COND			6
CHKPNT	7			SMP1			6
PARAM	8			CHKPNT			6
PARAM	9			LABEL			6
EMG			3	RBMG2			7
SAVE			3	CHKPNT			7
CHKPNT			3	SSG1			8
COND	8			CHKPNT			8
EMA	8			EQUIV			9
CHKPNT	8			CHKPNT			9
LABEL	8			COND			9
COND	9			SSG2			9
EMA	9			CHKPNT			9
CHKPNT	9			LABEL			9
COND				SSG3			0
GPWG				SAVE			0
OFF				CHKPNT			0
LABEL	7 9			COND			
EQUIV		0		MATGPR			
CHKPNT		0		MATGPR			
COND		0		LABEL			
SMA3		0		SDR1			1
CHKPNT		0		CHKPNT			1
LABEL		0		SDR2			2
PARAM		1		PARAM			
GP4		1		OFF			
SAVE		1		SAVE			
COND		1		COND			
PARAM		1		PLOT			
PURGE		1 3 56	901	SAVE			

RIGID FORMAT RESTART TABLES

DMAP Inst.	94	Bit Position		120		DMAP Inst.	94	Bit Position		120
		100	110					100	110	
PRTMSG						OFF				8
LABEL						SAVE				8
TA1			3			SDR1				9
DSMG1			3			CHKPNT				9
SAVE			3			CASE				0
CHKPNT			3			SDR2				0
EQUIV			4			OFF				
CHKPNT			4			SAVE				
COND			4			COND				
MCE2			4			PLOT				
CHKPNT			4			SAVE				
LABEL			4			PRTMSG				
EQUIV			5			LABEL				
CHKPNT			5			JUMP				
COND			5			LABEL				
SCE1			5			PRTPARM				
CHKPNT			5			LABEL				
LABEL			5			PRTPARM				
EQUIV			6	2		LABEL				
CHKPNT			6	2		PRTPARM				
COND			6	2		LABEL				
SMP2			6			PRTPARM				
SMP2				2		LABEL				
CHKPNT			6	2		PRTPARM				
LABEL			6	2		LABEL				
PARAM					5	PRTPARM				
EQUIV					5	LABEL				
CHKPNT					5	END				
PARAM					56					
DSMG2					56					
SAVE					56					
CHKPNT					56					
REMG2					7					
SAVE					7					
CHKPNT					7					
PRTPARM										
PRTPARM										
SSG3					8					
SAVE					8					
CHKPNT					8					
COND										
MATGPR										
LABEL										
SDR1					9					
CHKPNT					9					
SDR2					0					
OFF					0					
OPD			7							
SAVE			7							
COND			7							
CHKPNT			7							
PARAM			8							
READ			8							
SAVE			8							
CHKPNT			8							
OFF			8							
SAVE			8							
COND			9							

STATIC ANALYSIS USING CYCLIC TRANSFORMATION

7.15 RESTART TABLES FOR STATIC ANALYSIS USING CYCLIC TRANSFORMATION

7.15.1 Bit Positions for Card Name Restart Table

Card Name	Bit Pos.	Card Name	Bit Pos.	Card Name	Bit Pos.
AXIC	1	CQDMEM3	2	PTRPLT	3
AXIF	1	CQDPLT	2	PTRPLT1	3
CELAS1	1	CQUAD1	2	PTRSHL	3
CELAS2	1	CQUAD2	2	PTUBE	3
CELAS3	1	CQUADTS	2	PTWIST	3
CELAS4	1	CROD	2	GENEL	4
CMASS1	1	CSHEAR	2	CONM1	5
CMASS2	1	CTETRA	2	CONM2	5
CMASS3	1	CTORDRG	2	PELAS	6
CMASS4	1	CTRAPAX	2	PMASS	7
CORD1C	1	CTRAPRG	2	MAT1	8
CORD1R	1	CTRBSC	2	MAT2	8
CORD1S	1	CTRIA1	2	MAT3	8
CORD2C	1	CTRIA2	2	MAT4	8
CORD2R	1	CTRIAAX	2	MAT5	8
CORD2S	1	CTRIARG	2	MATT1	8
GRDSET	1	CTRIATS	2	MATT2	8
GRID	1	CTRIM6	2	MATT3	8
GRIDB	1	CTRMEM	2	MATT4	8
POINTAX	1	CTRPLT	2	MATT5	8
RINGAX	1	CTRPLT1	2	TABLEM1	8
RINGFL	1	CTRSHL	2	TABLEM2	8
SECTAX	1	CTUBE	2	TABLEM3	8
SEQGP	1	CTWIST	2	TABLEM4	8
SPOINT	1	CWEDGE	2	TEMPMTS	8
ADUM1	2	PBAR	3	TEMPMXS	8
ADUM2	2	PCONEAX	3	AXISYMS	9
ADUM3	2	PDUM1	3	CRIGD1	9
ADUM4	2	PDUM2	3	CRIGD2	9
ADUM5	2	PDUM3	3	CRIGD3	9
ADUM6	2	PDUM4	3	CRIGOR	9
ADUM7	2	PDUM5	3	MPC	9
ADUM8	2	PDUM6	3	MPCADD	9
ADUM9	2	PDUM7	3	MPCAX	9
BAROR	2	PDUM8	3	MPCS	9
CBAR	2	PDUM9	3	SPC	10
CCONEAX	2	PHBDY	3	SPC1	10
CDUM1	2	PIHEX	3	SPCADD	10
CDUM2	2	PQDMEM	3	SPCAX	10
CDUM3	2	PQDMEM1	3	SPCS	10
CDUM4	2	PQDMEM2	3	ASET	11
CDUM5	2	PQDMEM3	3	ASET1	11
CDUM6	2	PQDPLT	3	OMIT	11
CDUM7	2	PQUAD1	3	OMIT1	11
CDUM8	2	PQUAD2	3	OMITAX	11
CDUM9	2	PQUADTS	3	CYJOIN	12
CHBDY	2	PROD	3	TEMP	13
CHEXA1	2	PSHEAR	3	TEMPAX	13
CHEXA2	2	PTORDRG	3	TEMPO	13
CIHEX1	2	PTRAPAX	3	TEMPP1	13
CIHEX2	2	PTRBSC	3	TEMPP2	13
CIHEX3	2	PTRIA1	3	TEMPP3	13
CONROD	2	PTRIA2	3	TEMPRB	13
CQDMEM	2	PTRIAAX	3	WTHASS	14
CQDMEM1	2	PTRIATS	3	GROPNT	15
CQDMEM2	2	PTRIM6	3	PLOTET	16
		PTRMEM	3	IRES	17

RIGID FORMAT RESTART TABLES

Card Name Bit Pos.

PLOTS	18
POUTS	19
COUPMASS	24
CPBAR	24
CPQDPLT	24
CPQUAD1	24
CPQUAD2	24
CPROD	24
CPTRBSC	24
CPTRIA1	24
CPTRIA2	24
CPTRPLT	24
CPTUBE	24
DEFORM	59
DEFORMS	59
LOADS	59
RFORCE\$	59
SPCD	59
FORCE	60
FORCE1	60
FORCE2	60
FORCEAX	60
LOAD	60
MOMAX	60
MOMENT	60
MOMENT1	60
MOMENT2	60
PLOAD	60
PLOAD1	60
PLOAD2	60
PLOAD3	60
PRESAX	60
QBDY1	60
QBDY2	60
QHBDY	60
QVECT	60
QVOL	60
SLOAD	60
GRAV	61
RFORCE	61
TEMPLDS	62

STATIC ANALYSIS USING CYCLIC TRANSFORMATION

7.15.2 Bit Positions for File Name Restart Table

<u>File Name</u>	<u>Bit Pos.</u>	<u>File Name</u>	<u>Bit Pos.</u>
BGPD	94	PX	108
CSTM	94	KKK	109
EQEXIN	94	PK	109
GPD	94	PG	110
GPL	94	PL	111
SIL	94	PO	111
ECT	95	PS	111
GPTT	96	QR	111
SLT	96	RUOV	112
GPECT	97	UOV	112
EST	97	PGG	113
GEI	97	QG	113
GPST	98	UGV	113
KGGX	98	OEF1	114
MGG	99	OES1	114
KGG	100	OPG1	114
ASET	101	OQG1	114
RG	101	OUGV1	114
USET	101	PUGV1	114
YS	101	ELSETS	115
OGPST	102	GPSETS	115
GM	103	PLTPAR	115
KNN	104	PLTSETX	115
KFF	105	KDICT	116
KFS	105	KELM	116
KSS	105	MDICT	116
GO	106	MELM	116
KAA	106	LKK	117
KOO	106	RUKV	118
LOO	106	UKV	118
CYCD	107	RUXV	119
GCYCF	108	UXV	119
		ULV	120
		GCYCB	120

RIGID FORMAT RESTART TABLES

7.15.3 Card Name Restart Table

DMAP Inst.	1	10	20	<u>Fit Position</u> 30	40	50	60
BEGIN	1234567890123456789			4			9012
FILE	1234567890123456789			4			9012
FILE	1234567890123456789			4			9012
PARAM	12345678901234						9012
GP1	1						
SAVE	1						
CHKPNT	1						
SSS		6					
GP2	12 45		6				
CHKPNT	12 45		6				
SSS		6					
PARAML			8				
SSS		7					
PURGE			8				
SSS		7					
COND			8				
SSS		7					
PLTSET			8				
SSS		7					
SAVE			8				
SSS		7					
PRTMSG			8				
SSS		7					
PARAM			8				
SSS		7					
PARAM			8				
SSS		7					
COND			8				
SSS		7					
PLOT			8				
SSS		7					
SAVE			8				
SSS		7					
PRTMSG			8				
SSS		7					
LABEL			8				
SSS		7					
CHKPNT			8				
SSS		67					
GP3	12		3				01
SAVE	12		3				01
PARAM	12		3 5				01
CHKPNT	12		3 5				01
SSS		6					
TA1	1234567		3				
SAVE	1234567		3				
PARAM	1234567		3				
COND	1234567		3				
PURGE	1234567		3				
CHKPNT	1234567		3				
SSS		6					
COND	123 5678		45	4			1
PAPAM	123 6 8						
ENG	123 5678		45	4			1
SAVE	123 5678		45	4			1
CHKPNT	123 5678		45	4			1
SSS		6					
COND	123 6 8						

STATIC ANALYSIS USING CYCLIC TRANSFORMATION

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
EMA	123	6 8					
CHKPNT	123	6 8					
SSS		6					
LABEL	123	6 8					
COND	123	5 7 8	45	4			1
EMA	123	5 7 8	45	4			1
CHKPNT	123	5 7 8	45	4			1
SSS		6					
LABEL	123	5 7 8	45	4			1
COND	123	5 7 8	45	4			
SSS		8					
COND	123	5 7 8	45	4			
SSS		8					
GPWG	123	5 7 8	45	4			
SSS		8					
OFF	123	5 7 8	45	4			
SSS		8					
LABEL	123	5 7 8	45	4			1
EQUIV	1234	6 8					
CHKPNT	1234	6 8					
SSS		6					
COND	1234	6 8					
SMA3	1234	6 8					
CHKPNT	1234	6 8					
SSS		6					
LABEL	1234	6 8					
PARAM	1	901					3
GP4	1	901					3
SAVE	1	901					3
COND	1	901					3
PARAM	1	901					3
COND	1	901					3
PURGE	1	901					3
CHKPNT	1234	6 8901					3
SSS		6					
GPCYC	1234	6 89012					3
SAVE	1234	6 89012					3
CHKPNT	1234	6 89012					3
SSS		6					
COND	1234	6 89012					3
COND	123	6 890					
GPSP	123	6 890					
SAVE	123	6 890					
COND	123	6 890					
OFF	123	6 890					
LABEL	123	6 890					
EQUIV	1234	6 89					
CHKPNT	1234	6 89					
SSS		6					
COND	1234	6 89					
MCE1	1	9					
CHKPNT	1	9					
SSS		6					
MCE2	1234	6 89					
CHKPNT	1234	6 89					
SSS		6					
LABEL	1234	6 89					
EQUIV	1234	6 890					

RIGID FORMAT RESTART TABLES

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
CHKPNT	1234	6 890					
SSS		6					
COND	1234	6 890					
SCE1	1234	6 890					
CHKPNT	1234	6 890					
SSS		6					
LABEL	1234	6 890					
EQUIV	1234	6 8901					
CHKPNT	1234	6 8901					
SSS		6					
COND	1234	6 8901					
SMP1	1234	6 8901					
CHKPNT	1234	6 8901					
SSS		6					
LABEL	1234	6 8901					
SSG1	123	5678					9012
CHKPNT	123	5678 .					9012
SSS		6					
EQUIV	123	5678901					9012
CHKPNT	123	5678901					9012
SSS		6					
COND	123	5678901					9012
SSG2	123	5678901					9012
CHKPNT	123	5678901					9012
SSS		6					
COND	123	5678901					9012
SSG3	1234	5678901					9012
CHKPNT	1234	5678901					9012
SSS		6					
COND	1234	5678901	7				9012
MATGPR	1234	5678901	7				9012
LABEL	1234	5678901	7				9012
EQUIV	1234	5678901					9012
COND	1234	5678901					9012
CYCT1	1234	56789012					9012
SAVE	1234	56789012					9012
LABEL	1234	56789012					9012
CHKPNT	1234	56789012					9012
SSS		6					
COND	1234	56789012					9012
PARAM	1234	56789012					9012
JUMP	1234	56789012					9012
LABEL	1234	56789012					9012
CYCT2	1234	56789012					9012
SAVE	1234	56789012					9012
CHKPNT	1234	56789012					9012
SSS		6					
COND	1234	56789012					9012
RBMG2	1234	56789012					9012
CHKPNT	1234	56789012					9012
SSS		6					
SSG3	1234	56789012					9012
CHKPNT	1234	56789012					9012
SSS		6					
CYCT2	1234	56789012					9012
SAVE	1234	56789012					9012
CHKPNT	1234	56789012					9012
SSS		6					

STATIC ANALYSIS USING CYCLIC TRANSFORMATION

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
COND	123456789012						9012
COND	123456789012						9012
MATGPR	123456789012						9012
LABEL	123456789012						9012
PARAM	123456789012						9012
PARAM	123456789012						9012
COND	123456789012						9012
REPT	123456789012						9012
JUMP	123456789012						9012
LABEL	123456789012						9012
EQUIV	123456789012						9012
COND	123456789012						9012
CYCT1	123456789012						9012
SAVE	123456789012						9012
COND	123456789012						9012
LABEL	123456789012						9012
CHKPNT	123456789012						9012
\$\$\$	6						
SDR1	123456789012						9012
CHKPNT	123456789012						9012
\$\$\$	6						
COND			89				
EQMCK			89				
OFF			89				
SAVE			89				
LABEL			89				
SDR2			89				
PARAM			9				
OFF			9				
SAVE			9				
COND			8				
\$\$\$	7						
PLOT			8				
\$\$\$	7						
SAVE			8				
\$\$\$	7						
PRTMSG			8				
\$\$\$	7						
LABEL			8				
\$\$\$	7						
JUMP	12345678901	3456789	4				9012
LABEL	12345678901	3456789	4				9012
\$\$\$	1 3						
PRTPARM	12345678901	3456789	4				9012
\$\$\$	1 3						
LABEL	12345678901	3456789	4				9012
\$\$\$	8						
PRTPARM	12345678901	3456789	4				9012
\$\$\$	8						
LABEL	12345678901	3456789	4				9012
PRTPARM	12345678901	3456789	4				9012
LABEL	12345678901	3456789	4				9012
PRTPARM	12345678901	3456789	4				9012
LABEL	12345678901	3456789	4				9012
PRTPARM	12345678901	3456789	4				9012
LABEL	12345678901	3456789	4				9012
PRTPARM	12345678901	3456789	4				9012
LABEL	12345678901	3456789	4				9012
PRTPARM	12345678901	3456789	4				9012
LABEL	12345678901	3456789	4				9012
END	12345678901	3456789	4				9012

RIGID FORMAT RESTART TABLES

7.15.4 Rigid Format Change Restart Table

DMAP Inst.	63	Bit Position 70	80
BEGIN	3456789012345	7	345
FILE	3456789012345	7	345
FILE	3456789012345	7	345
PARAM			
GP1			
SAVE			
CHKPNT			
GP2			
CHKPNT			
PARAML			
PURGE			
COND			
PLTSET			
SAVE			
PRTMSG			
PARAM			
PARAM			
COND			
PLOT			
SAVE			
PRTMSG			
LABEL			
CHKPNT			
GP3			
SAVE			
PARAM	3456789012345	7	345
CHKPNT			
TA1			
SAVE			
PARAM	3456789012345	7	345
COND	3456789012345	7	345
PURGE			
CHKPNT			
COND			
PARAM			
EMG			
SAVE			
CHKPNT			
COND			
EMA			
CHKPNT			
LABEL			
COND			
EMA			
CHKPNT			
LABEL			
COND			
COND			
GPWG			
OFF			
LABEL			
EQUIV			
CHKPNT			
COND			
SMA3			
CHKPNT			
LABEL			
PARAM			
GP4			

DMAP Inst.	63	Bit Position 70	80
SAVE			
COND	3456789012345	7	345
PARAM			
COND			
PURGE			
CHKPNT			
GPCYC			
SAVE			
CHKPNT			
COND			
COND			
GPSP			
SAVE			
COND			
OFF			
LABEL			
EQUIV			
CHKPNT			
COND			
MCE1			
CHKPNT			
MCE2			
CHKPNT			
LABEL			
EQUIV			
CHKPNT			
COND			
SCE1			
CHKPNT			
LABEL			
EQUIV			
CHKPNT			
COND			
SMP1			
CHKPNT			
LABEL			
SSG1			
CHKPNT			
EQUIV			
CHKPNT			
COND			
SSG2			
CHKPNT			
COND			
SSG3	4		
CHKPNT	4		
COND	3456789012345	7	345
MATGPR	3456789012345	7	345
LABEL	3456789012345	7	345
EQUIV			
COND			
CYCT1			
SAVE			
LABEL			
CHKPNT			
COND			
PARAM			
JUMP			
LABEL			

STATIC ANALYSIS USING CYCLIC TRANSFORMATION

DMAP Inst.	63	<u>Bit Position</u> 70	80
CYCT2			
SAVE			
CHKPNT			
COND			
RBMG2			
CHKPNT			
SSG3			
CHKPNT			
CYCT2			
SAVE			
CHKPNT			
COND			
COND			
MATGPR			
LABEL			
PARAM			
PARAM			
COND			
REPT			
JUMP			
LABEL			
EQUIV			
COND			
CYCT1			
SAVE			
COND			
LABEL			
CHKPNT			
SDR1			
CHKPNT			
COND			
EQMCK			
OFF			
SAVE			
LABEL			
SDP2			
PARAM			
OFF			
SAVE			
COND			
PLOT			
SAVE			
PRTMSG			
LABEL			
JUMP	3456789012345	7	345
LABEL	3456789012345	7	345
PRTPARM	3456789012345	7	345
LABEL	3456789012345	7	345
PRTPARM	3456789012345	7	345
LABEL	3456789012345	7	345
PRTPARM	3456789012345	7	345
LABEL	3456789012345	7	345
PRTPARM	3456789012345	7	345
LABEL	3456789012345	7	345
PRTPARM	3456789012345	7	345
LABEL	3456789012345	7	345
PRTPARM	3456789012345	7	345
LABEL	3456789012345	7	345
PRTPARM	3456789012345	7	345
LABEL	3456789012345	7	345
PRTPARM	3456789012345	7	345
LABEL	3456789012345	7	345
END	3456789012345	7	345

RIGID FORMAT RESTART TABLES

7.15.5 File Name Restart Table

DMAP	Bit Position				DMAP	Bit Position			
Inst.	94	100	110	120	Inst.	94	100	110	120
BEGIN					PARAM		1		
FILE					GP4		1		
FILE					SAVE		1		
PARAM					COND		1		
GP1	4				PARAM		1		
SAVE	4				COND				
CHKPNT	4				PURGE		1 3 56		1
GP2	5				CHKPNT		1 3 56		1
CHKPNT	5				GPCYC			7	
PARAML				5	SAVE			7	
PURGE				5	CHKPNT			7	
COND				5	COND			7	
PLTSET				5	COND		2		
SAVE				5	GPSP		2		
PRTMSG				5	SAVE		2		
PARAM				5	COND		2		
PARAM				5	OFF		2		
COND					LABEL		2		
PLOT					EQUIV			4	
SAVE					CHKPNT			4	
PRTMSG					COND		34		
LABEL					MCE1		3		
CHKPNT				5	CHKPNT		3		
GP3	6				MCE2		4		
SAVE	6				CHKPNT		4		
PARAM	6 9				LABEL		34		
CHKPNT	6				EQUIV			5	
TA1	7				CHKPNT			5	
SAVE	7				COND			5	
PAPAM	7				SCE1			5	
COND	7				CHKPNT			5	
PURGE	78 2				LABEL			5	
CHKPNT	7				EQUIV			6	
COND	89				CHKPNT			6	
PARAM	8				COND			6	
EMG				6	SMP1			6	
SAVE				6	CHKPNT			6	
CHKPNT				6	LABEL			6	
COND	8				SSG1				0
EMA	8				CHKPNT				0
CHKPNT	8				EQUIV				1
LABEL	8				CHKPNT				1
COND	9				COND				1
EMA	9				SSG2				1
CHKPNT	9				CHKPNT				1
LABEL	9				COND				
COND					SSG3				2
COND					CHKPNT				2
GPWG					COND				
OFF					MATGPR				
LABEL	89				LABEL				
EQUIV	0				EQUIV			8	
CHKPNT	0				COND			8	
COND	0				CYCT1			8	
SMA3	0				SAVE			8	
CHKPNT	0				LABEL			8	
LABEL	0				CHKPNT			8	
					COND				

STATIC ANALYSIS USING CYCLIC TRANSFORMATION

DMAP Inst.	94	100	Bit Position 110	120
PARAM				
JUMP				
LABEL				
CYCT2			9	
SAVE			9	
CHKPNT			9	
COND				
RBMG2				7
CHKPNT				7
SSG3				8
CHKPNT				8
CYCT2				9
SAVE				9
CHKPNT				9
COND				
COND				
MATGPR				
LABEL				
PARAM				
PARAM				
COND				
REPT				
JUMP				
LABEL				
EQUIV				0
COND				0
CYCT1				0
SAVE				0
COND				0
LABEL				0
CHKPNT				0
SDR1				
CHKPNT				
COND			4	
EQMCK			4	
OFF			4	
SAVE			4	
LABEL			4	
SDR2			4	
PARAM				
OFF				
SAVE				
COND				
PLOT				
SAVE				
PRTMSG				
LABEL				
JUMP				
LABEL				
PRTPARM				
LABEL				
PRTPARM				
LABEL				
PRTPARM				
LABEL				
PRTPARM				
LABEL				
PRTPARM				
LABEL				
PRTPARM				
LABEL				
END				

NORMAL MODES ANALYSIS USING CYCLIC TRANSFORMATION

7.16 RESTART TABLES FOR NORMAL MODES ANALYSIS USING CYCLIC TRANSFORMATION

7.16.1 Bit Positions for Card Name Restart Table

<u>Card Name</u>	<u>Bit Pos.</u>	<u>Card Name</u>	<u>Bit Pos.</u>	<u>Card Name</u>	<u>Bit Pos.</u>
AXIC	1	CIHEX1	2	PROD	3
AXIF	1	CIHEX2	2	PSHEAR	3
AXSLOT	1	CIHEX3	2	PTORDRG	3
CELAS1	1	CFLUID2	2	PTRAPAX	3
CELAS2	1	CFLUID3	2	PTRBSC	3
CELAS3	1	CFLUID4	2	PTRIA1	3
CELAS4	1	CONROD	2	PTRIA2	3
CMASS1	1	CQDMEM	2	PTRIAAX	3
CMASS2	1	CQDMEM1	2	PTRIATS	3
CMASS3	1	CQDMEM2	2	PTRIM6	3
CMASS4	1	CQDMEM3	2	PTRMEM	3
CORD1C	1	CQDPLT	2	PTRPLT	3
CORD1R	1	CQUAD1	2	PTRPLT1	3
CORD1S	1	CQUAD2	2	PTRSHL	3
CORD2C	1	CQUADTS	2	PTUBE	3
CORD2R	1	CROD	2	PTWIST	3
CORD2S	1	CSHEAR	2	GENEL	4
FREETPT	1	CSLOT3	2	CONM1	5
GRDSET	1	CSLOT4	2	CONM2	5
GRID	1	CTETRA	2	FSLIST	5
GRIDB	1	CTORDRG	2	PELAS	6
GRIDF	1	CTRAPAX	2	PMASS	7
GRIDS	1	CTRAPRG	2	MAT1	8
POINTAX	1	CTRBSC	2	MAT2	8
PPESPT	1	CTRIA1	2	MAT3	8
RINGAX	1	CTRIA2	2	MATT1	8
RINGFL	1	CTRIAAX	2	MATT2	8
SECTAX	1	CTRIARG	2	MATT3	8
SEQGP	1	CTRIATS	2	TABLEM1	8
SLBDY	1	CTRIM6	2	TABLEM2	8
SPOINT	1	CTRMEM	2	TABLEM3	8
ADUM1	2	CTRPLT	2	TABLEM4	8
ADUM2	2	CTRPLT1	2	TEMPMTS	8
ADUM3	2	CTRSHL	2	TEMPMXS	8
ADUM4	2	CTUBE	2	AXISYMS	9
ADUM5	2	CTWIST	2	CRIGD1	9
ADUM6	2	CWEDGE	2	CRIGD2	9
ADUM7	2	PBAR	3	CRIGD3	9
ADUM8	2	PCONEAX	3	CRIGDR	9
ADUM9	2	PDUM1	3	MPC	9
BAROR	2	PDUM2	3	MPCADD	9
CAXIF2	2	PDUM3	3	MPCAX	9
CAXIF3	2	PDUM4	3	MPCS	9
CAXIF4	2	PDUM5	3	SPC	10
CBAR	2	PDUM6	3	SPC1	10
CCONEAX	2	PDUM7	3	SPCADD	10
CDUM1	2	PDUM8	3	SPCAX	10
CDUM2	2	PDUM9	3	SPCS	10
CDUM3	2	PIHEX	3	ASET	11
CDUM4	2	PQDMEM	3	ASET1	11
CDUM5	2	PQDMEM1	3	OMIT	11
CDUM6	2	PQDMEM2	3	OMIT1	11
CDUM7	2	PQDMEM3	3	OMITAX	11
CDUM8	2	PQDPLT	3	CYJOIN	12
CDUM9	2	PQUAD1	3	TEMP	13
CHEXA1	2	PQUAD2	3	TEMPAX	13
CHEXA2	2	PQUADTS	3	TEMPO	13

RIGID FORMAT RESTART TABLES

<u>Card Name</u>	<u>Bit Pos.</u>
TEMPP1	13
TEMPP2	13
TEMPP3	13
TEMPRB	13
WTHASS	14
GRDPNT	15
PLOTEL	16
PLOTS	18
POUTS	19
COUPMASS	24
CPBAR	24
CPQDPLT	24
CPQUAD1	24
CPQUAD2	24
CPROD	24
CPTRBSC	24
CPTRIA1	24
CPTRIA2	24
CPTRPLT	24
CPTUBE	24
EIGR	61
METHODS	62

NORMAL MODES ANALYSIS USING CYCLIC TRANSFORMATION

7.16.2 Bit Positions for File Name Restart Table

<u>File Name</u>	<u>Bit Pos.</u>	<u>File Name</u>	<u>Bit Pos.</u>
BGPD	94	MKK	108
CSTM	94	LAMK	109
EQEXIN	94	PHIK	109
GPD	94	MI	109
GPL	94	OEIGS	109
SIL	94	EED	111
ECT	95	EQDYN	111
GPTT	96	GPLD	111
EST	97	SILD	111
GEI	97	USETD	111
GPECT	97	LAMA	112
GPST	98	PHIA	112
KGGX	98	PHIG	113
MGG	99	QG	113
KGG	100	OEF1	114
ASET	101	OES1	114
RG	101	OPHIG	114
USET	101	OQG1	114
OGPST	102	PPHIG	114
GM	103	BGPD	115
KNN	104	SIP	115
MNN	104	ELSETS	116
KFF	105	GPSETS	116
KFS	105	PLTPAR	116
MFF	105	PLTSETX	116
GO	106	MAA	117
KAA	106	KDICT	118
CYCD	107	KELM	118
KKK	108	MDICT	118
		MELM	118

RIGID FORMAT RESTART TABLES

7.16.3 Card Name Restart Table

DMAP Inst.											<u>Bit Position</u>					
	1								10		20	30	40	50	60	
BEGIN	1234567890123456										89	4				12
GP1	1															
SAVE	1															
CHKPNT	1															
\$\$\$					6											
GP2	12	45									6					
CHKPNT	12	45									6					
\$\$\$					6											
PARAML											8					
\$\$\$					7											
PURGE											8					
\$\$\$					7											
COND											8					
\$\$\$					7											
PLTSET											8					
\$\$\$					7											
SAVE											8					
\$\$\$					7											
PFTMSG											8					
\$\$\$					7											
PARAM											8					
\$\$\$					7											
PARAM											8					
\$\$\$					7											
COND.											8					
\$\$\$					7											
PLOT											8					
\$\$\$					7											
SAVE											8					
\$\$\$					7											
PRTMSG											8					
\$\$\$					7											
LABEL											8					
\$\$\$					7											
CHKPNT											8					
\$\$\$					67											
GP3	1										3					
CHKPNT	1										3					
\$\$\$					6											
TA1	1234567										3					
SAVE	1234567										3					
COND	12345678										34					4
PURGE	1234567										3					
CHKPNT	1234567										3					
\$\$\$					6											
PARAM	123				6						8					
PARAM	123				5						78					
EMG	123				5678						4					4
SAVE	123				5678						4					4
CHKPNT	123				5678						4					4
\$\$\$					6											
COND	123				6						8					
EMA	123				6						8					
CHKPNT	123				6						8					
\$\$\$					6											
LABEL	123				6						8					
COND	123				5						78					4

NORMAL MODES ANALYSIS USING CYCLIC TRANSFORMATION

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
EMA	123 5 78	4	4				
CHKPNT	123 5 78	4	4				
SSS	6						
COND	123 5 78	45	4				
SSS	8						
GPWG	123 5 78	45	4				
SSS	8						
OFP	123 5 78	45	4				
SSS	8						
LABEL	123 5 78	45	4				
SSS	8						
EQUIV	1234 6 8						
CHKPNT	1234 6 8						
SSS	6						
COND	1234 6 8						
SMA3	1234 6 8						
CHKPNT	1234 6 8						
SSS	6						
LABEL	1234 6 8						
PARAM	1	901					
GP4	1	901					
SAVE	1	901					
COND	1	901					
PARAM	1	901					
COND	1	901					
PURGE	1	901					
CHKPNT	12345678901	4	4				
SSS	6						
GPCYC	1	9012					
SAVE	1	9012					
CHKPNT	1	9012					
SSS	6						
COND	1	9012					
COND	123 6 890						
GPSP	123 6 890						
SAVE	123 6 890						
COND	123 6 890						
OFP	123 6 890						
LABEL	123 6 890						
EQUIV	123456789	4	4				
CHKPNT	123456789	4	4				
SSS	6						
COND	123456789	4	4				
MCE1	1	9					
CHKPNT	1	9					
SSS	6						
MCE2	123456789	4	4				
CHKPNT	123456789	4	4				
SSS	6						
LABEL	123456789	4	4				
EQUIV	1234567890	4	4				
CHKPNT	1234567890	4	4				
SSS	6						
COND	1234567890	4	4				
SCE1	1234567890	4	4				
CHKPNT	1234567890	4	4				
SSS	6						
LABEL	1234567890	4	4				

RIGID FORMAT RESTART TABLES

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
EQUIV	1234 6 8901						
EQUIV	12345678901	4		4			
CHKPNT	12345678901	4		4			
\$\$\$	6						
COND	12345678901	4		4			
SMP1	1234 6 8901						
CHKPNT	1234 6 8901						
\$\$\$	6						
SMP2	12345678901	4		4			
CHKPNT	12345678901	4		4			
\$\$\$	6						
LABEL	12345678901	4		4			
DPO	1 901						1
SAVE	1 901						1
COND	1 901						1
CHKPNT	1 901						1
\$\$\$	6						
CYCT2	123456789012						1
SAVE	123456789012						1
CHKPNT	123456789012						1
\$\$\$	6						
COND	123456789012						1
READ	123456789012	4		4			12
SAVE	123456789012	4		4			12
CHKPNT	123456789012	4		4			12
\$\$\$	6						
PARAM	123456789012	4		4			12
OFF	123456789012	4		4			12
SAVE	123456789012	4		4			12
COND	123456789012	4		4			12
OFF	123456789012	4		4			12
SAVE	123456789012	4		4			12
CYCT2	123456789012	4		4			12
SAVE	123456789012	4		4			12
CHKPNT	123456789012	4		4			12
\$\$\$	6						
COND	123456789012	4		4			12
SDR1	123456789012	4		4			12
CHKPNT	123456789012	4		4			12
\$\$\$	6						
PARAM			89				
EQUIV			89				
CHKPNT			89				
\$\$\$	6						
COND			89				
PLTTRAN			89				
SAVE			89				
CHKPNT			89				
\$\$\$	6						
LABEL			89				
COND			89				
EQMCK			89				
OFF			89				
SAVE			89				
LABEL			89				
SDR2			89				
OFF			9				
SAVE			9				

NORMAL MODES ANALYSIS USING CYCLIC TRANSFORMATION

DMAP	Bit Position						
Inst.	1	10	20	30	40	50	60
COND			8				
\$\$\$	7						
PLOT			8				
\$\$\$	7						
SAVE			8				
\$\$\$	7						
PRTMSG			8				
\$\$\$	7						
LABEL			8				
\$\$\$	7						
JUMP	12345678901	3456	89	4			12
LABEL	12345678901	3456	89	4			12
PRTPARM	12345678901	3456	89	4			12
LABEL	12345678901	3456	89	4			12
PRTPARM	12345678901	3456	89	4			12
LABEL	12345678901	3456	89	4			12
PRTPARM	12345678901	3456	89	4			12
LABEL	12345678901	3456	89	4			12
PPTPARM	12345678901	3456	89	4			12
LABEL	12345678901	3456	89	4			12
PRTPARM	12345678901	3456	89	4			12
LABEL	12345678901	3456	89	4			12
PRTPARM	12345678901	3456	89	4			12
LABEL	12345678901	3456	89	4			12
END	12345678901	3456	89	4			12

RIGID FORMAT RESTART TABLES

7.17.4 Rigid Format Change Table

DMAP Inst.	63	Bit Position 70	80		DMAP Inst.	63	Bit Position 70	80
BEGIN		34567890123456	345		SAVE			
GP1					CHKPNT			
SAVE					COND			
CHKPNT					COND			
GP2					GPSP			
CHKPNT					SAVE			
PARAML					COND			
PURGE					OFF			
COND					LABEL			
PLTSET					EQUIV			
SAVE					CHKPNT			
PRMSG					COND			
PARAM					MCE1			
PARAM					CHKPNT			
COND					MCE2			
PLOT					CHKPNT			
SAVE					LABEL			
PRMSG					EQUIV			
LABEL					CHKPNT			
CHKPNT					COND			
GP3					SCE1			
CHKPNT					CHKPNT			
TA1					LABEL			
SAVE					EQUIV			
COND					EQUIV			
PURGE					CHKPNT			
CHKPNT					COND			
PARAM					SMP1			
PARAM	3	678			CHKPNT			
EMG	3	678			SMP2			
SAVE	3	678			CHKPNT			
CHKPNT	3	678			LABEL			
COND					DPD			
EMA					SAVE			
CHKPNT					COND	34567890123456		345
LABEL					CHKPNT			
COND	34567890123456		345		CYCT2			
EMA	3	678			SAVE			
CHKPNT	3	678			CHKPNT			
COND					COND			
GPNG					READ			
OFF					SAVE			
LABEL					CHKPNT			
EQUIV					PARAM			
CHKPNT					OFF			
COND					SAVE			
SMA3					COND			
CHKPNT					OFF			
LABEL					SAVE			
PARAM					CYCT2			
GP4					SAVE			
SAVE					CHKPNT			
COND					COND			
PARAM					SDR1	34567890123456		345
COND					CHKPNT	34567890123456		345
PURGE					PARAM			
CHKPNT					EQUIV			
GPCYC					CHKPNT			

NORMAL MODES ANALYSIS USING CYCLIC TRANSFORMATION

DMAP	<u>Bit Position</u>		
Inst.	63	70	80
COND			
PLTTRAN			
SAVE			
CHKPNT			
LABEL			
COND			
EQMCK			
OFF			
SAVE			
LABEL			
SDR2			
OFF			
SAVE			
COND			
PLOT			
SAVE			
PRTMSG			
LABEL			
JUMP	34567890123456		345
LABEL	34567890123456		345
PRTPARM	34567890123456		345
LABEL	34567890123456		345
PRTPARM	34567890123456		345
LABEL	34567890123456		345
PRTPARM	34567890123456		345
LABEL	34567890123456		345
PRTPARM	34567890123456		345
LABEL	34567890123456		345
PRTPARM	34567890123456		345
LABEL	34567890123456		345
PRTPARM	34567890123456		345
LABEL	34567890123456		345
END	34567890123456		345

RIGID FORMAT RESTART TABLES

7.16.5 File Name Restart Table

DMAP Inst.	94	Bit Position		120
		100	110	
BEGIN				
GP1	4			
SAVE	4			
CHKPNT	4			
GP2	5			
CHKPNT	5			
PARAML				6
PURGE				6
COND				6
PLTSET				6
SAVE				6
PRTMSG				6
PARAM				6
PARAM				6
COND				
PLOT				
SAVE				
PRTMSG				
LABEL				
CHKPNT				6
GP3	6			
CHKPNT	6			
TA1	7			
SAVE	7			
COND	7			
PURGE	7	2		
CHKPNT	7			
PARAM	8			
PARAM	9			
ENG				8
SAVE				8
CHKPNT				8
COND	8			
EMA	8			
CHKPNT	8			
LABEL	8			
COND	9			
EMA	9			
CHKPNT	9			
COND				
GPWG				
OFF				
LABEL				
EQUIV	0			
CHKPNT	0			
COND	0			
SMA3	0			
CHKPNT	0			
LABEL	0			
PARAM	1			
GP4	1			
SAVE	1			
COND	1			
PARAM				
COND				
PURGE		1 3 56		3
CHKPNT		1 3 56		3
GPCYC			7	

DMAP Inst.	94	Bit Position		120
		100	110	
SAVE			7	
CHKPNT			7	
COND			7	
COND		2		
GPSP		2		
SAVE		2		
COND		2		
OFF		2		
LABEL		2		
EQUIV			4	
CHKPNT			4	
COND		34		
MCE1		3		
CHKPNT		3		
MCE2			4	
CHKPNT			4	
LABEL		34		
EQUIV			5	
CHKPNT			5	
COND			5	
SCE1			5	
CHKPNT			5	
LABEL			5	
EQUIV			6	
CHKPNT				7
COND		6		7
SMP1		6		7
CHKPNT		6		
SMP2				7
CHKPNT				7
LABEL		6		7
DPD			1	
SAVE			1	
COND			1	
CHKPNT			1	
CYCT2				
SAVE		8		
CHKPNT		8		
COND		8		
READ			9	
SAVE			9	
CHKPNT			9	
PARAM			9	
OFF			9	
SAVE			9	
COND				34
OFF		9		
SAVE		9		
CYCT2				2
SAVE				2
CHKPNT				2
COND				2
SDR1				3
CHKPNT				3
PARAM				5
EQUIV				5
CHKPNT				5

NORMAL MODES ANALYSIS USING CYCLIC TRANSFORMATION

DMAP		Bit Position		
Inst.	94	100	110	120
COND				5
PLTTRAN				5
SAVE				5
CHKPNT				5
LABEL				5
COND				5
EQMCK				5
OFF				5
SAVE				5
LABEL				5
SDR2				4
OFF				
SAVE				
COND				
PLOT				
SAVE				
PRTHSG				
LABEL				
JUMP				
LABEL				
PPTPARM				
LABEL				
PRTPARM				
LABEL				
PRTPARM				
LABEL				
PRTPARM				
LABEL				
PRTPARM				
LABEL				
PRTPARM				
LABEL				
END				

STATIC HEAT TRANSFER ANALYSIS

7.17 RESTART TABLES FOR STATIC HEAT TRANSFER ANALYSIS

7.17.1 Bit Positions for Card Name Restart Table

<u>Card Name</u>	<u>Bit Pos.</u>	<u>Card Name</u>	<u>Bit Pos.</u>	<u>Card Name</u>	<u>Bit Pos.</u>
AXIC	1	CQDMEM1	2	PTRMEM	3
AXIF	1	CQDMEM2	2	PTRPLT	3
CELAS1	1	CQDMEM3	2	PTUBE	3
CELAS2	1	CQDPLT	2	PTWIST	3
CELAS3	1	CQUAD1	2	GENEL	4
CELAS4	1	CQUAD2	2	CONM1	5
CMASS1	1	CQUADTS	2	CONM2	5
CMASS2	1	CROD	2	PELAS	6
CMASS3	1	CSHEAR	2	PMASS	7
CMASS4	1	CTETRA	2	MAT1	8
CORD1C	1	CTORDRG	2	MAT2	8
CORD1R	1	CTRAPAX	2	MAT3	8
CORD1S	1	CTRAPRG	2	MAT4	8
CORD2C	1	CTRBSC	2	MAT5	8
CORD2R	1	CTRIA1	2	MATT1	8
CORD2S	1	CTRIA2	2	MATT2	8
GRDSET	1	CTRIAAX	2	MATT3	8
GRID	1	CTRIARG	2	MATT4	8
GRIDB	1	CTRIATS	2	MATT5	8
POINTAX	1	CTRMEM	2	TABLEM1	8
RINGAX	1	CTRPLT	2	TABLEM2	8
RINGFL	1	CTUBE	2	TABLEM3	8
SECTAX	1	CTWIST	2	TABLEM4	8
SEQGP	1	CWEDGE	2	TEMPHT\$	8
SPOINT	1	PBAR	3	TEMPHX\$	8
ADUM1	2	PCONEAX	3	AXISYM\$	9
ADUM2	2	PDUM1	3	CRIGD1	9
ADUM3	2	PDUM2	3	CRIGD2	9
ADUM4	2	PDUM3	3	MPC	9
ADUM5	2	PDUM4	3	MPCADD	9
ADUM6	2	PDUM5	3	MPCAX	9
ADUM7	2	PDUM6	3	MPC\$	9
ADUM8	2	PDUM7	3	SPC	10
ADUM9	2	PDUM8	3	SPC1	10
BAROR	2	PDUM9	3	SPCADD	10
CBAR	2	PHBDY	3	SPCAX	10
CCONEAX	2	PIHEX	3	SPC\$	10
CDUM1	2	PQDMEM	3	ASET	11
CDUM2	2	PQDMEM1	3	ASET1	11
CDUM3	2	PQDMEM2	3	OMIT	11
CDUM4	2	PQDMEM3	3	OMIT1	11
CDUM5	2	PQDPLT	3	OMITAX	11
CDUM6	2	PQUAD1	3	SUPAX	12
CDUM7	2	PQUAD2	3	SUPORT	12
CDUM8	2	PQUADTS	3	TEMP	13
CDUM9	2	PROD	3	TEMPAX	13
CHBDY	2	PSHEAR	3	TEMPO	13
CHEXA1	2	PTORDRG	3	TEMPP1	13
CHEXA2	2	PTRAPAX	3	TEMPP2	13
CIHEX1	2	PTRBSC	3	TEMPP3	13
CIHEX2	2	PTRIA1	3	TEMPR8	13
CIHEX3	2	PTRIA2	3	WTMASS	14
CONROD	2	PTRIAAX	3	GRDPNT	15
CQDMEM	2	PTRIATS	3	PLOTEL	16

RIGID FORMAT RESTART TABLES

<u>Card Name</u>	<u>Bit Pos.</u>
------------------	-----------------

IRES	17
PLOTS	18
POUTS	19
LOOPS	22
LOOP1S	23
COUPMASS	24
CPBAR	24
CPQDPLT	24
CPQUAD1	24
CPQUAD2	24
CPROD	24
CPTRBSC	24
CPTRIA1	24
CPTRIA2	24
CPTRPLT	24
CPTUBE	24
DEFORM	59
DEFORMS	59
LOADS	59
RFORCES	59
SPCD	59
FORCE	60
FORCE1	60
FORCE2	60
FORCEAX	60
LOAD	60
MOMAX	60
MOMENT	60
MOMENT1	60
MOMENT2	60
PLOAD	60
PLOAD1	60
PLOAD2	60
PLOAD3	60
PRESAX	60
QBODY1	60
QBODY2	60
QHBDY	60
QVECT	60
QVOL	60
SLOAD	60
GRAV	61
RFORCE	61
TEMPLDS	62

STATIC HEAT TRANSFER ANALYSIS

7.17.2 Bit Positions for File Name Restart Table

<u>File Name</u>	<u>Bit Pos.</u>	<u>File Name</u>	<u>Bit Pos.</u>
BGPD	94	HLL	108
CST	94	HDM	109
HEQEXIN	94	HPG	110
GPD	94	HPL	111
GPL	94	HPO	111
HSIL	94	HPS	111
ECT	95	HQR	111
GPT	96	HRULV	112
HSLT	96	HROV	112
HGPECT	97	HULV	112
HST	97	HROV	112
GPST	98	HPGG	113
HKGGX	98	HOG	113
HKGG	100	HUGV	113
HSET	101	HOEF1	114
RG	101	HOES1	114
HSET	101	HOPG1	114
YS	101	HQGG1	114
OGPST	102	HUGV1	114
GM	103	PUGV1	114
HKNN	104	ELSETS	115
HKFF	105	GPSETS	115
HKFS	105	PLTPAR	115
HGO	106	PLTSETX	115
HKAA	106	HKDICT	116
HKO	106	HKELM	116
HLOO	106	HMDICT	116
HKLL	107	HMELM	116
HKLR	107		
HKRR	107		

RIGID FORMAT RESTART TABLES

7.17.3 Card Name Restart Table

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
BEGIN	1234567890123456789		234				9012
FILE	1234567890123456789		234				9012
FILE	1234567890123456789		234				9012
\$\$\$	1 3						
GP1	1						
SAVE	1						
CHKPNT	1						
\$\$\$		6					
GP2	12 45		6				
CHKPNT	12 45		6				
\$\$\$		6					
PARAML			8				
\$\$\$		7					
PURGE			8				
\$\$\$		7					
COND			8				
\$\$\$		7					
PLTSET			9				
\$\$\$		7					
SAVE			8				
\$\$\$		7					
PRTMSG			8				
\$\$\$		7					
PARAM			8				
\$\$\$		7					
PARAM			8				
\$\$\$		7					
COND			9				
\$\$\$		7					
PLOT			8				
\$\$\$		7					
SAVE			8				
\$\$\$		7					
PRTMSG			8				
\$\$\$		7					
LABEL			8				
\$\$\$		7					
CHKPNT			8				
\$\$\$		67					
GP3	12		3				01
CHKPNT	12		3 5				01
\$\$\$		6					
TA1	1234567		3				
SAVE	1234567		3				
COND	1234567		3				
PURGE	1234567		3				
CHKPNT	1234567		3				
\$\$\$		6					
COND	123 5678		45	4			1
PARAM	123 6 8						
EMG	123 5678		45	4			1
SAVE	123 5678		45	4			1
CHKPNT	123 5678		45	4			1
\$\$\$		6					
COND	123 6 8						
EMA	123 6 8						
CHKPNT	123 6 8						
\$\$\$		6					

STATIC HEAT TRANSFER ANALYSIS

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
LABEL	123	6 8					
PARAM	1	9012	23				
JUMP			23				
SSS	1 3						
LABEL			23				
SSS	1 3						
GP4	1	9012	23				9
SAVE	1	9012	23				9
COND	1	9012	23				9
PARAM	1	9012	23				9
PURGE	1	9012	23				9
CHKPNT	1234	6 89012	23				9
SSS		6					
GPSP	123	6 890	23				
SAVE	123	6 890	23				
COND	123	6 890	23				
OFF	123	6 890	23				
LABEL	123	6 890	23				
EQUIV	1234	6 89	23				
CHKPNT	1234	6 89	23				
SSS		5					
COND	1234	6 89	23				
MCE1	1	9	23				
CHKPNT	1	9	23				
SSS		6					
MCE2	1234	6 89	23				
CHKPNT	1234	6 89	23				
SSS		6					
LABEL	1234	6 89	23				
EQUIV	1234	6 890	23				
CHKPNT	1234	6 890	23				
SSS		6					
COND	1234	6 890	23				
SCE1	1234	6 890	23				
CHKPNT	1234	6 890	23				
SSS		6					
LABEL	1234	6 890	23				
EQUIV	1234	6 8901	23				
CHKPNT	1234	6 8901	23				
SSS		6					
COND	1234	6 8901	23				
SMP1	1234	6 8901	23				
CHKPNT	1234	6 8901	23				
SSS		6					
LABEL	1234	6 8901	23				
EQUIV	1234	6 89012	23				
CHKPNT	1234	6 89012	23				
SSS		6					
COND	1234	6 89012	23				
RBMG1	1234	6 89012	23				
CHKPNT	1234	6 89012	23				
SSS		6					
LABEL	1234	6 89012	23				
RBMG2	1234	6 89012	23				
CHKPNT	1234	6 89012	23				
SSS		6					
COND	1234	6 89012	23				
RBMG3	1234	6 89012	23				

RIGID FORMAT RESTART TABLES

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
CHKPNT	1234	6 89012	23				
SSS		6					
LABEL	1234	6 89012	23				
SSG1	123	5678	23				9012
CHKPNT	123	5678	23				9012
SSS		6					
EQUIV	123	56789012	23				9012
CHKPNT	123	56789012	23				9012
SSS		6					
COND	123	56789012	23				9012
SSG2	123	56789012	23				9012
CHKPNT	123	56789012	23				9012
SSS		6					
LABEL	123	56789012	23				9012
SSG3	1234	56789012	23				9012
SAVE	1234	56789012	23				9012
CHKPNT	1234	56789012	23				9012
SSS		6					
COND	1234	56789012	7 23				9012
MATGPR	1234	56789012	7 23				9012
MATGPR	1234	56789012	7 23				9012
LABEL	1234	56789012	7 23				9012
SDR1	1234	56789012	23				9012
CHKPNT	1234	56789012	23				9012
SSS		6					
COND			23				
SSS	1 3						
REPT			23				
SSS	1 3						
JUMP			23				
SSS	1 3						
PARAM			23				
COND			23				
LABEL			23				
SSS	1 3						
CHKPNT	1234	56789012					9012
SSS		6					
SDR2			89				
PARAM			9				
OFF			9				
SAVE			9				
COND			8				
SSS		7					
PLOT			8				
SSS		7					
SAVE			8				
SSS		7					
PRTMSG			8				
SSS		7					
LABEL			8				
SSS		7					
JUMP	1234	5678901234	56789 234				9012
LABEL	1234	5678901234	56789 234				9012
SSS	1 3						
PRTPARM	1234	5678901234	56789 234				9012
SSS	1 3						
LABEL	1234	5678901234	56789 234				9012
PRTPARM	1234	5678901234	56789 234				9012

STATIC HEAT TRANSFER ANALYSIS

DMAP Inst.	1	10	20	<u>Bit Position</u> 30	40	50	60
LABEL	1234567890	123456789	234				9012
PRTPARM	1234567890	123456789	234				9012
LABEL	1234567890	123456789	234				9012
PRTPARM	1234567890	123456789	234				9012
LABEL	1234567890	123456789	234				9012
END	1234567890	123456789	234				9012

RIGID FORMAT RESTART TABLES

7.17.4 Rigid Format Change Restart Table

DMAP Inst.	63	Bit Position 70	80	DMAP Inst.	63	Bit Position 70	80
BEGIN		345678901234567	45	MCE2			
FILE		345678901234567	45	CHKPNT			
FILE		345678901234567	45	LABEL			
GP1				EQUIV			
SAVE				CHKPNT			
CHKPNT				COND			
GP2				SCE1			
CHKPNT				CHKPNT			
PARAML				LABEL			
PURGE				EQUIV			
COND				CHKPNT			
PLTSET				COND			
SAVE				SMP1			
PRTMSG				CHKPNT			
PARAM				LABEL			
PARAM				EQUIV			
COND				CHKPNT			
PLOT				COND			
SAVE				RBMG1			
PRTMSG				CHKPNT			
LABEL				LABEL			
CHKPNT				RBMG2			
GP3				CHKPNT			
CHKPNT				COND			
TA1				RBMG3			
SAVE				CHKPNT			
COND		345678901234567	45	LABEL			
PURGE				SSG1			
CHKPNT				CHKPNT			
COND				EQUIV			
PARAM				CHKPNT			
EMG				COND			
SAVE				SSG2			
CHKPNT				CHKPNT			
COND				LABEL			
EMA				SSG3		4	
CHKPNT				SAVE		4	
LABEL				CHKPNT		4	
PARAM				COND		345678901234567	45
JUMP				MATGPR		345678901234567	45
LABEL				MATGPR		345678901234567	45
GP4				LABEL		345678901234567	45
SAVE				SDR1		345678901234567	45
COND		345678901234567	45	CHKPNT		345678901234567	45
PARAM				COND		345678901234567	45
PURGE				REPT		345678901234567	45
CHKPNT				JUMP		345678901234567	45
GPSP				PARAM		345678901234567	45
SAVE				COND		345678901234567	45
COND				LABEL		345678901234567	45
OFF				CHKPNT		345678901234567	45
LABEL				SDR2			
EQUIV				PARAM			
CHKPNT				OFF			
COND				SAVE			
MCE1				COND			
CHKPNT				PLOT			

STATIC HEAT TRANSFER ANALYSIS

DMAP		Bit Position	
Inst.	63	70	80

SAVE		
PRTMSG		
LABEL		
JUMP	345678901234567	45
LABEL	345678901234567	45
PRTPARM	345678901234567	45
LABEL	345678901234567	45
PRTPARM	345678901234567	45
LABEL	345678901234567	45
PRTPARM	345678901234567	45
LABEL	345678901234567	45
PRTPARM	345678901234567	45
LABEL	345678901234567	45
END	345678901234567	45

RIGID FORMAT RESTART TABLES

7.17.5 File Name Restart Table

DMAP		Bit Position		DMAP		Bit Position	
Inst.	94	100	110	Inst.	94	100	110
BEGIN				MCE1		3	
FILE				CHKPNT		3	
FILE				MCE2		4	
GP1	4			CHKPNT		4	
SAVE	4			LABEL		34	
CHKPNT	4			EQUIV		5	
GP2	5			CHKPNT		5	
CHKPNT	5			COND		5	
PARAML			5	SCE1		5	
PURGE			5	CHKPNT		5	
COND			5	LABEL		5	
PLTSET			5	EQUIV		6	
SAVE			5	CHKPNT		6	
PRTMSG			5	COND		6	
PARAM			5	SMP1		6	
PARAM			5	CHKPNT		6	
COND				LABEL		6	
PLOT				EQUIV		7	
SAVE				CHKPNT		7	
PRTMSG				COND		7	
LABEL				RBMG1		7	
CHKPNT			5	CHKPNT		7	
GP3	6			LABEL		7	
CHKPNT	6			RBMG2		8	
TA1	7			CHKPNT		8	
SAVE	7			COND		9	
COND	7			RBMG3		9	
PURGE	78	2		CHKPNT		9	
CHKPNT	7			LABEL		9	
COND	89			SSG1		0	
PARAM	8			CHKPNT		0	
EMG			6	EQUIV		1	
SAVE			6	CHKPNT		1	
CHKPNT			6	COND		1	
COND	8			SSG2		1	
EMA	8			CHKPNT		1	
CHKPNT	8			LABEL		1	
LABEL	8			SSG3		2	
PARAM		1		SAVE		2	
JUMP		1		CHKPNT		2	
LABEL	0			COND			
GP4		1		MATGPR			
SAVE		1		MATGPR			
COND		1		LABEL			
PARAM		1		SDR1		3	
PURGE		1 3 567 9 1		CHKPNT		3	
CHKPNT		1 3 567 9 1		COND			
GPSP		2		REPT			
SAVE		2		JUMP			
COND		2		PARAM			
OFF		2		COND			
LABEL		2		LABEL			
EQUIV		4		CHKPNT		3	
CHKPNT		4		SDR2		4	
COND		34		PARAM			

STATIC HEAT TRANSFER ANALYSIS

		<u>Bit Position</u>		
DMAP	Inst.	100	110	120
94				
OFF				
SAVE				
COND				
PLOT				
SAVE				
PRTMSG				
LABEL				
JUMP				
LABEL				
PRTPARM				
LABEL				
PRTPARM				
LABEL				
PRTPARM				
LABEL				
PRTPARM				
LABEL				
END				

NONLINEAR STATIC HEAT TRANSFER ANALYSIS

7.18 RESTART TABLES FOR NONLINEAR STATIC HEAT TRANSFER ANALYSIS

7.18.1 Bit Positions for Card Name Restart Table

Card Name	Bit Pos.	Card Name	Bit Pos.	Card Name	Bit Pos.
CELAS1	1	.CTRIATS	2	.SPCADD	10
CELAS2	1	.CTRMEM	2	.SPCS	10
CELAS3	1	CTUBE	2	TEMP	13
CELAS4	1	.CWEDGE	2	TEMPPD	13
CORD1C	1	PBAR	3	TEMPP1	13
CORD1R	1	.PDUM1	3	TEMPP2	13
CORD1S	1	.PDUM2	3	TEMPP3	13
CORD2C	1	.PDUM3	3	TEMPPB	13
CORD2R	1	PDUM4	3	PLOTEL	16
CORD2S	1	PDUM5	3	HIRES	17
GROSET	1	PDUM6	3	.PLOTS	18
GRID	1	PDUM7	3	POUTS	19
SEJGP	1	.PDUM8	3	EPSHT	54
SPOINT	1	PDUM9	3	MAXIT	54
ADUM1	2	.PHBDY	3	.RADMTX	55
ADUM2	2	PIHEX	3	.RADLST	55
ADUM3	2	PQDMEM	3	SIGMA	55
ADUM4	2	PQDMEM1	3	TABS	55
ADUM5	2	PQDMEM2	3	LOADS	59
ADUM6	2	.PQDMEM3	3	SPCD	59
ADUM7	2	PQUAD1	3	LOAD	60
ADUM8	2	PQUAD2	3	QBDY1	60
ADUM9	2	PQUADTS	3	QBDY2	60
BAROR	2	PRQD	3	.QHBDY	60
CBAR	2	PTRAPAX	3	.QVECT	60
CDUM1	2	PTRBSC	3	QVJL	60
CDUM2	2	PTRIA1	3	.SLJAD	60
CDUM3	2	PTRIA2	3	TEMPLDS	62
CDUM4	2	.PTRIAAX	3		
CDUM5	2	PTRIATS	3		
CDUM6	2	PTRMEM	3		
CDUM7	2	PTUBE	3		
CDUM8	2	PELAS	6		
CDUM9	2	MAT1	8		
CHBDY	2	MAT2	8		
CHEXA1	2	MAT3	8		
CHEXA2	2	MAT4	8		
CIHEX1	2	MAT5	8		
CIHEX2	2	MATT1	8		
CIHEX3	2	MATT2	8		
.CONROD	2	MATT3	8		
CQDMEM	2	MATT4	8		
CQDMEM1	2	MATT5	8		
CQDMEM2	2	TABLEM1	8		
CQDMEM3	2	TABLEM2	8		
CQUAD1	2	TABLEM3	8		
.CQUAD2	2	TABLEM4	8		
.CQUADTS	2	TEMPMTS	8		
CRQD	2	TEMPMXS	8		
CTETRA	2	CRIGD1	9		
.CTRAPAX	2	CRIGD2	9		
.CTRAPRG	2	MPC	9		
.CTRIA1	2	MPCADD	9		
CTRIA2	2	MPCS	9		
CTRIAAX	2	SPC	10		
.CTRIARG	2	SPC1	10		

RIGID FORMAT RESTART TABLES

7.18.2 Bit Positions for File Name Restart Table

<u>File Name</u>	<u>Bit Pos.</u>
------------------	-----------------

BGPD	94
CSTH	94
HEQEXIN	94
GPD	94
GPL	94
HSIL	94
ECT	95
GPTT	96
HSLT	96
HST	97
HGEI	97
HGPCT	97
GPECT	97
HKDICT	98
HKELM	98
GPST	99
HKGGX	99
HRGG	100
HKGG	100
HQGE	100
HASET	101
RG	101
HUSET	101
YS	101
HOGPST	102
GM	103
HKNN	104
HRNN	104
HKFF	105
HKFS	105
HKSF	105
HKSS	105
HRFN	107
HRSN	107
HLLL	108
HULL	108
HPG	110
HPF	111
HPS	111
HRULV	112
HULV	112
HPGG	113
HQG	112
HUGV	112
HBGPD	113
HSIP	113
HOEF1	114
HOES1	114
HOPG1	114
HOQG1	114
HOUGV1	114
HPUGV1	114
ELSETS	115
GPSETS	115
PLTPAR	115
PLTSETX	115
VFS	116
HOEFIX	117

NONLINEAR STATIC HEAT TRANSFER ANALYSIS

7.18.3 Card Name Restart Table

DMAP Inst.	Bit Position									
	1	10	20	30	40	50	60			
BEGIN	123	6 8 9 0	3 6 7 8 9							
GP1	1							4 5	9 0	2
SAVE	1									
CHKPNT	1									
\$\$\$		6								
GP2	12		6							
CHKPNT	12		6							
\$\$\$		6								
PARAML			8							
\$\$\$		7								
PURGE			8							
\$\$\$		7								
COND			8							
\$\$\$		7								
PLTSET			8							
\$\$\$		7								
SAVE			8							
\$\$\$		7								
PRTMSG			8							
\$\$\$		7								
PARAM			8							
\$\$\$		7								
PARAM			8							
\$\$\$		7								
COND			8							
\$\$\$		7								
PLOT			8							
\$\$\$		7								
SAVE			8							
\$\$\$		7								
PRTMSG			8							
\$\$\$		7								
LABEL			8							
\$\$\$		7								
CHKPNT			8							
\$\$\$		6 7								
GP3	12		3							
CHKPNT	12		3							
\$\$\$		6								
TA1	123	6								
SAVE	123	6								
COND	123	6 8								
CHKPNT	123	6								
\$\$\$		6								
PARAM	123	6 8								
EMG	123	6 8								
SAVE	123	6 8								
CHKPNT	123	6 8								
\$\$\$		6								
COND	123	6 8								
EMA	123	6 8								
CHKPNT	123	6 8								
\$\$\$		6								
LABEL	123	6 8								
RMG	123	6 8								
SAVE	123	6 8								
EQUIV	123	6 8								
PURGE	123	6 8								

RIGID FORMAT RESTART TABLES

DMAP Inst.	<u>Bit Position</u>									
	1	10	20	30	40	50	60			
CHKPNT	123	6 8					5			
SSS		6								
GP4	1	90						9		
SAVE	1	90						9		
COND	1	90						9		
PURGE	1	90						9		
CHKPNT	123	6 8 9						9		
SSS		6								
GPSP	123	6 8 90								
SAVE	123	6 8 90								
COND	123	6 8 90								
DFP	123	6 8 90								
LABEL	123	6 8 90								
EQUIV	123	6 8 9	3							
CHKPNT	123	6 8 9	3							
SSS		6								
COND	123	6 8 9	3							
MCE1	1	9								
CHKPNT	1	9								
SSS		6								
MCE2	123	6 8 9				5				
CHKPNT	123	6 8 9				5				
SSS		6								
LABEL	123	6 8 9				5				
EQUIV	123	6 8 90				5				
CHKPNT	123	6 8 90				5				
SSS		6								
COND	123	6 8 90				5				
VEC	123	6 8 90				5				
PARTN	123	6 8 90				5				
PARTN	123	6 8 90				5				
LABEL	123	6 8 90				5				
CHKPNT	123	6 8 90				5				
SSS		6								
DECOMP	123	6 8 90				5				
SAVE	123	6 8 90				5				
COND	123	6 8 90				5				
CHKPNT	123	6 8 90				5				
SSS		6								
SSG1	123	6 8	3			5	90	2		
CHKPNT	123	6 8	3			5	90	2		
SSS		6								
EQUIV	123	6 8 90	3			5	90	2		
COND	123	6 8 90	3			5	90	2		
SSG2	123	6 8 90	3			5	90	2		
LABEL	123	6 8 90	3			5	90	2		
CHKPNT	123	6 8 90	3			5	90	2		
SSS		6								
SSGHT	123	6 8 90	3	7		45	90	2		
CHKPNT	123	6 8 90	3			45	90	2		
SSS		6								
COND	123	6 8 90	3	7		45	90	2		
MATGPR	123	6 8 90	3	7		45	90	2		
LABEL	123	6 8 90	3	7		45	90	2		
PLTTRAN	1									
SAVE	1									
CHKPNT	1									

NONLINEAR STATIC HEAT TRANSFER ANALYSIS

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
\$\$\$		6					
SDR2			89				
PARAM			9				
QFP			9				
SAVE			9				
SDRHT			89				
QFP			9				
SAVE			9				
COND			8				
\$\$\$		7					
PLOT			8				
\$\$\$		7					
PRTMSG			8				
\$\$\$		7					
LABEL			8				
\$\$\$		7					
JUMP	123	5 890	3 6789				
LABEL	123	6 890	3 6789			45	90 2
PRTPARM	123	6 890	3 6789			45	90 2
LABEL	123	6 890	3 6789			45	90 2
PRTPARM	123	6 890	3 6789			45	90 2
LABEL	123	6 890	3 6789			45	90 2
PRTPARM	123	6 890	3 6789			45	90 2
LABEL	123	6 890	3 6789			45	90 2
END	123	6 890	3 6789			45	90 2

RIGID FORMAT RESTART TABLES

7.18.4 Rigid Format Change Restart Table

DMAP Inst.	63	Bit Position 70	80	DMAP Inst.	63	Bit Position 70	80
BEGIN		345678901234567	3 5	CHKPNT			
GP1				COND			
SAVE				VEC			
CHKPNT				PARTN			
GP2				PARTN			
CHKPNT				LABEL			
PARAML				CHKPNT			
PURGE				DECOMP			
COND				SAVE			
PLTSET				COND			
SAVE				CHKPNT			
PRTMSG				SSG1			
PARAM				CHKPNT			
PARAM				EQUIV			
COND				COND			
PLOT				SSG2			
SAVE				LABEL			
PRTMSG				CHKPNT			
LABEL				SSGHT			
CHKPNT				CHKPNT			
GP3				COND			
CHKPNT				MATGPR			
TAL				LABEL			
SAVE				PLTTRAN			
COND		345678901234567	3 5	SAVE			
CHKPNT				CHKPNT			
PARAM				SDR2			
EMG				PARAM			
SAVE				QFP			
CHKPNT				SAVE			
COND				SDRHT			
EMA				QFP			
CHKPNT				SAVE			
LABEL				COND			
RMG				PLOT			
SAVE				PPTMSG			
EQUIV				LABEL		345678901234567	3 5
PURGE				JUMP		345678901234567	3 5
CHKPNT				LABEL		345678901234567	3 5
GP4				PRTPARM		345678901234567	3 5
SAVE				LABEL		345678901234567	3 5
COND		345678901234567	3 5	PRTPARM		345678901234567	3 5
PURGE				LABEL		345678901234567	3 5
CHKPNT				PRTPARM		345678901234567	3 5
GPSP				LABEL		345678901234567	3 5
SAVE				END			
COND							
QFP							
LABEL							
EQUIV							
CHKPNT							
COND							
MCE1							
CHKPNT							
MCE2							
CHKPNT							
LABEL							
EQUIV							

NONLINEAR STATIC HEAT TRANSFER ANALYSIS

7.18.5 File Name Restart Table

DMAP	Bit Position			
Inst.	94	100	110	120
BEGIN				
GP1	4			
SAVE	4			
CHKPNT	4			
GP2	5			
CHKPNT	5			
PARAML				5
PURGE				5
COND				5
PLTSET				5
SAVE				5
PRTMSG				5
PARAM				5
PARAM				5
COND				5
PLOT				
SAVE				
PRTMSG				
LABEL				
CHKPNT				5
GP3	6			
CHKPNT	6			
TA1	7			
SAVE	7			
COND	7			
CHKPNT	7			
PARAM	9			
EMG	8			
SAVE	8			
CHKPNT	8			
COND	9			
EPA	9			
CHKPNT	9			
LABEL	9			
RMG	0			
SAVE	0			
EQUIV	0			
PURGE	0			
CHKPNT	0			
GP4	1			
SAVE	1			
COND	1			
PURGE	1	3	5	7
CHKPNT	1	3	5	7
GPSP	2			
SAVE	2			
COND	2			
OFF	2			
LABEL	2			
EQUIV	4			
CHKPNT	4			
COND	34			
MCE1	3			
CHKPNT	3			
MCE2	4			
CHKPNT	4			
LABEL	34			

RIGID FORMAT RESTART TABLES

DMAP Inst.	94	Bit Position		120
		100	110	
EQUIV		5 7		6
CHKPNT		5 7		6
COND		5 7		6
VEC				6
PARTN		5		
PARTN		7		
LABEL		5 7		
CHKPNT		5 7		6
DECOMP		8		
SAVE		8		
COND		8		
CHKPNT		8		
SSG1			0	
CHKPNT			0	
EQUIV			1	
COND			1	
SSG2			1	
LABEL			1	
CHKPNT			1	
SSGHT			2	
CHKPNT			2	
COND				
MATGPR				
LABEL				
PLTTRAN			3	
SAVE			3	
CHKPNT			3	
SDR2			4	
PARAM				
DFP				
SAVE				
SDRHT				7
DFP				
SAVE				
COND				
PLGT				
PPTMSG				
LABEL				
JUMP				
LABEL				
PRTPARM				
LABEL				
PRTPARM				
LABEL				
PRTPARM				
LABEL				
END				

TRANSIENT HEAT TRANSFER ANALYSIS

7.19 RESTART TABLES FOR TRANSIENT HEAT TRANSFER ANALYSIS

7.19.1 Bit Positions for Card Name Restart Table

Card Name	Bit Pos.	Card Name	Bit Pos.	Card Name	Bit Pos.
CDAMP1	1	CTRIA1	2	MPCS	9
CDAMP2	1	CTRIA2	2	SPC	10
CDAMP3	1	CTRIAAX	2	SPC1	10
CDAMP4	1	CTRIARG	2	SPCADD	10
CELAS1	1	CTRIATS	2	SPCS	10
CELAS2	1	CTRMEM	2	ASET	11
CELAS3	1	CTUBE	2	ASET1	11
CELAS4	1	CWEDGE	2	OMIT	11
CORD1C	1	PBAR	3	OMIT1	11
CORD1R	1	PDUM1	3	TEMP	13
CORD1S	1	PDUM2	3	TEMPD	13
CORD2C	1	PDUM3	3	TEMPP1	13
CORD2R	1	PDUM4	3	TEMPP2	13
CORD2S	1	PDUM5	3	TEMPP3	13
GRDSET	1	PDUM6	3	TEMPRE	13
GRID	1	PDUM7	3	PLOTEL	16
SEQGP	1	PDUM8	3	PLOTS	18
SPOINT	1	PDUM9	3	POUTS	19
ADUM1	2	PHSDY	3	XYOUTS	20
ADUM2	2	PIHEX	3	AOUTS	21
ADUM3	2	PQDMEM	3	LOOPS	22
ADUM4	2	PQDMEM1	3	LOOP1S	23
ADUM5	2	PQDMEM2	3	NOLOPS	25
ADUM6	2	PQDMEM3	3	AXYOUTS	27
ADUM7	2	PQUAD1	3	RADMTX	55
ADUM8	2	PQUAD2	3	RADLST	55
ADUM9	2	PQUADTS	3	SIGMA	55
BAROR	2	PROD	3	TABS	55
CBAR	2	PTRAPAX	3	TREF	55
CDUM1	2	PTRIA1	3	QB DY1	56
CDUM2	2	PTRIA2	3	QB DY2	56
CDUM3	2	PTRIAAX	3	OJECT	56
CDUM4	2	PTRIATS	3	QHBDY	56
CDUM5	2	PTRMEM	3	QVOL	56
CDUM6	2	PTUBE	3	LOAD	56
CDUM7	2	PELAS	6	SLOAD	56
CDUM8	2	MAT1	8	EPOINT	57
CDUM9	2	MAT2	8	SEQEP	57
CHBDY	2	MAT3	8	TF	57
CHEXA1	2	MAT4	8	PDAMP	59
CHEXA2	2	MAT5	8	DMIG	60
CIHEX1	2	MATT1	8	B2PPS	60
CIHEX2	2	MATT2	8	K2PPS	60
CIHEX3	2	MATT3	8	TFS	60
CONROD	2	MATT4	8	DAREA	61
CQDMEM	2	MATT5	8	DELAY	61
CQDMEM1	2	TABLEM1	8	DLOAD	61
CQDMEM2	2	TABLEM2	8	DLOADS	61
CQDMEM3	2	TABLEM3	8	TABLED1	61
CQUAD1	2	TABLEM4	8	TABLED2	61
CQUAD2	2	TEMPMTS	8	TABLED3	61
CQUADTS	2	TEMPMXS	8	TABLED4	61
CROD	2	CRIGD1	9	TSTEPS	61
CTETRA	2	CRIGD2	9	TLOAD1	61
CTRAPRG	2	MPC	9	TLOAD2	61
CTRAPAX	2	MPCADD	9	TSTEP	61

RIGID FORMAT RESTART TABLES

<u>Card Name</u>	<u>Bit Pos.</u>
BETA	62
ICS	62
NLFORCE	62
NOLIN1	62
NOLIN2	62
NOLIN3	62
NOLIN4	62
RADLIN	62
TIC	62

TRANSIENT HEAT TRANSFER ANALYSIS

7.19.2 Bit Positions for File Name Restart Table

<u>File Name</u>	<u>Bit Pos.</u>	<u>File Name</u>	<u>Bit Pos.</u>
BGPD	94	HRDD	110
CSTM	94	HPDT	111
HEQEXIN	94	HPNLD	111
GPD	94	HUDVT	111
GPL	94	HOUDV1	112
HSIL	94	HOPNL1	112
ECT	95	HOUDV2	113
GPTT	96	HOPNL2	113
HSLT	96	HQP	114
HST	97	HUPV	114
HGPECT	97	HOEF1	115
GPST	98	HOPP1	115
HKGGX	98	HOQP1	115
HGG	99	HOUPV1	115
HKGG	100	HPUGV	115
HRGG	100	HOEF2	116
HQGE	100	HOPP2	116
RG	101	HOQP2	116
ASET	101	HOUPV2	116
HUSET	101	HPDO	117
OGPST	102	HPDT	117
GM	103	HPPO	117
HBNN	104	HPSO	117
HKNN	104	HTOL	117
HRNN	104	HGPD	118
HBBF	105	HSIP	118
HKFF	105	ELSETS	120
HKFS	105	GPSETS	120
HRFF	105	PLTPAR	120
HGO	106	PLTSETX	120
HKAA	106	HRAA	121
HDLT	107	HBA	122
HEQDYN	107	HKDICT	123
GPLD	107	HKELM	123
HNLFT	107	HBDICT	123
HSILO	107	HBELM	123
TFPOOL	107		
HTRL	107		
HUSETD	107		
HCASEXX	108		
HB2PP	109		
HK2PP	109		
HB2DD	110		
HBD	110		
GMD	110		
HGD	110		
HK2DD	110		
HKDD	110		

RIGID FORMAT RESTART TABLES

7.19-3 Card Name Restart Table

DMAP Inst.	Bit Position					
	1	10	20	30	40	50 60
BEGIN	123	6 890123	6 890123	5		567 9012
FILE	123	6 890123	6 890123	5		567 9012
GP1	1					
SAVE	1					
PURGE	1					
CHKPNT	1					
\$\$\$		6				
COND	1					
GP2	12		6			
CHKPNT	12		6			
\$\$\$		6				
PARAML			8			
\$\$\$		7				
PURGE			8			
\$\$\$		7				
COND			8			
\$\$\$		7				
PLTSET			8			
\$\$\$		7				
SAVE			8			
\$\$\$		7				
PRTMSG			8			
\$\$\$		7				
PARAM			8			
\$\$\$		7				
PARAM			8			
\$\$\$		7				
COND			8			
\$\$\$		7				
PLOT			8			
\$\$\$		7				
SAVE			8			
\$\$\$		7				
PRTMSG			8			
\$\$\$		7				
LABEL			8			
\$\$\$		7				
CHKPNT			8			
\$\$\$		67				
GP3	1		3		6	
CHKPNT	1		3		6	
\$\$\$		6				
TA1	123	6				9
SAVE	123	6				9
CHKPNT	123	6				9
\$\$\$		6				
COND	123	6 8	3			
PARAM	123	6 8				
PARAM	123	8				9
ENG	123	6 8				9
SAVE	123	6 8				9
CHKPNT	123	6 8				9
\$\$\$		6				
COND	123	6 8				
EMA	123	6 8				
CHKPNT	123	6 8				
\$\$\$		6				

TRANSIENT HEAT TRANSFER ANALYSIS

DMAP Inst.	Bit Position						
	1	10	20	30	40	50	60
LABEL	123	6 8					
COND	123	8					3
EMA	123	8					9
CHKPNT	123	8					9
\$\$\$		6					
LABEL	123	8					3
PURGE	123	8					9
CHKPNT	123	8					9
\$\$\$		6					
LABEL	123	6 8					
RMG	123	6 8				5	
SAVE	123	6 8				5	
EQUIV	123	6 8				5	
PURGE	123	6 8				5	
CHKPNT	123	6 8				5	
\$\$\$		6					
GP4	1	901					
SAVE	1	901					
PURGE	1	901					
CHKPNT	123	6 89					
\$\$\$		6					
COND	123	6 890					
GPSP	123	6 890					
SAVE	123	6 890					
COND	123	6 890					
OFF	123	6 890					
LABEL	123	6 890					
EQUIV	123	6 89					
CHKPNT	123	6 89					
\$\$\$		6					
COND	123	6 89				5	9
MCE1	1	9					
CHKPNT	1	9					
\$\$\$		6					
MCE2	123	6 89				5	9
CHKPNT	123	6 89				5	9
\$\$\$		6					
LABEL	123	6 89				5	9
EQUIV	123	6 890				5	9
CHKPNT	123	6 890				5	9
\$\$\$		6					
COND	123	6 890				5	9
SCE1	123	6 890				5	9
CHKPNT	123	6 890				5	9
\$\$\$		6					
LABEL	123	6 890				5	9
EQUIV	123	6 8901					
EQUIV	123	89				5	
EQUIV	123	89					9
CHKPNT	123	6 8901				5	9
\$\$\$		6					
COND	123	6 8901				5	9
SMP1	123	6 8901				5	
CHKPNT	123	6 8901				5	
\$\$\$		6					
COND	123	6 8901				5	
SMP2	123	6 8901				5	
CHKPNT	123	6 8901				5	
\$\$\$		6					

RIGID FORMAT RESTART TABLES

DMAP Inst.	Bit Position									
	1	10	20	30	40	50	60			
LABEL	123	6 8901					5			
COND	123	6 8901						9		
SMP2	123	6 8901						9		
CHKPNT	123	6 8901						9		
\$\$\$		6								
LABEL	123	6 8901					5	9		
DPD	1	901					7	012		
SAVE	1	901					7	1		
COND	1						7	1		
EQUIV	1						7	1		
PURGE	1						7	1		
CHKPNT	1	901					7	01		
\$\$\$		6								
MTRXIN	1						7	0		
SAVE	1						7	0		
PARAM	1						7	0		
PURGE	123	6 8901					7	90		
EQUIV	123	6 8901					7	90		
CHKPNT	123	6 8901					7	90		
\$\$\$		6								
COND	123	6 8901					7	90		
GKAD	123	6 8901					7	90		
LABEL	123	6 8901					7	90		
EQUIV	123	6 8901					7	90		
CHKPNT	123	6 8901					7	90		
\$\$\$		6								
TRLG	123	6 8901					5	7	1	
SAVE	123	6 8901					5	7	1	
EQUIV	123	6 8901					5	7	1	
EQUIV	123	6 8901					5	7	1	
CHKPNT	123	6 8901					5	7	1	
\$\$\$		6								
TRHT	123	6 8901 3					567	9012		
CHKPNT	123	6 8901 3	23				567	9012		
\$\$\$		6								
VDR		3	901	7			567	9012		
SAVE		3	901	7			567	9012		
CHKPNT		3	1	7			567	9012		
\$\$\$		6								
COND		3	1	7			567	9012		
SDR3		3	1	7			567	9012		
PARAM			9 1							
OFF		3	1				567	9012		
SAVE		3	1				567	9012		
CHKPNT		3	1	7			567	9012		
\$\$\$		6								
XYTRAN										
\$\$\$		7								
SAVE										
\$\$\$		7								
XYPLOT										
\$\$\$		7								
LABEL			1							
PARAM	123	6 8901	23				7	9012		
COND	123	6 8901	23				7	9012		
EQUIV	123	6 8901	23				7	9012		
COND	123	6 8901	23				7	9012		
SDR1	123	6 8901	23				7	9012		

TRANSIENT HEAT TRANSFER ANALYSIS

DMAP Inst.	Bit Position						
	1	10	20	30	40	50	60
LABEL	123	6 8901					
CHKPNT	123	6 8901	23				7 9012
\$\$\$		6	23				7 9012
PLTTRAN	1						
SAVE	1						
SDR2			890				
SDRHT			890				
EQUIV			890				
SDR3			890				
CHKPNT			890				
\$\$\$		6					
OFF			9				
SAVE			9				
COND			8				
\$\$\$		7					
PLOT			8				
\$\$\$		7					
SAVE			8				
\$\$\$		7					
PRTMSG			8				
\$\$\$		7					
LABEL			8				
\$\$\$		7					
XYTRAN			0				
\$\$\$		7					
SAVE			0				
\$\$\$		7					
XYPLOT			0				
\$\$\$		7					
LABEL			0				
JUMP	123	6 890123	6 890123	5			
LABEL	123	6 890123	6 890123	5	56		7 9012
PRTPARM	123	6 890123	6 890123	5	56		7 9012
LABEL	123	6 890123	6 890123	5	56		7 9012
END	123	6 890123	6 890123	5	56		7 9012
					56		7 9012

RIGID FORMAT RESTART TABLES

7.19.4 Rigid Format Change Restart Table

Bit Position				Bit Position					
DMAPI	Inst.	63	70	80	DMAPI	Inst.	63	70	80
BEGIN		345678901234567		34	OFF				
FILE		345678901234567		34	LABEL				
GP1					EQUIV				
SAVE					CHKPNT				
PURGE					COND				
CHKPNT					MCE1				
COND					CHKPNT				
GP2					MCE2				
CHKPNT					CHKPNT				
PARAML					LABEL				
PURGE					EQUIV				
COND					CHKPNT				
PLTSET					COND				
SAVE					SCE1				
PRTHSG					CHKPNT				
PARAM					LABEL				
PARAM					EQUIV				
COND					EQUIV				
PLOT					EQUIV				
SAVE					CHKPNT				
PRTHSG					COND				
LABEL					SMP1				
CHKPNT					CHKPNT				
GP3					COND				
CHKPNT					SMP2				
TA1					CHKPNT				
SAVE					LABEL				
CHKPNT					COND				
COND					SMP2				
PARAM					CHKPNT				
PARAM					LABEL				
EMG					DPO				
SAVE					SAVE				
CHKPNT					COND				
COND					EQUIV				
EMA					PURGE				
CHKPNT					CHKPNT				
LABEL					MTRXIN				
COND					SAVE				
EMA					PARAM				
CHKPNT					PURGE				
LABEL					EQUIV				
PURGE					CHKPNT				
CHKPNT					COND				
LABEL					GKAD				
RMG					LABEL				
SAVE					EQUIV				
EQUIV					CHKPNT				
PURGE					TRLG				
CHKPNT					SAVE				
GP4					EQUIV				
SAVE					EQUIV				
PURGE					CHKPNT				
CHKPNT					TRHT				
COND					CHKPNT				
GPSP					VDR				
SAVE					SAVE				
COND					CHKPNT				

TRANSIENT HEAT TRANSFER ANALYSIS

DMAP	<u>Bit Position</u>		
Inst.	63	70	80
COND			
SDR3			
PARAM			
OFF			
SAVE			
CHKPNT			
XYTRAN			
SAVE			
XYPLOT			
LABEL			
PARAM	345678901234567		34
COND	345678901234567		34
EQUIV	345678901234567		34
COND	345678901234567		34
SDR1	345678901234567		34
LABEL	345678901234567		34
CHKPNT	345678901234567		34
PLTTRAN			
SAVE			
SDR2			
SDRHT			
EQUIV			
SDR3			
CHKPNT			
OFF			
SAVE			
COND			
PLOT			
SAVE			
PRTMSG			
LABEL			
XYTRAN			
SAVE			
XYPLOT			
LABEL			
JUMP	345678901234567		34
LABEL	345678901234567		34
: PRTPARM	345678901234567		34
LABEL	345678901234567		34
END	345678901234567		34

RIGID FORMAT RESTART TABLES

7.19.5 File Name Restart Table

DMAP	Bit Position			
Inst.	94	100	110	120
BEGIN				
FILE				
GP1	4			
SAVE	4			
PURGE				
CHKPNT				
COND				
GP2	5			
CHKPNT	5			
PARAML				0
PURGE				0
COND				0
PLTSET				0
SAVE				0
PRTMSG				0
PARAM				0
PARAM				0
COND				
PLOT				
SAVE				
PRTMSG				
LABEL				
CHKPNT				0
GP3	6			
CHKPNT	6			
TA1	7			
SAVE	7			
CHKPNT	7			
COND	8			
PARAM	8			
PARAM	9			
EMG				3
SAVE				3
CHKPNT				3
COND	8			
EMA	8			
CHKPNT	8			
LABEL	8			
COND	9			
EMA	9			
CHKPNT	9			
LABEL	9			
PURGE	9			
CHKPNT	9			
LABEL	8			
RMG	0			
SAVE	0			
EQUIV	0			
PURGE	0			
CHKPNT	0			
GP4	1			
SAVE	1			
PURGE	1 3 56	01	4	
CHKPNT	1 3 56	01	4	
COND	2			
GPSP	2			
SAVE	2			
COND	2			

TRANSIENT HEAT TRANSFER ANALYSIS

DMAP Inst.	94	Bit Position		120
		100	110	
OFF		2		
LABEL		2		
EQUIV		4		
CHKPNT		4		
COND		34		
MCE1		3		
CHKPNT		3		
MCE2		4		
CHKPNT		4		
LABEL		34		
EQUIV		5		
CHKPNT		5		
COND		5		
SCE1		5		
CHKPNT		5		
LABEL		5		
EQUIV		6		
EQUIV				1
EQUIV				2
CHKPNT		6		12
COND		6		12
SMP1		6		
CHKPNT		6		
COND				1
SMP2				1
CHKPNT				1
LABEL				1
COND				2
SMP2				2
CHKPNT				2
LABEL		6		12
DPD		7		
SAVE		7		
COND				
EQUIV			0	
PURGE		7		
CHKPNT				
MTRXIN		9		
SAVE		9		
PARAM		9		
PURGE			0	
EQUIV			0	
CHKPNT			0	
COND			0	
GKAD			0	
LABEL			0	
EQUIV			0	
CHKPNT			0	
TRLG				7
SAVE				7
EQUIV				7
EQUIV				7
CHKPNT				7
TRHT			1	
CHKPNT			1	
VOR			2	
SAVE			2	
CHKPNT			2	
COND			3	

RIGID FORMAT RESTART TABLES

DMAP Inst.	94	Bit Position		120
		100	110	
SDR3			3	
PARAM				
OFF				
SAVE				
CHKPNT			3	
XYTRAN				
SAVE				
XYPLOT				
LABEL				
PARAM			4	
COND			4	
EQUIV			4	
COND			4	
SDR1			4	
LABEL			4	
CHKPNT			4	
PLTTRAN				8
SAVE				8
SDR2			5	
SDRHT			5	
EQUIV			5	
SDR3			6	
CHKPNT			6	
OFF				
SAVE				
COND				
PLOT				
SAVE				
PRTMSG				
LABEL				
XYTRAN				
SAVE				
XYPLOT				
LABEL			4	
JUMP				
LABEL				
PRTPARM				
LABEL				
END				

7.20 RESTART TABLES FOR MODAL FLUTTER ANALYSIS

7.20.1 Bit Positions for Card Name Restart Table

<u>Card Name</u>	<u>Bit Pos.</u>	<u>Card Name</u>	<u>Bit Pos.</u>	<u>Card Name</u>	<u>Bit Pos.</u>
ADUM1	2	CMASS2	1	DMIG	57
ADUM2	2	CMASS3	1	EIGC	60
ADUM3	2	CMASS4	1	EIGP	60
ADUM4	2	CMETHODS	61	EIGR	58
ADUM5	2	CNGRNT	2	EPOINT	56
ADUM6	2	CONM1	5	FLFACT	36
ADUM7	2	CONM2	5	FLUTTER	36
ADUM8	2	CONROD	2	FMETHODS	38
ADUM9	2	CORD1C	24	GENEL	4
AEFACT	35	CORD1R	24	GRDPNT	15
AERO	37	CORD1S	24	GRDSET	1
AOUTS	21	CORD2C	24	GRID	1
ASET	11	CORD2R	24	GRIDB	1
ASET1	11	CORD2S	24	GUSTAERO	54
AXIC	1	COUPMASS	14	HFREQ	62
AXIF	1	CPBAR	14	LFREQ	62
AXSLOT	1	CPDPLT	14	LMODES	62
BAROR	2	CPENTA	2	K2PPS	57
B2PPS	57	CPQUAD1	14	KDAMP	62
CAERO1	37	CPQUAD2	14	MAT1	8
CAERO2	37	CPROD	14	MAT2	8
CAERO3	37	CPTRIA1	14	MAT3	8
CAERO4	37	CPTRIA2	14	MAT9	8
CAERO5	37	CPTRPLT	14	MATT1	8
CAXIF2	2	CPTUBE	14	MATT2	8
CAXIF3	2	CQDMEM	2	MATT3	8
CAXIF4	2	CQDMEM1	2	MATT9	8
CBAR	2	CQDMEM2	2	METHODS	59
CBARAO	2	CQDPLT	2	MKAERO1	34
CBEAM	2	CQUAD1	2	MKAERO2	34
CCONEAX	2	CQUAD2	2	MPC	9
CDUM1	2	CQUAD4	2	MPCADD	9
CDUM2	2	CQUADTS	2	MPCS	9
CDUM3	2	CRIGD1	9	MPCAX	9
CDUM4	2	CRIGD2	9	M2PPS	57
CDUM5	2	CROD	2	NODJE	26
CDUM6	2	CSHEAR	2	OMIT	11
CDUM7	2	CSLOT3	2	OMIT1	11
CDUM8	2	CSLOT4	2	OMITAX	11
CDUM9	2	CTETRA	2	PAERO1	29
CELAS1	1	CTRBSC	2	PAERO2	29
CELAS2	1	CTRAPAX	2	PAERO3	29
CELAS3	1	CTRIAAX	2	PAERO4	29
CELAS4	1	CTRIAX5	2	PAERO5	29
CFLUID2	2	CTRIARG	2	PBAR	3
CFLUID3	2	CTORDRG	2	PBEAM	3
CFLUID4	2	CTRAPRG	2	PCONEAX	3
CHEXA1	2	CTRIATS	2	PDUM1	3
CHEXA2	2	CTRIA1	2	PDUM2	3
CHEXA	2	CTRIA2	2	PDUM3	3
CHEX20	2	CTRIA3	2	PDUM4	3
CHEX8	2	CTRIM6	2	PDUM5	3
CIHEX1	2	CTRMEM	2	PDUM6	3
CIHEX2	2	CTRPLT	2	PDUM7	3
CIHEX3	2	CTUBE	2	PDUM8	3
CMASS1	1	CTWIST	2	PDUM9	3
		CHEDGE	2	PHEX	3

RIGID FORMAT RESTART TABLES

Card Name Bit Pos.

PELAS	6
PLOTEL	16
PLOTS	18
PMASS	7
POINTAX	1
POUTS	19
PQDMEM	3
PQDMEM1	3
PQDMEM2	3
PQDPLT	3
PQUAOTS	3
PQUAD1	3
PQUAD2	3
PQUAD4	3
PROD	3
PSHEAR	3
PSHELL	3
PSOLID	3
PTORDRG	3
PTRAPAX	3
PTRBSC	3
PTRIATS	3
PTRIA1	3
PTRIA2	3
PTRIM6	3
PTRIAAX	3
PTRMEM	3
PTRPLT	3
PTUBE	3
PTWIST	3
RBAR	9
RBE1	9
RBE2	9
RBE3	9
RINGAX	1
RINGFL	1
RROD	9
RSPLINE	9
RTRPLT	9
SDAMPS	55
SECTAX	1
SET1	32
SET2	32
SEQEP	56
SEQGP	1
SPC	10
SPC1	10
SPCADD	10
SPCAX	10
SPCS	10
SPLINE1	32
SPLINE2	32
SPLINE3	32
SPOINT	1
SUPAX	12
SUPPORT	12
TABDMP1	55
TABLEM1	8

Card Name Bit Pos.

TABLEM2	8
TABLEM3	8
TABLEM4	8
TEMP	13
TEMPAX	13
TEMPO	13
TEMPMTS	8
TEMPMXS	8
TEMPP1	13
TEMPP2	13
TEMPP3	13
TEMPRB	13
TF	40
TFS	57
VREF	39
WTMASS	14
XYOUTS	20

MODAL FLUTTER ANALYSIS

7.20.2 Bit Positions for File Name Restart Table

<u>File Name</u>	<u>Bit Pos.</u>	<u>File Name</u>	<u>Bit Pos.</u>	<u>File Name</u>	<u>Bit Pos.</u>
ACPT	124	KGG	100	SILD	111
AERO	124	KGGX	98	SILGA	124
AJJL	127	KHH	116	SKJ	127
BGPA	124	KLL	107	SPLINE	124
BGPDY	94	KLR	107	TFPOOL	111
BHH	116	KRR	107	ULL	108
BXHH	129	KNN	104	UOO	143
B2DD	139	KXHH	129	USET	101
B2PP	114	KOA	142	USETA	124
CASEYY	130	KOO	142	USETD	111
CASEZZ	131	K2DD	139	V	141
CLAMA	117	K2PP	114	XYPLTCE	120
CLAMAL	130	LAMA	112		
CLAMAL1	131	LLL	108		
CPHIA	132	LOO	143		
CPHID	122	MXHH	129		
CPHIH1	131	MAA	123		
CPHIK	133	MFF	105		
CPHIP	144	MGG	99		
CPHIPA	136	MHH	116		
CPHIPS	134	MI	112		
CSTM	94	MLL	107		
CSTMA	124	MLR	107		
DM	109	MNN	104		
D1JE	128	MR	110		
D1JK	127	MRR	107		
D2JE	128	M2DD	139		
D2JK	127	M2PP	114		
ECPT	97	OCEIGS	117		
ECT	95	OCPHIPA	137		
ECTA	124	OEFC1	137		
EED	111	OEIGS	112		
ELSETSA	125	OESC1	137		
EQAERO	124	OGPST	102		
EQEXIN	94	OGPWG	140		
EQDYN	111	OPHIH	119		
EST	97	OQPAC1	137		
FLIST	124	OVG	130		
FSAVE	129	PCPHIPA	137		
GEI	97	PKF	121		
GM	103	PHIA	112		
GMD	115	PHIDH	116		
GO	113	PHIH	117		
GOD	115	PHIHL	130		
GPCT	97	PLOTX2	118		
GPDY	94	PLOTX3	145		
GPL	94	PLTPARA	125		
GPLA	124	PLTSETA	125		
GPLD	111	QHHL	138		
GPSETSA	125	QKHL	138		
GPST	98	QHJL	138		
GPTT	96	QPAC	136		
GTKA	126	QPC	144		
KAA	106	RG	101		
KAAB	142	RP	132		
KFF	105	SIL	94		
KFS	105	SILA	124		

RIGID FORMAT RESTART TABLES

7.20.3 Card Name Restart Table

DMAP Inst.	1	10	20	Bit Position				40	50	60
				30						
BEGIN	1234567890123456	8901234	6	9	2	4567890			56789012	
FILE	12345678901234		234	6	9	2	4567890		56789012	
\$\$\$	4									
GP1	1			4						
SAVE	1			4						
COND	1			4						
GP2	12 45		6							
PARAML			8							
\$\$\$	1									
GP3	12		3							
\$\$\$	5									
TA1	1234567		3			4				
\$\$\$	5									
SAVE	1234567		3			4				
\$\$\$	5									
COND	1234567		3							
\$\$\$	5									
PARAM	123 6 8		3			4				
\$\$\$	4									
PARAM	123 5 7 8		34			4				
\$\$\$	4									
EMG	123 567 8		34			4				
\$\$\$	4									
SAVE	123 567 8		34			4				
\$\$\$	4									
COND	123 6 8		3			4				
\$\$\$	4									
EMA	123 6 8		3			4				
\$\$\$	4									
CHKPNT	123 6 8		3			4				
\$\$\$	4									
LABEL	123 6 8		3			4				
\$\$\$	4									
COND	123 5 7 8		34			4				
\$\$\$	4									
EMA	123 5 7 8		34			4				
\$\$\$	4									
CHKPNT	123 5 7 8		34			4				
\$\$\$	4									
COND	123 5 7 8		345			4				
\$\$\$	4									
GPWG	123 5 7 8		345			4				
\$\$\$	4									
OFF	123 5 7 8		345			4				
\$\$\$	4									
LABEL	123 5 7 8		345			4				
\$\$\$	4									
EQUIV	1234 6 8		3			4				
\$\$\$	4									
CHKPNT	1234 6 8		3			4				
\$\$\$	4									
COND	1234 6 8		3			4				
\$\$\$	4									
SMA3	1 4									
\$\$\$	4									
CHKPNT	1 4									
\$\$\$	4									
ADD	1234 6 8		3			4				

MODAL FLUTTER ANALYSIS

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
\$\$\$		4					
CHKPNT	1234	6 8	3	4			
\$\$\$		4					
LABEL	1234	6 8	3	4			
\$\$\$		4					
GP4	1		9012	4			
SAVE	1		9012	4			
CHKPNT	1		9012	4			
\$\$\$		4					
GPSP	1234	6 890	3	4			
\$\$\$		4					
SAVE	1234	6 890	3	4			
\$\$\$		4					
COND	1234	6 890	3	4			
\$\$\$		4					
OFF	1234	6 890	3	4			
\$\$\$		4					
LABEL	1234	6 890	3	4			
\$\$\$		4					
EQUIV	123456789		34	4			
\$\$\$		4					
COND	123456789		34	4			
\$\$\$		4					
MCE1	1		9	4			
\$\$\$		4					
MCE2	123456789		34	4			
\$\$\$		4					
CHKPNT	123456789		34	4			
\$\$\$		4					
LABEL	123456789		34	4			
\$\$\$		4					
CHKPNT	1		9	4			
\$\$\$		4					
EQUIV	1234567890		34	4			
\$\$\$		4					
COND	1234567890		34	4			
\$\$\$		4					
SCE1	1234567890		34	4			
\$\$\$		4					
LABEL	1234567890		34	4			
\$\$\$		4					
CHKPNT	1234567890		34	4			
\$\$\$		4					
EQUIV	12345678901		34	4			
\$\$\$		4					
CHKPNT	12345678901		34	4			
\$\$\$		4					
PURGE	1234	6 8901	3	4			
\$\$\$		4					
CHKPNT	1234	6 8901	3	4			
\$\$\$		4					
COND	12345678901		34	4			
\$\$\$		4					
PARAM							
\$\$\$		4					
VEC	1		901				
\$\$\$		4					
PARTN	1234	6 8901	3	4			
\$\$\$		4					

RIGID FORMAT RESTART TABLES

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
CHKPNT	1234	6 8901 3	4				
\$\$\$	4						
DECOMP	1234	6 8901 3	4				
\$\$\$	4						
SAVE	1234	6 8901 3	4				
\$\$\$	4						
COND	1234	6 8901 3	4				
\$\$\$	4						
JUMP	1234	6 8901 3	4				
\$\$\$	4						
LABEL	1234	6 8901 3	4				
\$\$\$	4						
PRTPARM	1234	6 8901 3	4				
\$\$\$	4						
LABEL	1234	6 8901 3	4				
\$\$\$	4						
CHKPNT	1234	6 8901 3	4				
\$\$\$	4						
FBS	1234	6 8901 3	4				
\$\$\$	4						
CHKPNT	1234	6 8901 3	4				
\$\$\$	4						
MPYAD	1234	6 8901 3	4				
\$\$\$	4						
CHKPNT	1234	6 8901 3	4				
\$\$\$	4						
SMP2	1234567	8901 34	4				
\$\$\$	4						
CHKPNT	1234567	8901 34	4				
\$\$\$	4						
LABEL	1234567	8901 34	4				
\$\$\$	4						
COND	1234567	8901234	4				
\$\$\$	4						
RBMG1	1234567	8901234	4				
\$\$\$	4						
CHKPNT	1234567	8901234	4				
\$\$\$	4						
RBMG2	1234	6 890123	4				
\$\$\$	4						
CHKPNT	1234	6 890123	4				
\$\$\$	4						
RBMG3	1234	6 890123	4				
\$\$\$	4						
CHKPNT	1234	6 890123	4				
\$\$\$	4						
RBMG4	1234567	8901234	4				
\$\$\$	4						
CHKPNT	1234567	8901234	4				
\$\$\$	4						
LABEL	1234567	8901234	4				
\$\$\$	4						
DPD	1	9012			0		6 8 0
SAVE	1	9012			0		6 8 0
COND	1	9012			0		6 8 0
\$\$\$	4						
EQUIV	1234	6 901 34	4			6	
\$\$\$	4						
READ	1234567	8901234	4				89

MODAL FLUTTER ANALYSIS

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
\$\$\$		4					
SAVE	12345678901234		4				89
\$\$\$		4					
CHKPNT	12345678901234		4				89
\$\$\$		4					
OFF	12345678901234		4				89
\$\$\$		4					
SAVE	12345678901234		4				89
\$\$\$		4					
COND	12345678901234		4				89
\$\$\$		4					
OFF	12345678901234		4				89
\$\$\$		4					
SAVE	12345678901234		4				89
\$\$\$		4					
MTRXIN	1		23		0		67
\$\$\$		5					
SAVE	1		23		0		67
\$\$\$		5					
EQUIV	1	901	23		0		67
\$\$\$		4					
CHKPNT	1	901	23		0		67
\$\$\$		4					
GKAD	1234 6 8901 34		234		0		67
\$\$\$		4					
CHKPNT	1234 6 8901 34		234		0		67
\$\$\$		4					
GKAM	12345678901234		234		0		56789 2
\$\$\$		4					
SAVE	12345678901234		234		0		56789 2
\$\$\$		4					
CHKPNT	12345678901234		234		0		56789 2
\$\$\$		4					
APD	12	9012 6	4	9 2 4567		6	
SAVE	12	9012 6	4	9 2 4567		6	
PARAM			8				
\$\$\$	1						
COND			8				
\$\$\$	1						
PARAM			8				
\$\$\$	1						
PLTSET			8				
\$\$\$	1						
SAVE			8				
\$\$\$	1						
PRTMSG			8				
\$\$\$	1						
COND			8				
\$\$\$	1						
PLOT			8				
\$\$\$	1						
SAVE			8				
\$\$\$	1						
PRTMSG			8				
\$\$\$	1						
LABEL			8				
\$\$\$	1						
COND							
GI	1234 6 8901 3		4	2 5 7		8 0	

RIGID FORMAT RESTART TABLES

DMAP Inst.	1	10	20	<u>Bit Position</u>		40	50	60
				30				
\$\$\$	4							
CHKPNT	1234	6 8901 3		4	2 5 7			
\$\$\$	4							
PARAM				4	9 5 7			
\$\$\$	7							
AMG				4	9 45 7			
\$\$\$	7							
SAVE				4	9 45 7			
\$\$\$	7							
CHKPNT				4	9 45 7			
\$\$\$	7							
COND				6	7			6
\$\$\$	4							
INPUTT2				6	7			6
\$\$\$	4							
LABEL				6	7			6
\$\$\$	4							
PARAM	12345678901234			4 6 9 2 5 7				6 89 2
\$\$\$	4							
AMP	12345678901234			4 6 9 2 45 7				4 6 89 2
\$\$\$	4							
SAVE	12345678901234			4 6 9 2 45 7				4 6 89 2
\$\$\$	4							
CHKPNT	12345678901234			4 6 9 2 45 7				4 6 89 2
\$\$\$	4							
PARAM			01					
\$\$\$	4							
PARAM			89					
\$\$\$	4							
PARAM			1					
\$\$\$	4							
PARAM	12345678901234			4 6 9 2 4567890				56789012
\$\$\$	4							
JUMP	12345678901234			4 6 9 2 4567890				56789012
\$\$\$	34							
LABEL	12345678901234			4 6 9 2 4567890				56789012
\$\$\$	34							
FA1	12345678901234			4 6 9 2 4567890				56789012
\$\$\$	4							
SAVE	12345678901234			4 6 9 2 4567890				56789012
\$\$\$	4							
CHKPNT	12345678901234			4 6 9 2 4567890				56789012
\$\$\$	4							
EQUIV	12345678901234			4 6 9 2 4567890				56789012
\$\$\$	4							
CHKPNT	12345678901234			4 6 9 2 4567890				56789012
\$\$\$	4							
COND	12345678901234			4 6 9 2 4567890				56789012
\$\$\$	4							
CEAD	12345678901234			4 6 9 2 4567890				56789012
\$\$\$	4							
SAVE	12345678901234			4 6 9 2 4567890				56789012
\$\$\$	4							
COND	12345678901234			4 6 9 2 4567890				56789012
\$\$\$	4							
LABEL	12345678901234			4 6 9 2 4567890				56789012
\$\$\$	4							
COND			8 1					

MODAL FLUTTER ANALYSIS

DMAP Inst.	1	10	20	<u>Bit Position</u>				40	50	60
				30						
\$\$\$	4									
VDR			1							
\$\$\$	4									
SAVE			1							
\$\$\$	4									
COND			1							
\$\$\$	4									
OFF			1							
\$\$\$	4									
SAVE			1							
\$\$\$	4									
LABEL			8	1						
\$\$\$	4									
FA2	12345678901234		8	1	4	6	9	2	4567890	56789012
\$\$\$	4									
SAVE	12345678901234		8	1	4	6	9	2	4567890	56789012
\$\$\$	4									
CHKPNT	12345678901234		8	1	4	6	9	2	4567890	56789012
\$\$\$	34									
COND	12345678901234		8	1	4	6	9	2	4567890	56789012
\$\$\$	34									
LABEL	12345678901234		8	1	4	6	9	2	4567890	56789012
\$\$\$	34									
COND	12345678901234		8	1	4	6	9	2	4567890	56789012
\$\$\$	34									
REPT	12345678901234		8	1	4	6	9	2	4567890	56789012
\$\$\$	34									
JUMP	12345678901234		8	1	4	6	9	2	4567890	56789012
\$\$\$	34									
LABEL	12345678901234		8	1	4	6	9	2	4567890	56789012
\$\$\$	34									
CHKPNT	12345678901234		8	1	4	6	9	2	4567890	56789012
\$\$\$	34									
PARAML			0							
\$\$\$	4									
COND			0							
\$\$\$	4									
XYTRAN			0							
\$\$\$	4									
SAVE			0							
\$\$\$	4									
COND			0							
\$\$\$	4									
XYPLOT			0							
\$\$\$	4									
LABEL			0							
\$\$\$	4									
PARAM	1234567890123		01	4	6	9	2	4567890		56789012
\$\$\$	4 6									
COND	1234567890123		01	4	6	9	2	4567890		56789012
\$\$\$	4 6									
MODACC	1234567890123		1	4	6	9	2	4567890		56789012
\$\$\$	4 6									
ADR			1							
\$\$\$	4 6									
DDR1	1234567890123			4	6	9	2	4 67890		56789012
\$\$\$	4			4						
CHKPNT	1234567890123			4	6	9	2	4 67890		56789012

RIGID FORMAT RESTART TABLES

DMAP Inst.	1	10	20	Bit Position 30				40	50	60
\$\$\$		4 6								
EQUIV		1234567890123		4 6	9	2	4567890			56789012
\$\$\$		4 6								
COND		1234567890123		4 6	9	2	4567890			56789012
\$\$\$		4 6								
SDR1		1234567890123		4 6	9	2	4567890			56789012
\$\$\$		4 6								
LABEL		1234567890123		4 6	9	2	4567890			56789012
\$\$\$		4								
CHKPNT		1234567890123		4 6	9	2	4567890			56789012
\$\$\$		4 6								
EQUIV		1234567890123		4 6	9	2	4567890			56789012
\$\$\$		4 6								
COND		1234567890123		4 6	9	2	4567890			56789012
\$\$\$		4 6								
VEC		1234567890123		4 6	9	2	4567890			56789012
\$\$\$		4 6								
PARTN		1234567890123		4 6	9	2	4567890			56789012
\$\$\$		4 6								
LABEL		1234567890123		4 6	9	2	4567890			56789012
\$\$\$		4 6								
MPYAD		1234567890123		4 6	9	2	4567890			56789012
\$\$\$		4 6								
UMERGE		1234567890123		4 6	9	2	4567890			56789012
\$\$\$		4 6								
UMERGE		1234567890123		4 6	9	2	4567890			56789012
\$\$\$		4 6								
CHKPNT		1234567890123		4 6	9	2	4567890			56789012
\$\$\$		4 6								
UMERGE		1234567890123		4 6	9	2	4567890			56789012
\$\$\$		4 6								
CHKPNT		1234567890123		4 6	9	2	4567890			56789012
\$\$\$		4 6								
SDR2		4	89	4						
\$\$\$		4 6								
CHKPNT		4	89	4						
\$\$\$		4 6								
OFF			9							
\$\$\$		4 6								
COND			8							
\$\$\$	1	4 6								
PLOT			8							
\$\$\$	1	4 6								
PRTMSG			8							
\$\$\$	1	4 6								
JUMP		1234567890123456	8901234	6	9	2	4567890			56789012
LABEL		1234567890123456	8901234	6	9	2	4567890			56789012
\$\$\$		4								
PRTPARM		1234567890123456	8901234	6	9	2	4567890			56789012
\$\$\$		4								
LABEL		1234567890123456	8901234	6	9	2	4567890			56789012
PRTPARM		1234567890123456	8901234	6	9	2	4567890			56789012
LABEL		1234567890123456	8901234	6	9	2	4567890			56789012
PRTPARM		1234567890123456	8901234	6	9	2	4567890			56789012
LABEL		1234567890123456	8901234	6	9	2	4567890			56789012
\$\$\$		4								
PRTPARM		1234567890123456	8901234	6	9	2	4567890			56789012
\$\$\$		4								
LABEL		1234567890123456	8901234	6	9	2	4567890			56789012
END		1234567890123456	8901234	6	9	2	4567890			56789012

MODAL FLUTTER ANALYSIS

7.20.4 Rigid Format Change Restart Table

DMAP Inst.	63	Bit Position 70	80	DMAP Inst.	63	Bit Position 70	80
BEGIN		345678901234567	901234567890123	CHKPNT		567	01234567890123
FILE		345678901234567	901234567890123	COND		567	01234567890123
GP1		567	01234567890123	PARAM	345678901234567	901234567890123	
SAVE		567	01234567890123	VEC		567	01234567890123
COND		345678901234567	901234567890123	PARTN		567	01234567890123
GP2		567	01234567890123	CHKPNT		567	01234567890123
PARAML		345678901234567	901234567890123	DECOMP		567	01234567890123
GP3		567	01234567890123	SAVE		567	01234567890123
TA1		567	01234567890123	COND		567	01234567890123
SAVE		567	01234567890123	JUMP		567	01234567890123
COND		345678901234567	901234567890123	LABEL		567	01234567890123
PARAM		567	01234567890123	PRTPARM		567	01234567890123
PARAM		567	01234567890123	LABEL		567	01234567890123
EMG		567	01234567890123	CHKPNT		567	01234567890123
SAVE		567	01234567890123	FBS		567	01234567890123
COND		567	01234567890123	CHKPNT		567	01234567890123
EMA		567	01234567890123	MPYAD		567	01234567890123
CHKPNT		567	01234567890123	CHKPNT		567	01234567890123
LABEL		567	01234567890123	SMP2		567	01234567890123
COND		345678901234567	901234567890123	CHKPNT		567	01234567890123
EMA		567	01234567890123	LABEL		567	01234567890123
CHKPNT		567	01234567890123	COND		567	01234567890123
COND		567	01234567890123	RBMG1		567	01234567890123
GPWG		567	01234567890123	CHKPNT		567	01234567890123
OFF		567	01234567890123	RBMG2		567	01234567890123
LABEL		567	01234567890123	CHKPNT		567	01234567890123
EQUIV		567	01234567890123	RBMG3		567	01234567890123
CHKPNT		567	01234567890123	CHKPNT		567	01234567890123
COND		567	01234567890123	RBMG4		567	01234567890123
SMA3		567	01234567890123	CHKPNT		567	01234567890123
CHKPNT		567	01234567890123	LABEL		567	01234567890123
ADD		567	01234567890123	DPD		567	01234567890123
CHKPNT		567	01234567890123	SAVE		567	01234567890123
LABEL		567	01234567890123	COND	345678901234567	901234567890123	
GP4		567	01234567890123	EQUIV		567	01234567890123
SAVE		567	01234567890123	READ		567	01234567890123
CHKPNT		567	01234567890123	SAVE		567	01234567890123
GPSP		567	01234567890123	CHKPNT		567	01234567890123
SAVE		567	01234567890123	OFF		567	01234567890123
COND		567	01234567890123	SAVE		567	01234567890123
OFF		567	01234567890123	COND	345678901234567	901234567890123	
LABEL		567	01234567890123	OFF		567	01234567890123
EQUIV		567	01234567890123	SAVE		567	01234567890123
COND		567	01234567890123	MTRXIN		567	01234567890123
MCE1		567	01234567890123	SAVE		567	01234567890123
MCE2		567	01234567890123	EQUIV		567	01234567890123
CHKPNT		567	01234567890123	CHKPNT		567	01234567890123
LABEL		567	01234567890123	GKAD		567	01234567890123
CHKPNT		567	01234567890123	CHKPNT		567	01234567890123
EQUIV		567	01234567890123	GKAM		567	01234567890123
COND		567	01234567890123	SAVE		567	01234567890123
SCE1		567	01234567890123	CHKPNT		567	01234567890123
LABEL		567	01234567890123	APD		567	01234567890123
CHKPNT		567	01234567890123	SAVE		567	01234567890123
EQUIV		567	01234567890123	PARAM		567	01234567890123
CHKPNT		567	01234567890123	COND		567	01234567890123
PURGE		567	01234567890123	PARAM		567	01234567890123

RIGID FORMAT RESTART TABLES

DMAP Inst.	63	Bit Position 70	80	DMAP Inst.	63	Bit Position 70	80
PLTSET			567 01234567890123	XYPLOT			567 01234567890123
SAVE			567 01234567890123	LABEL			567 01234567890123
PRTMSG			567 01234567890123	PARAM			567 01234567890123
COND			567 01234567890123	COND			567 01234567890123
PLOT			567 01234567890123	MODACC			567 01234567890123
SAVE			567 01234567890123	ADR			567 01234567890123
PRTMSG			567 01234567890123	DDR1			567 01234567890123
LABEL			567 01234567890123	CHKPNT			567 01234567890123
COND			567 01234567890123	EQUIV			567 01234567890123
GI			567 01234567890123	COND			567 01234567890123
CHKPNT			567 01234567890123	SDR1			567 01234567890123
PARAM	345678901234	567	901234567890123	LABEL			567 01234567890123
AMG			567 01234567890123	CHKPNT			567 01234567890123
SAVE			567 01234567890123	EQUIV			567 01234567890123
CHKPNT			567 01234567890123	COND			567 01234567890123
COND			567 01234567890123	VEC			567 01234567890123
INPUTT2			567 01234567890123	PARTN			567 01234567890123
LABEL			567 01234567890123	LABEL			567 01234567890123
PARAM			567 01234567890123	MPYAD			567 01234567890123
AMP			567 01234567890123	UMERGE			567 01234567890123
SAVE			567 01234567890123	UMERGE			567 01234567890123
CHKPNT			567 01234567890123	CHKPNT			567 01234567890123
PARAM			567 01234567890123	UMERGE			567 01234567890123
PARAM			567 01234567890123	CHKPNT			567 01234567890123
PARAM	345678901234	567	901234567890123	SDR2			
PARAM	345678901234	567	901234567890123	CHKPNT			
JUMP	345678901234	567	901234567890123	OFF			
LABEL	345678901234	567	901234567890123	COND			
FA1			567 01234567890123	PLOT			
SAVE			567 01234567890123	PRTMSG			
CHKPNT			567 01234567890123	JUMP	345678901234	567	901234567890123
EQUIV			567 01234567890123	LABEL	345678901234	567	901234567890123
CHKPNT			567 01234567890123	PRTPARM	345678901234	567	901234567890123
COND			567 01234567890123	LABEL	345678901234	567	901234567890123
CEAD			567 01234567890123	PRTPARM	345678901234	567	901234567890123
SAVE			567 01234567890123	LABEL	345678901234	567	901234567890123
COND			567 01234567890123	PRTPARM	345678901234	567	901234567890123
LABEL			567 01234567890123	LABEL	345678901234	567	901234567890123
COND			567 01234567890123	PRTPARM	345678901234	567	901234567890123
VDR			567 01234567890123	LABEL	345678901234	567	901234567890123
SAVE			567 01234567890123	END	345678901234	567	901234567890123
COND			567 01234567890123				
OFF			567 01234567890123				
SAVE			567 01234567890123				
LABEL			567 01234567890123				
FA2			567 01234567890123				
SAVE			567 01234567890123				
CHKPNT			567 01234567890123				
COND	345678901234	567	901234567890123				
LABEL	345678901234	567	901234567890123				
COND	345678901234	567	901234567890123				
REPT	345678901234	567	901234567890123				
JUMP	345678901234	567	901234567890123				
LABEL	345678901234	567	901234567890123				
CHKPNT	345678901234	567	901234567890123				
PARAML			567 01234567890123				
COND			567 01234567890123				
XYTRAN			567 01234567890123				
SAVE			567 01234567890123				
COND			567 01234567890123				

MODAL FLUTTER ANALYSIS

7.20.5 File Name Restart Table

DMAP	Bit Position		
Inst.	94	100	110
BEGIN			120
FILE			7 8
GP1	4		
SAVE	4		
COND	4		
GP2	5		
PAPAML		2	5
GP3	6		
TA1	7		
SAVE	7		
COND	7		
PARAM	8		
PARAM	9		
EMG	89		
SAVE	89		
COND	8		
EMA	8		
CHKPNT	8		
LABEL	8		
COND	9		
EMA	9		
CHKPNT	9		
COND			0
GPWG			0
OFF			0
LABEL			0
EQUIV	0		
CHKPNT	0		
COND	0		
SMA3	0		
CHKPNT	0		
ADD	0		
CHKPNT	0		
LABEL	0		
GP4	1		
SAVE	1		
CHKPNT	1		
GPSP	2		
SAVE	2		
COND	2		
OFF	2		
LABEL	2		
EQUIV	4		
COND	34		
MCE1	3		
MCE2	4		
CHKPNT	4		
LABEL	34		
CHKPNT	3		
EQUIV	5		
COND	5		
SCE1	5		
LABEL	5		
CHKPNT	5		
EQUIV	6		3
CHKPNT	6		3
PURGE		3	
CHKPNT		3	

RIGID FORMAT RESTART TABLES

DMAP Inst.	94	Bit Position		120	
		100	110		
COND		6	3	3	
PARAM		6			
VEC					1
PARTN					2
CHKPNT					2
DECOMP					3
SAVE					3
COND					3
JUMP					3
LABEL					3
PRTPARM					3
LABEL					3
CHKPNT					3
FBS			3		
CHKPNT			3		
MPYAD		6			
CHKPNT		6			
SMP2				3	
CHKPNT				3	
LABEL		6	3	3	
COND		7890			
RBMG1		7			
CHKPNT		7			
RBMG2		8			
CHKPNT		8			
RBMG3		9			
CHKPNT		9			
RBMG4		0			
CHKPNT		0			
LABEL		7890			
DPD			1		
SAVE			1		
COND			1		
EQUIV				5	
READ			2		
SAVE			2		
CHKPNT			2		
OFF			2		
SAVE			2		
COND			2		
OFF			2		
SAVE			2		
MTRXIN			4		
SAVE			4		
EQUIV					
CHKPNT			4		9
GKAD			5		9
CHKPNT			5		9
GKAM			6		
SAVE			6		
CHKPNT			6		
APD				4	
SAVE				4	
PARAM					
COND				5	
PARAM				5	
PLTSET				5	
SAVE				5	
PRTMSG				5	

MODAL FLUTTER ANALYSIS

DMAP		Bit Position			
Inst.	94	100	110	120	
COND				5	
PLOT			8		
SAVE			8		
PRMSG			8		
LABEL				5	
COND			1		
GI				6	
CHKPNT				6	
PARAM				7	
AMG				7	
SAVE				7	
CHKPNT				7	
COND				8	
INPUTT2				8	
LABEL				8	
PARAM					8
AMP					8
SAVE					8
CHKPNT					8
PARAM				9	
PARAM				9	
PARAM				9	
PARAM				9	
JUMP				9	
LABEL				9	
FA1				9	
SAVE				9	
CHKPNT		7			
EQUIV		7			
CHKPNT		7			
COND		7		9	
CEAD		7			
SAVE		7			
COND		7			
LABEL		7		9	
COND		9			
VDR		9			
SAVE		9			
COND		9			
OFF		9			
SAVE		9			
LABEL		9			
FA2				0	
SAVE				0	
CHKPNT				0	
COND					
LABEL					
COND					
REPT					
JUMP					
LABEL				0	
CHKPNT				0	
PARAML		0			
COND		0			
XYTRAN		0			
SAVE		0			
COND		0			
XYPLOT		0			
LABEL		0			
PARAM		0			

RIGID FORMAT RESTART TABLES

DMAP		Bit Position			
Inst.	94	100	110	120	
COND					
MODACC				1	
ADR			1		
DDR1			2		
CHKPNT			2		
EQUIV					4
COND					4
SDR1					4
LABEL					4
CHKPNT					4
EQUIV				2	
COND				2	
VEC				2	
PARTN				2	
LABEL				2	
MPYAD				3	
UMERGE				4	
UMERGE					6
CHKPNT					6
UMERGE					6
CHKPNT					6
SDR2					7
CHKPNT					7
OFF					
COND					5
PLOT					5
PRTMSG					5
JUMP					5
LABEL					
PRTPARM					
LABEL			1		
PRTPARM			1		
LABEL	4	7	9		
PRTPARM	4	7	9		
LABEL			2		
PRTPARM			2		
LABEL					
END					

MODAL AEROELASTIC RESPONSE

7.21 RESTART TABLES FOR MODAL AEROELASTIC RESPONSE

7.21.1 Bit Positions for Card Name Restart Table

Card Name	Bit Pos.	Card Name	Bit Pos.	Card Name	Bit Pos.
ADUM1	2	CHASS3	1	DLOAD	52
ADUM2	2	CHASS4	1	DPHASE	52
ADUM3	2	CONM1	5	DLOADS	61
ADUM4	2	CONM2	5	DMIG	57
ADUM5	2	CONROD	2	EPOINT	56
ADUM6	2	CORD1C	24	EIGR	58
ADUM7	2	CORD1R	24	FREQ	52
ADUM8	2	CORD1S	24	FREQ1	52
ADUM9	2	CORD2C	24	FREQ2	52
AEFACT	35	CORD2R	24	FREQS	61
AERO	37	CORD2S	24	GENEL	4
ASET	11	COUPMASS	14	GRDPNT	15
ASET1	11	CNGRNT	2	GRDSET	1
AOUTS	21	CPBAR	14	GRID	1
AXYOUTS	22	CPENTA	2	GRIDB	1
AXIC	1	CPDPLT	14	GUST	49
AXIF	1	CPQUAD1	14	GUSTS	49
BAROR	2	CPQUAD2	14	GUSTAERO	48
B2PPS	57	CPROD	14	HFREQ	62
CAERO1	37	CPTRIA1	14	IFTM	27
CAERO2	37	CPTRIA2	14	LFREQ	62
CAERO3	37	CPTRPLT	14	LMODES	62
CAERO4	37	CPTUBE	14	K2PPS	57
CAERO5	37	CQDMEM	2	KDAMP	62
CAXIF2	2	CQDMEM1	2	MACH	28
CAXIF3	2	CQDMEM2	2	MAT1	8
CAXIF4	2	CQDPLT	2	MAT2	8
CBAR	2	CQUAD1	2	MAT3	8
CBEAM	2	CQUAD2	2	MAT9	8
CCONEAX	2	CQUAD4	2	MATT1	8
CDUM1	2	CQUADTS	2	MATT2	8
CDUM2	2	CTRIATS	2	MATT3	8
CDUM3	2	CRIGD1	9	MATT9	8
CDUM4	2	CRIGD2	9	METHODS	59
CDUM5	2	CROD	2	MKAERO1	34
CDUM6	2	CSHEAR	2	MKAERO2	34
CDUM7	2	CSLOT3	2	MPC	9
CDUM8	2	CSLOT4	2	MPCADD	9
CDUM9	2	CTETRA	2	MPCS	9
CELAS1	1	CTORDRG	2	MPCAX	9
CELAS2	1	CTRAPRG	2	M2PPS	57
CELAS3	1	CTRBSC	2	NODJE	26
CELAS4	1	CTRIAAX	2	OMIT	11
CFLUID2	2	CTRIAX6	2	OMIT1	11
CFLUID3	2	CTRIA1	2	OMITAX	11
CFLUID4	2	CTRIA2	2	PAERO1	29
CHEXA	2	CTRIA3	2	PAERO2	29
CHEXA1	2	CTRIARG	2	PAERO3	29
CHEXA2	2	CTRIM6	2	PAERO4	29
CHEX20	2	CTRMEM	2	PAERO5	29
CHEX8	2	CTRPLT	2	PBAR	3
CIHEX1	2	CTUBE	2	PCONEAX	3
CIHEX2	2	CTWIST	2	PDUM1	3
CIHEX3	2	CWEDGE	2	PDUM2	3
CHASS1	1	DAREA	52	PDUM3	3
CHASS2	1	DELAY	52	PDUM4	3

RIGID FORMAT RESTART TABLES

Card Name Bit Pos.

PDUM5	3
PDUM6	3
PDUM7	3
PDUM8	3
PDUM9	3
PHEX	3
PELAS	6
PLOTEL	16
PLOTS	18
PMASS	7
POINTAX	1
POUTS	19
PQDMEM	3
PQDMEM1	3
PQDMEM2	3
PQDPLT	3
PQUAD1	3
PQUAD2	3
PQUAD4	3
PQUADTS	3
PROD	3
PSHEAR	3
PSHELL	3
PSOLID	3
PTORDRG	3
PTRAPAX	3
PTRBSC	3
PTRIA1	3
PTRIA2	3
PTRIATS	3
PTRIAAX	3
PTRIM6	3
PTRMEM	3
PTRPLT	3
PTUBE	3
PTWIST	3
Q	28
RANDPS	53
RANDT1	53
RANDOMS	54
RBAR	9
RBE1	9
RBE2	9
RBE3	9
RINGAX	1
RINGFL	1
RROD	9
RLOAD1	52
RLOAD2	52
RSPLINE	9
RTRPLT	9
SDAMPS	55
SECTAX	1
SET1	32
SET2	32
SEQEP	56
SEQGP	1
SPC	10

Card Name Bit Pos.

SPC1	10
SPCADD	10
SPCAX	10
SPCS	10
SPLINE1	32
SPLINE2	32
SPLINE3	32
SPOINT	1
SUPAX	12
SUPPORT	12
TABDMP1	55
TABLED1	51
TABLED2	51
TABLED3	51
TABLED4	51
TABRND1	54
TABRNDG	54
TABLEM1	8
TABLEM2	8
TABLEM3	8
TABLEM4	8
TEMP	13
TEMPAX	13
TEMPD	13
TEMPHTS	8
TEMPHXS	8
TEMPP1	13
TEMPP2	13
TEMPP3	13
TEMPRB	13
TF	40
TFS	57
TLOAD1	52
TLOAD2	52
TSTEP	50
TSTEPS	60
WTHASS	14
XYOUTS	20

MODAL AEROELASTIC RESPONSE

7.21.2 Bit Positions for File Name Restart Table

<u>File Name</u>	<u>Bit Pos.</u>	<u>File Name</u>	<u>Bit Pos.</u>	<u>File Name</u>	<u>Bit Pos.</u>
ACPT	124	K2PP	114	QP	137
AERO	124	LAMA	112	QPA	140
AJJL	127	LLL	108	RG	101
AUTO	146	LOO	151	SIL	94
BGPA	124	MAA	123	SILA	124
BGPD	94	MEF1	151	SILD	111
BHH	116	MEF2	142	SILGA	124
B2DD	139	MES1	141	SKJ	127
B2PP	114	MES2	142	SPLINE	124
CSTM	94	MFF	105	TFPOOL	111
CSTMA	124	MGG	99	TOL1	133
DLT	111	MHH	116	TOL	132
DM	109	MI	112	TRL	111
D1JE	128	MLL	107	UHVF	131
D1JK	127	MLR	107	UHV1	133
D2JE	128	MNN	104	UHV2	132
D2JK	127	MPHIPA1	141	UOO	151
ECPT	97	MPHIPA2	142	ULL	108
ECT	95	MQP1	151	UVT1	
ECTA	124	MQP2	142	USET	101
EED	111	MR	110	USETA	124
ELSETSA	125	MRR	107	USETD	111
EQAERO	124	M2DD	139	V	153
EQEXIN	94	M2PP	114	XYPLTR	146
EQDYN	111	OEFC1	143	XYPLTT	145
EST	97	OEFC2	137	XYPTTA	136
FLIST	124	OEIGS	112		
FOL	129	OES2	143		
FRL	111	OGPWG	154		
GEI	97	OGPST	102		
GM	103	OPP1	142		
GMD	115	OPP2	142		
GO	113	OQP2	143		
GOD	115	OUHV1	135		
GPCT	97	OUHV2	135		
GPD	94	OUPV2	143		
GPL	94	PDF	129		
GPLA	124	PHF1	129		
GPLD	111	PHF	130		
GPSETSA	125	PHIA	112		
GPST	98	PHIDH	116		
GPTT	96	PHIK	140		
GTKA	126	PHIPA	140		
KAA	106	PHIPS	140		
KAA8	152	PHIP	137		
KFF	105	PKF	134		
KFS	105	PLOTX2	150		
KGG	100	PLOTX3	144		
KGGX	98	PLTPARA	125		
KGGY	100	PLTSETA	125		
KHH	116	PPF	129		
KLL	107	PSDF	146		
KLR	107	PSDL	111		
KRR	107	PSF	129		
KNN	104	PUPPAT	144		
KOA	152	QHHL	138		
KOO	152	QKHL	138		
K2DD	139	QHJL	138		

RIGID FORMAT RESTART TABLES

7.21.3 Card Name Restart Table

DMAP Inst.	Bit Position												
	1	10	20	30	40	50	60						
BEGIN	1234567890123456	89012	4 6	9 2	4567890	90123456789012							
FILE	12345678901234	9 12	4 6	9 2	4567890	90123456789012							
\$\$\$	4												
GP1	1			4									
SAVE	1			4									
COND	1			4									
GP2	12 45	6											
PARAML		8											
\$\$\$	1												
PARAML			0 2										
\$\$\$	4												
PARAM													
\$\$\$	4												
PARAM													
\$\$\$	4												
GP3	12	3											
\$\$\$	5												
TA1	1234567	3		4									
\$\$\$	5												
SAVE	1234567	3		4									
\$\$\$	5												
COND	1234567	3		4									
\$\$\$	5												
PARAM	123 6 8	3		4									
\$\$\$	4												
PARAM	123 5 7 8	34		4									
\$\$\$	4												
EMG	123 567 8	34		4									
\$\$\$	4												
SAVE	123 567 8	34		4									
\$\$\$	4												
COND	123 6 8	3		4									
\$\$\$	4												
EMA	123 6 8	3		4									
\$\$\$	4												
CHKPNT	123 6 8	3		4									
\$\$\$	4												
LABEL	123 6 8	3		4									
\$\$\$	4												
COND	123 5 7 8	34		4									
\$\$\$	4												
EMA	123 5 7 8	34		4									
\$\$\$	4												
CHKPNT	123 5 7 8	34		4									
\$\$\$	4												
COND	123 5 7 8	345		4									
\$\$\$	4												
GPWG	123 5 7 8	345		4									
\$\$\$	4												
OFF	123 5 7 8	345		4									
\$\$\$	4												
LABEL	123 5 7 8	345		4									
\$\$\$	4												
EQUIV	1234 6 8	3		4									
\$\$\$	4												
CHKPNT	1234 6 8	3		4									

MODAL AEROELASTIC RESPONSE

DMA Inst.	1	10	20	Bit Position 30	40	50	60
\$\$\$	4						
COND	1234	6 8	3	4			
\$\$\$	4						
SMA3	1	4					
\$\$\$	4						
CHKPNT	1	4					
\$\$\$	4						
ADD	1234	6 8	4				
\$\$\$	4						
CHKPNT	1234	6 8	4				
\$\$\$	4						
LABEL	1234	6 8	3	4			
\$\$\$	4						
GP4	1	9012	4				
SAVE	1	9012	4				
CHKPNT	1	9012	4				
GPSP	1234	6 890	3	4			
\$\$\$	4						
SAVE	1234	6 890	3	4			
\$\$\$	4						
COND	1234	6 890	3	4			
\$\$\$	4						
OFF	1234	6 890	3	4			
\$\$\$	4						
LABEL	1234	6 890	3				
\$\$\$	4						
EQUIV	123456789	34	4				
\$\$\$	4						
COND	123456789	34	4				
\$\$\$	4						
MCE1	1	9	4				
\$\$\$	4						
MCE2	123456789	34	4				
\$\$\$	4						
CHKPNT	123456789	34	4				
\$\$\$	4						
LABEL	123456789	34	4				
\$\$\$	4						
CHKPNT	1	9	4				
\$\$\$	4						
EQUIV	1234567890	34	4				
\$\$\$	4						
COND	1234567890	34	4				
\$\$\$	4						
SCE1	1234567890	34	4				
\$\$\$	4						
LABEL	1234567890	34	4				
\$\$\$	4						
CHKPNT	1234567890	34	4				
\$\$\$	4						
EQUIV	12345678901	34	4				
\$\$\$	4						
CHKPNT	12345678901	34	4				
\$\$\$	4						
PURGE	1234	6 8901	3	4			
\$\$\$	4						

RIGID FORMAT RESTART TABLES

DMAP Inst.	Bit Position						
	1	10	20	30	40	50	60
CHKPNT	1234	6 8901	3				
\$\$\$	4			4			
COND	1234	5678901	34				
\$\$\$	4			4			
PARAM							
\$\$\$	4						
VEC	1	901					
\$\$\$	4						
PARTN	1234	6 8901	3				
\$\$\$	4			4			
CHKPNT	1234	6 8901	3				
\$\$\$	4			4			
DECOMP	1234	6 8901	3				
\$\$\$	4			4			
SAVE	1234	6 8901	3				
\$\$\$	4			4			
COND	1234	6 8901	3				
\$\$\$	4			4			
JUMP	1234	6 8901	3				
\$\$\$	4			4			
LABEL	1234	6 8901	3				
\$\$\$	4						
PRTPARM	1234	6 8901	3				
\$\$\$	4			4			
LABEL	1234	6 8901	3				
\$\$\$	4			4			
CHKPNT	1234	6 8901	3				
\$\$\$	4			4			
FBS	1234	6 8901	3				
\$\$\$	4			4			
CHKPNT	1234	6 8901	3				
\$\$\$	4			4			
MPYAD	1234	6 8901	3				
\$\$\$	4			4			
CHKPNT	1234	6 8901	3				
\$\$\$	4			4			
SMP2	1234	5678901	34				
\$\$\$	4			4			
CHKPNT	1234	5678901	34				
\$\$\$	4			4			
LABEL	1234	5678901	34				
\$\$\$	4			4			
COND	1234	5678901234					
\$\$\$	4			4			
RBMG1	1234	5678901234					
\$\$\$	4			4			
CHKPNT	1234	5678901234					
\$\$\$	4			4			
RBMG2	1234	6 890123					
\$\$\$	4			4			
CHKPNT	1234	6 890123					
\$\$\$	4			4			
RBMG3	1234	6 890123					
\$\$\$	4			4			
CHKPNT	1234	6 890123					
\$\$\$	4			4			

MODAL AEROELASTIC RESPONSE

DMAP Inst.	1	10	20	Bit Position 30	40	50	60
RBMG4	12345678901234		4				
\$\$\$	4						
CHKPNT	12345678901234		4				
\$\$\$	4						
LABEL	12345678901234		4				
\$\$\$	4						
DPD	1	9012			0	0 23	6 8
SAVE	1	9012			0	0 23	6 8
COND	1	9012			0	0 23	6 8
\$\$\$	4						
EQUIV	1234 67 901 34		4				6
\$\$\$	4						
READ	12345678901234		4				89
\$\$\$	4						
SAVE	12345678901234		4				89
\$\$\$	4						
CHKPNT	12345678901234		4				89
\$\$\$	4						
OFF	12345678901234		4				89
\$\$\$	4						
SAVE	12345678901234		4				89
\$\$\$	4						
COND	12345678901234		4				89
\$\$\$	4						
OFF	12345678901234		4				89
\$\$\$	4						
SAVE	12345678901234		4				89
\$\$\$	4						
MTRXIN	1				0		67
\$\$\$	5						
SAVE	1				0		67
\$\$\$	5						
EQUIV	1	901			0		67
\$\$\$	4						
CHKPNT	1	901			0		67
\$\$\$	4						
GKAD	1234 6 8901 34		4		0		67
\$\$\$	4						
CHKPNT	1234 6 8901 34		4		0		67
\$\$\$	4						
GKAM	12345678901234		4		0		56789 2
\$\$\$	4						
SAVE	12345678901234		4		0		56789 2
\$\$\$	4						
CHKPNT	12345678901234		4		0		56789 2
\$\$\$	4						
APD	12	9012	6	4	9 2 45 7		6
SAVE	12	9012	6	4	9 2 45 7		6
PARAM			8 0				
\$\$\$	1						
COND			8				
\$\$\$	1						
PARAM			8				
\$\$\$	1						
PLTSET			8				
\$\$\$	1						

RIGID FORMAT RESTART TABLES

DMAP Inst.	1	10	20	<u>Bit Position</u>				40	50	60
SAVE			8							
\$\$\$	1									
PRTMSG			8							
\$\$\$	1									
COND			8							
\$\$\$	1									
PLOT			8							
\$\$\$	1									
SAVE			8							
\$\$\$	1									
PRTMSG			8							
\$\$\$	1									
LABEL			8							
\$\$\$	1									
GI	1234	6 8901 3		4		2 5 7				
\$\$\$	4									
CHKPNT	1234	6 8901 3		4		2 5 7				
\$\$\$	4									
PARAM				4	9	5 7				
\$\$\$		7								
AMG				4	9	45 7				
\$\$\$		7								
SAVE				4	9	45 7				
\$\$\$		7								
CHKPNT				4	9	45 7				
\$\$\$		7								
COND				6		7			6	
\$\$\$	4									
INPUTT2				6		7			6	
\$\$\$	4									
LABEL				6		7			6	
\$\$\$	4									
PARAM	12345678901234			4 6	9 2	5 7			6 89	2
\$\$\$	4									
AMP	12345678901234			4 6	9 2	45 7		8	6 89	2
\$\$\$	4									
SAVE	12345678901234			4 6	9 2	45 7		8	6 89	2
\$\$\$	4									
CHKPNT	12345678901234			4 6	9 2	45 7			6 89	2
\$\$\$	4									
FRLG	12345678901234			4				12	56 89	12
\$\$\$	4									
SAVE	1234567890123			4		890		12	56 89	12
\$\$\$	4									
CHKPNT	12345678901234			4		0		12	56 89	12
\$\$\$	4									
PARAM	12345678901234			4		0		12	56 89	12
\$\$\$	4									
PURGE	12345678901234			4		0		12	56 89	12
\$\$\$	4									
CHKPNT	12345678901234			4		0		12	56 89	12
\$\$\$	4									
GUST	12345678901234			4 6	9 2	4 7 0		9 12	56 89	12
\$\$\$	4									
SAVE	12345678901234			4 6	9 2	4 7 0		9 12	56 89	12
\$\$\$	4									

MODAL AEROELASTIC RESPONSE

DMAP Inst.	1	10	20	Bit Position 30				40	50	60					
EQUIV	12345678901234			4	6	9	2	4	7	0	9	12	56	89	12
\$\$\$	4														
CHKPNT	12345678901234			4	6	9	2	4	7	0	9	12	56	89	12
\$\$\$	4														
FRRD2	12345678901234			4	6	89	2	4	7	0	9	12	56	89	12
\$\$\$	4														
CHKPNT	12345678901234			4	6	89	2	4	7	0	9	12	56	89	12
\$\$\$	4														
EQUIV	12345678901234			4	6	89	2	4	7	0	9	12	56	89	12
\$\$\$	4														
CHKPNT	12345678901234			4	6	89	2	4	7	0	9	12	56	89	12
\$\$\$	4														
COND	12345678901234			4	6	89	2	4	7	0	9012	56	89012		
\$\$\$	4														
IFT	12345678901234			4	6789	2	4	7	0		9012	56	89012		
\$\$\$	4														
CHKPNT	12345678901234			4	6789	2	4	7	0		9012	56	89012		
\$\$\$	4														
LABEL	12345678901234			4	6789	2	4	7	0		9012	56	89012		
\$\$\$	4														
MODACC	12345678901234			4	6789	2	4	7	0		9012	56	89012		
\$\$\$	4														
ADR			1												
\$\$\$	4														
VDR			12												
\$\$\$	4														
SAVE			12												
\$\$\$	4														
COND			12												
\$\$\$	4														
SDR3			12												
\$\$\$	4														
QFP			1												
\$\$\$	4														
SAVE			1												
\$\$\$	4														
COND			2												
\$\$\$	4														
XYTRAN			2												
\$\$\$	4														
SAVE			2												
\$\$\$	4														
COND			2												
\$\$\$	4														
XYPLOT			2												
\$\$\$	4														
LABEL			2												
\$\$\$	4														
PARAM	12345678901234			4	6789	2	4	7	0		9012	56789012			
\$\$\$	4 6														
COND	12345678901234			4	6789	2	4	7	0		9012	56789012			
\$\$\$	4 6														
SDR1	12345678901234			4	6789	2	4	7	0		9012	56789012			
\$\$\$	4 6														
EQUIV	12345678901234			4	6789	2	4	7	0		9012	56789012			
\$\$\$	4 6														

RIGID FORMAT RESTART TABLES

DMAP Inst.	Bit Position									
	1	10	20	30	40	50	60			
COND	12345678901234			4 6789	2 4 7 0	9012	56789012			
\$\$\$	4 6									
VEC	12345678901234			4 6789	2 4 7 0	9012	56789012			
\$\$\$	4 6									
PARTN	12345678901234			4 6789	2 4 7 0	9012	56789012			
\$\$\$	4 6									
LABEL	12345678901234			4 6789	2 4 7 0	9012	56789012			
\$\$\$	4 6									
MPYAD	12345678901234			4 67890	2 4 7 0	9012	56789012			
\$\$\$	4 6									
UMERGE	12345678901234			4 6789	2 4 7 0	9012	56789012			
\$\$\$	4 6									
UMERGE	12345678901234			4 6789	2 4 7 0	9012	56789012			
\$\$\$	4 6									
CHKPNT	12345678901234			4 6789	2 4 7 0	9012	56789012			
\$\$\$	4 6									
UMERGE	12345678901234			4 6789	2 4 7 0	9012	56789012			
\$\$\$	4 6									
CHKPNT	12345678901234			4 6789	2 4 7 0	9012	56789012			
\$\$\$	4 6									
SDR2			90							
\$\$\$	4 6									
COND			90							
\$\$\$	4 6									
SDR2			90							
\$\$\$	4 6									
SDR3			90							
\$\$\$	4 6									
LABEL			90							
\$\$\$	4 6									
SDR3			90							
\$\$\$	4 6									
DDRMH			90							
\$\$\$	4 6									
OFF			90							
\$\$\$	4 6									
SAVE			90							
\$\$\$	4 6									
COND			8							
\$\$\$	1 4									
MPYAD			8							
\$\$\$	1 4									
SDR2			8							
\$\$\$	1 4									
PLOT			8							
\$\$\$	1 4									
PRTMSG			8							
\$\$\$	1 4									
LABEL			8							
\$\$\$	1 4									
COND			0							
\$\$\$	4 6									
XYTRAN			0							
\$\$\$	4 6									
SAVE			0							
\$\$\$	4 6									

MODAL AEROELASTIC RESPONSE

DMAP Inst.	1	10	20	Bit Position 30			40	50	60
COND			0						
\$\$\$	4	6							
XYPLOT			0						
\$\$\$	4	6							
LABEL			0						
\$\$\$	4	6							
COND			0						
\$\$\$	4	6							
COND			0					4	
\$\$\$	4	6							
RANDOM			0					4	
\$\$\$	4	6							
SAVE			0					4	
\$\$\$	4	6							
COND			0					4	
\$\$\$	4	6							
XYTRAN			0					4	
\$\$\$	4	6							
SAVE			0					4	
\$\$\$	4	6							
COND			0					4	
\$\$\$	4	6							
XYPLOT			0					4	
\$\$\$	4	6							
JUMP	1234567890123456	8901234	6	9	2	4567890		90123456789012	
LABEL	1234567890123456	8901234	6	9	2	4567890		90123456789012	
PRTPARM	1234567890123456	8901234	6	9	2	4567890		90123456789012	
LABEL	1234567890123456	8901234	6	9	2	4567890		90123456789012	
PRTPARM	1234567890123456	8901234	6	9	2	4567890		90123456789012	
LABEL	1234567890123456	8901234	6	9	2	4567890		90123456789012	
\$\$\$	4								
PRTPARM	1234567890123456	8901234	6	9	2	4567890		90123456789012	
\$\$\$	4								
LABEL	1234567890123456	8901234	6	9	2	4567890		90123456789012	
END	1234567890123456	8901234	6	9	2	4567890		90123456789012	

RIGID FORMAT RESTART TABLES

21.4 Rigid Format Change Restart Table

DMAP Inst.	63	Bit Position 70	80	DMAP Inst.	63	Bit Position 70	80
BEGIN		3456789012345678	01234567890123	CHKPNT			01234567890123
FILE		3456789012345678	01234567890123	PURGE			01234567890123
GP1			01234567890123	CHKPNT			01234567890123
SAVE			01234567890123	COND			01234567890123
COND	3456789012345678		01234567890123	PARAM			01234567890123
GP2			01234567890123	VEC			01234567890123
PARAML			01234567890123	PARTN			01234567890123
PARAML			01234567890123	CHKPNT			01234567890123
PARAM			01234567890123	DECOMP			01234567890123
PARAM			01234567890123	SAVE			01234567890123
GP3			01234567890123	COND			01234567890123
TA1			01234567890123	JUMP			01234567890123
SAVE			01234567890123	LABEL			01234567890123
COND	3456789012345678		01234567890123	PRTPARM			01234567890123
PARAM			01234567890123	LABEL			01234567890123
PARAM			01234567890123	CHKPNT			01234567890123
EMG			01234567890123	FBS			01234567890123
SAVE			01234567890123	CHKPNT			01234567890123
COND			01234567890123	MPYAD			01234567890123
EMA			01234567890123	CHKPNT			01234567890123
CHKPNT			01234567890123	SMP2			01234567890123
LABEL			01234567890123	CHKPNT			01234567890123
COND			01234567890123	LABEL			01234567890123
EMA			01234567890123	COND			01234567890123
CHKPNT			01234567890123	RBMG1			01234567890123
COND			01234567890123	CHKPNT			01234567890123
GPWG			01234567890123	RBMG2			01234567890123
OFF			01234567890123	CHKPNT			01234567890123
LABEL			01234567890123	RBMG3			01234567890123
EQUIV			01234567890123	CHKPNT			01234567890123
CHKPNT			01234567890123	RBMG4			01234567890123
COND			01234567890123	CHKPNT			01234567890123
SMA3			01234567890123	LABEL			01234567890123
CHKPNT			01234567890123	DPD			01234567890123
ADD			01234567890123	SAVE			01234567890123
CHKPNT			01234567890123	COND	3456789012345678		01234567890123
LABEL			01234567890123	EQUIV			01234567890123
GP4			01234567890123	READ			01234567890123
SAVE			01234567890123	SAVE			01234567890123
CHKPNT			01234567890123	CHKPNT			01234567890123
GPSP			01234567890123	OFF			01234567890123
SAVE			01234567890123	SAVE			01234567890123
COND			01234567890123	COND	3456789012345678		01234567890123
OFF			01234567890123	OFF			01234567890123
LABEL			01234567890123	SAVE			01234567890123
EQUIV			01234567890123	MTRXIN			01234567890123
COND			01234567890123	SAVE			01234567890123
MCE1			01234567890123	EQUIV			01234567890123
MCE2			01234567890123	CHKPNT			01234567890123
CHKPNT			01234567890123	GKAD			01234567890123
LABEL			01234567890123	CHKPNT			01234567890123
CHKPNT			01234567890123	GKAM			01234567890123
EQUIV			01234567890123	SAVE			01234567890123
COND			01234567890123	CHKPNT			01234567890123
SCE1			01234567890123	APD			01234567890123
LABEL			01234567890123	SAVE			01234567890123
CHKPNT			01234567890123	PARAM			01234567890123
EQUIV			01234567890123	COND			01234567890123

MODAL AEROELASTIC RESPONSE

DMAP Inst.	63	Bit Position 70	80	DMAP Inst.	63	Bit Position 70	80
PARAM			01234567890123	COND			01234567890123
PLTSET			01234567890123	VEC			01234567890123
SAVE			01234567890123	PARTN			01234567890123
PRTMSG			01234567890123	LABEL			01234567890123
COND			01234567890123	MPYAD			01234567890123
PLOT			01234567890123	UMERGE			01234567890123
SAVE			01234567890123	UMERGE			01234567890123
PRTMSG			01234567890123	CHKPNT			01234567890123
LABEL			01234567890123	UMERGE			01234567890123
GI			01234567890123	CHKPNT			
CHKPNT			01234567890123	SDR2			
PARAM	3456789012345678		01234567890123	COND			
AMG			01234567890123	SDR2			
SAVE			01234567890123	SDR3			
CHKPNT			01234567890123	LABEL			
COND			01234567890123	SDR3			
INPUTT2			01234567890123	DDRMH			
LABEL			01234567890123	OFF			
PARAM			01234567890123	SAVE			
AMP			01234567890123	COND			
SAVE			01234567890123	MPYAD			
CHKPNT			01234567890123	SDR2			
FRLG	0	3	01234567890123	PLOT			
SAVE	0	3	01234567890123	PRTMSG			
CHKPNT	0	3	01234567890123	LABEL			
PARAM			01234567890123	COND			
PURGE			01234567890123	XYTRAN			
CHKPNT			01234567890123	SAVE			
GUST			01234567890123	COND			
SAVE			01234567890123	XYPLOT			
EQUIV			01234567890123	LABEL			
CHKPNT			01234567890123	COND			
FRRD2	0	3	01234567890123	COND			
CHKPNT	0	3	01234567890123	RANDOM			
EQUIV	1	4	01234567890123	SAVE			
CHKPNT	1	4	01234567890123	COND			
COND	1	4	01234567890123	XYTRAN			
IFT	1	4	01234567890123	SAVE			
CHKPNT	1	4	01234567890123	COND			
LABEL	1	4	01234567890123	XYPLOT			
MODACC			01234567890123	JUMP	3456789012345678		01234567890123
ADR				LABEL	3456789012345678		01234567890123
VDR				PRTPARM	3456789012345678		01234567890123
SAVE				LABEL	3456789012345678		01234567890123
COND				PRTPARM	3456789012345678		01234567890123
SDR3				LABEL	3456789012345678		01234567890123
OFF				PRTPARM	3456789012345678		01234567890123
SAVE				LABEL	3456789012345678		01234567890123
COND				END	3456789012345678		01234567890123
XYTRAN							
SAVE							
COND							
XYPLOT							
LABEL							
PARAM			01234567890123				
COND			01234567890123				
SDR1			01234567890123				
EQUIV			01234567890123				

RIGID FORMAT RESTART TABLES

7.21.5 File Name Restart Table

DMAP Inst.	94	Bit Position		120
		100	110	
BEGIN				
FILE				7 8
GP1	4			
SAVE	4			
COND	4			
GP2	5			
PARAML				2 5
PARAML				
PARAM				
PARAM				
GP3	6			
TA1	7			
SAVE	7			
COND	7			
PAPAM	8			
PARAM	9			
EMG	89			
SAVE	89			
COND	8			
EMA	8			
CHKPNT	8			
LABEL	8			
COND	9			
EMA	9			
CHKPNT	9			
COND				4
GPWG				4
OFF				4
LABEL				4
EQUIV	0			
CHKPNT	0			
COND	0			
SMA3	0			
CHKPNT	0			
ADD	0			
CHKPNT	0			
LABEL	0			
GP4	1			
SAVE	1			
CHKPNT	1			
GPSP	2			
SAVE	2			
COND	2			
OFF	2			
LABEL	2			
EQUIV	4			
COND	34			
MCE1	3			
MCE2	4			
CHKPNT	4			
LABEL	34			
CHKPNT	3			
EQUIV	5			
COND	5			
SCE1	5			
LABEL	5			
CHKPNT	5			

MODAL AEROELASTIC RESPONSE

DMAP Inst.	94	Bit Position		120		
		100	110			
EQUIV		6		3		
CHKPNT		6		3		
PURGE		6				
CHKPNT		6				
COND		6	3	3		
PARAM		6			0	
VEC						3
PARTN						2
CHKPNT						2
DECOMP						1
SAVE						1
COND						1
JUMP						1
LABEL						1
PRTPARM						1
LABEL						1
CHKPNT						1
FBS			3			
CHKPNT			3			
MPYAD		6				
CHKPNT		6				
SMP2				3		
CHKPNT				3		
LABEL		6	3	3		
COND		7890		3		
RBMG1		7				
CHKPNT		7				
RBMG2		8				
CHKPNT		8				
RBMG3		9				
CHKPNT		9				
RBMG4		0				
CHKPNT		0				
LABEL		7890				
DPD			1			
SAVE			1			
COND			1			
EQUIV				5		
READ			2			
SAVE			2			
CHKPNT			2			
OFF			2			
SAVE			2			
COND			2			
OFF			2			
SAVE			2			
MTRXIN				4		
SAVE				4		
EQUIV				4		
CHKPNT				4		
GKAD				5		
CHKPNT				5		
GKAM				6		
SAVE				6		
CHKPNT				6		
APD					4	
SAVE					4	

RIGID FORMAT RESTART TABLES

DMAP	94	Bit Position		120		
Inst.	100	110				
PARAM					5	
COND			5			
PARAM					5	
PLTSET			5			
SAVE			5			
PRTMSG			5			
COND			5			
PLOT						0
SAVE						0
PRTMSG						0
LABEL			5			
GI			6			
CHKPNT			6			
PARAM					7	
AMG			7			
SAVE			7			
CHKPNT			7			
COND			8			
INPUTT2			8			
LABEL			8			
PARAM					8	
AMP					8	
SAVE					8	
CHKPNT					8	
FRLG					9	
SAVE				9		
CHKPNT				9		
PARAM				9		
PURGE				9		
CHKPNT				9		
GUST				0		
SAVE				0		
EQUIV				0		
CHKPNT				0		
FRRD2				1		
CHKPNT				1		
EQUIV				2		
CHKPNT				2		
COND				2		
IFT				2		
CHKPNT				2		
LABEL				2		
MODACC				3		
ADR				4		
VOR					5	
SAVE					5	
COND					5	
SDR3					5	
OFF					5	
SAVE					5	
COND					6	
XYTRAN					6	
SAVE					6	
COND					6	
XYPLOT					6	
LABEL					56	
PARAM					7	
COND					7	

MODAL AEROELASTIC RESPONSE

DMAP		Bit Position	
Inst.	94	100	110 120
SDR1			7
EQUIV			0
COND			0
VEC			0
PARTN			0
LABEL			0
MPYAD			0
UMERGE			0
UMERGE			0
CHKPNT			0
UMERGE			0
CHKPNT			0
SDR2			1
COND			1
SDR2			1
SDR3			2
LABEL			2
SDR3			2
DORMM			3
OFF			3
SAVE			3
COND			4
MPYAD			4
SDR2			4
PLOT			4
PRTHSG			4
LABEL			4
COND			5
XYTRAN			5
SAVE			5
COND			5
XYPLOT			5
LABEL			5
COND			5
COND			6
RANDOM			6
SAVE			6
COND			6
XYTRAN			6
SAVE			6
COND			6
XYPLOT			6
JUMP			6
LABEL		1	
PRTPARM			
LABEL	7 9		
PRTPARM	7 9		
LABEL		4	
PRTPARM		4	
LABEL			7
END			6

8. STRUCTURAL ELEMENT DESCRIPTIONS

8.1 INTRODUCTION TO STRUCTURAL ELEMENT DESCRIPTIONS

The finite structural element subroutines used in NASTRAN have a number of different calculations associated with them. These subroutines are found in the modules SMA1, SMA2, SSG1, SDR2, DSMG1, PLA3 and PLA4.

All modules excluding IFP having anything to do with the NASTRAN structural elements, their geometry, or associated data blocks, use the basic element data found in common block /GPTA1/. /GPTA1/ is set in its own block data subprogram, and/or by (in the presence of dummy-user-elements) the routine DELSET. Refer to Section 2.5.2.1 for further information regarding /GPTA1/.

The element subroutines in the SMA1 (Structural Matrix Assembler - Phase 1) module generate element stiffness matrix partitions. The stiffness matrix, $[K]$, for a structural element consists of a 6 by 6 partition for each combination of the connected grid points. For example, a RØD element is connected to two grid points, "a" and "b". The stiffness matrix partitions are: $[K_{aa}]$, $[K_{ab}]$, $[K_{ba}]$ and $[K_{bb}]$. A triangular element (e.g., TRMEM) is connected to three points. It will generate nine partitions: $[K_{aa}]$, $[K_{ab}]$, $[K_{ac}]$, $[K_{ba}]$, $[K_{bb}]$, $[K_{bc}]$, $[K_{ca}]$, $[K_{cb}]$ and $[K_{cc}]$. In order to generate a particular partition, $[K_{ij}]$, it is often necessary to generate $[K]$. However, only those partitions $[K_{ij}]$, where i is the pivot point (see section 1.8) and $j = 1, 2, \dots, n$ (n being the number of grid points associated with the element), are output by an element stiffness matrix generation subroutine, e.g., KRØD. These partitions are output from an element subroutine in the form of calls to the "insertion" subroutine SMA1B (see Section 4.27). There is one call for each 6 by 6 partition if the element is a structural element, and one call for each 1 by 1 "partition" if the element is a scalar element. The unused partitions are recalculated and used when $j \neq i$ appears as a pivot point in a subsequent ECPT record. An alternate procedure for matrix generation, which is not used, would be to calculate all of the element matrices once and store them on an auxiliary storage unit for use when needed. The alternate procedure is less efficient for large problems, where efficiency really counts, because the recalculation time is less than the time required to recover element matrices from the auxiliary unit.

Element structural damping matrices, $[K^4]$, are proportional to the element stiffness matrices, the proportionality constant being g_e , the structural damping coefficient input on a material (e.g., MAT1) bulk data card. An element stiffness matrix generation routine, e.g., KRØD, of module SMA1 will output, through the calling sequence to subroutine SMA1B: 1) an element stiffness

STRUCTURAL ELEMENT DESCRIPTIONS

matrix partition, 2) the structural damping coefficient, and 3) a flag, which will signal SMA1B that the scalar multiplication of the matrix by the structural damping coefficient is to take place.

The element subroutines (e.g., MRØD, MCØNMX) in the SMA2 (Structural Matrix Assembler - Phase 2) module generate element mass matrix partitions. The remarks in the third paragraph above concerning element stiffness matrix partitions apply here also when the reader makes the substitutions: "mass" for "stiffness", "[M]" for "[K]", "MRØD" for "KRØD", and "SMA2B" for "SMA1B".

Only the element VISC and DAMPi generate viscous damping terms which contribute to the damping matrix, $[B_{gg}]$, and conversely, the only elements which contribute to $[B_{gg}]$ are the VISC and DAMPi elements. These terms are calculated in module SMA2. The damping matrix partitions are passed to subroutine SMA2B in a fashion similar to that for mass matrix partitions.

Element static loading functions due to temperature and enforced deformations are generated in the SSG1 (Static Solution Generator - Phase 1) module, and the mathematical descriptions for these functions are given in this Section (8). (See the Module Functional Description for SSG1, Section 4.41, for the equations governing direct applied loads and gravity loads.) The output of an element routine are load vectors which are placed in the $\{P_g\}$ load vector (see Section 4.41).

Element stresses and forces due to displacements are calculated in the SDR2 (Stress Data Recovery - Phase 2) module. These calculations are performed in two phases. Phase 1 generates element stress matrices for each element for which the user has requested element stress and/or force output. These element stress matrices are written on a scratch file for use in phase 2. In phase 2, the displacement vector for the current subcase is read into core, and, for each element for which stress and/or force output is requested, the corresponding element stress matrix is read and passed to the phase 2 element subroutine. The phase 2 element subroutine then calculates element stresses and forces. A list of the stresses and forces output in phase 2 for each element is given in Sections 2.3.51 and 2.3.52 respectively.

Differential stiffness matrix partitions are calculated for some elements. These are calculated in module DSMG1 (Differential Stiffness Matrix Generator - Phase 1) for large displacement analysis and buckling problems. The output of an element routine of the DSMG1 module are the 6 by 6 differential stiffness matrix partitions, $[K_{ij}^d]$, where i is the pivot point. The "insert" subroutine for module DSMG1, similar to subroutine SMA1B of module SMA1, is DS1b.

Nonlinear, plastic effects in the structure may be determined by solving for the element stress and modifying the elastic properties of an element in an iterative loop. Element stresses are calculated in the PLA3 (Piecewise Linear Analysis - Phase 3) module, and element stiffness matrices with modified elastic properties are calculated in the PLA4 (Piecewise Linear Analysis - Phase 4). The outputs of an element subroutine of the PLA3 module are: 1) element stresses, which have the same formats as the element stresses output from a phase 2 element subroutine of module SDR2, and 2) updated incremental stress data in the ESTNL1 data block, which are used as input to the PLA3 module in the next pass of the Piecewise Linear Analysis (PLA) Rigid Format DMAP loop. The outputs of an element subroutine of the PLA4 module are: 1) element stiffness matrix partitions (the remarks on element stiffness matrix partitions in the second paragraph apply here as well, except that the "insertion" subroutine is PLA4B) and 2) updated incremental stress data in the ECPTNL1 data block, which are used as input to the PLA4 module in the next pass of the PLA Rigid Format DMAP loop.

The following data are needed to generate the element matrices in the above modules.

1. Element Connection and Properties Table (ECPT) Data.
2. Transformation matrices, $[T_i]$, from the global coordinate system to the basic coordinate system.
3. Material Property Data.
4. Element Deformation Data (used only in modules SSG1, SDR2 and DSGM1).
5. Grid Point Temperature Data (used only in modules SSG1, SDR2 and DSGM1).

The ECPT data are input to an element subroutine by a module driver from the ECPT data block or the EST (Element Summary Table) data block. The data in each of these data blocks are identical, from an individual element subroutine point of view. The ECPT data block is used in modules SMA1, SMA2 and DSGM1; the EST data block is used in modules SSG1 and SDR2. For the special case of Piecewise Linear Analysis, the ECPTNL (Element Connection and Properties Table for Nonlinear Elements) data block is used in module PLA4, and the ESTNL (Element Summary Table for Nonlinear Elements) data block is used in module PLA3. The ECPT and EST data blocks are generated in the Table Assembler (TA1) module (see Section 4.26) from the following data blocks: ECT (Element Connection Table, Section 2.3.4.1), EPT (Element Property Table, Section 2.3.2.5), BGPDT (Basic Grid Point Definition Table, Section 2.3.3.5), and GPTT (Grid Point Temperature Table, Section 2.3.7.2).

The ECPT data for an element consist of four separate parts: 1) connection data 2) property data, 3) basic grid point definition data and 4) the element temperature for material properties. The connection data consists of data on a connection bulk data card (e.g., CRØD), except for the property identification number (the property identification number on the connection and property cards is used only to relate the two cards during the assembly of the ECPT and EST data blocks, and it does not appear in either the connection data or property data). Note also that grid point identification numbers have been converted to internal numbers, Scalar Index List (SIL) numbers, which correspond to degrees of freedom numbers. Property data consist of data on a property bulk data card (e.g., PRØD) with the above noted exception. Basic grid point definition data consist of, for each grid point connecting the element, 1) the identification number of the coordinate system in which displacements are defined at the grid point and 2) the coordinates of the grid point in the basic coordinate system. The element temperature for material properties is the average value given for each element in the GPTT data block. This temperature is placed in the ECPT/EST data in the Table Assembler module from the element temperatures in the GPTT data block and the set identification number, n , from the TEMPERATURE(MATERIAL) = n card in the User's Case Control Deck. Note that n is transmitted to the Table Assembler via the tenth word of /SYSTEM/ (see Section 2.4.1.8).

The transformation matrices, $[T_i]$, from the global coordinate system to the basic coordinate system, are supplied to an element subroutine by the utility routines TRANSD and TRANSS. These utility routines use the CSTM (Coordinate System Transformation Matrices, Section 2.3.3.4) data block in conjunction with the basic grid point definition data at a point i to compute $[T_i]$. Hence all modules which deal with element calculations require the CSTM data block as input. TRANSD returns, to an element subroutine, a double precision matrix, $[T_i]$, used by element routines in the following modules which use double precision arithmetic: SMA1, SMA2, DSGM1 and PLA4; TRANSS returns, to an element subroutine, a single precision matrix, $[T_i]$, used by element routines in the following modules which use single precision arithmetic: SSG1, SDR2 and PLA3.

Material property data are contained in the MPT (Material Properties Table, Section 2.3.2.6) and the DIT (Direct Input Tables, Section 2.3.2.7) data blocks. Both of these data blocks are output from the IFP (Input File Processor) Preface module. The utility routine MAT (see Section 3.4.36) fetches required material property data for element routines. These data are returned in single precision form.

STRUCTURAL ELEMENT DESCRIPTIONS

Element deformation data are contained in the EDT (Element Deformation Table, Section 2.3.2.8) data block which is output by the IFP Preface Module. Element deformation data is admissible only for the RØD (including CØNRØD), TUBE and BAR elements.

Element temperature data are contained in the GPTT (Grid Point Temperature Table, Section 2.3.7.2), which is output by the GP3 (Geometry Processor - Phase 3) module. The temperature data contained in this data block are used for static loading functions due to temperature.

Table 1 on the following page gives reference to the Theoretical and User's Manuals where more information on the elements can be found.

Table 2 summarizes the element types. The following overall conventions were used:

- No capability exists.
- * Capability Exists.
- u User defined capability.

Column conventions are defined as follows:

<u>General</u>		
<u>Column</u>	<u>Symbol</u>	<u>Meaning</u>
Neumonic		The BCD name that appears in field one of the Bulk Data Deck card.
Id		The integer number that EMG prints. This is the same as the /GPTA1/ element number.
Plot		The BCD element symbol for structure plots.
C	G F S R _A R _F A	GRID, GRIDB and in some cases, SPØINT connectivity. GRIDF connectivity. GRIDS connectivity. RINGAX connectivity. RINGFL connectivity. Aerodynamic point connectivity.
PM	P 1,2,3,4 or 5	Element has a property card. Type of MAT1 card(s) allowed.
So	*	If stress optimization is allowed.
FØRM	- I A H A _d F M P R	No special formulation of the matrices Isoparametric Axisymmetric element (and loads if H not specified) Harmonic coefficients Aerodynamic Axisymmetric fluid/acoustic Direct matrix input Plottable only Rigid element
RF	1 to 15 H1,H3,H9 A10	Displacement rigid formats that support the element Heat Aero

STRUCTURAL ELEMENT DESCRIPTIONS

Matrices

<u>Columns</u>	<u>Symbol</u>	<u>Meaning</u>
K	K	Stiffness excluding heat transfer
	D	Differential stiffness
	P	Piecewise linear stiffness
	1,2,...6	Maximum number of nonzero degrees of freedom.
M	L	Lumped mass
	C	Coupled mass
	1	Scalar mass
B	B	Damping element
	1 or 2	Maximum number of nonzero degrees of freedom.
K4	*	K ⁴ structural damping in direct formulation analysis.
H _{cc}	*	Heat conduction/convection
H _c	*	Heat capacity
H _R	*	Heat radiation
K ₂	*	Direct matrix input or equivalent

Loading

<u>Columns</u>	<u>Symbol</u>	<u>Meaning</u>
T	*	Static thermal by TEMP, TEMPAC only
	I	Inplane static thermal loads
	B	Bending static thermal loads
P	*	Pressure
G	*	Gravity
ED	*	Element deformation
H	*	Heat analysis - thermal
A	*	Aerodynamic

Output

<u>Column</u>	<u>Symbol</u>	<u>Meaning</u>
F _r		Integer number of real-force entries
F _c		Integer number of complex-force entries
S _R		Integer number of real-stress entries
S _c		Integer number of complex-stress entries
R	*	Random analysis
H	*	Heat analysis

STRUCTURAL ELEMENT DESCRIPTIONS

Table 1. Structural Element References

Bulk Data Connection Card Mnemonic	Programmer's Manual Reference	User's Manual Reference	Theoretical Manual Reference
CAXIF2	8.15	1.8	17.1
CAXIF3	8.15	1.8	17.1
CAXIF4	8.15	1.8	17.1
CBAR	8.2	1.3.2	5.2, 7.2
CCONEAX	8.9	1.3.6	5.9
CDAMP1	8.7	1.3.8	5.6
CDAMP2	8.7	1.3.8	5.6
CDAMP3	8.7	1.3.8	5.6
CDAMP4	8.7	1.3.8	5.6
CELAS1	8.7	1.3.8	5.6
CELAS2	8.7	1.3.8	5.6
CELAS3	8.7	1.3.8	5.6
CELAS4	8.7	1.3.8	5.6
CFLUID2	8.15	1.7	16.1
CFLUID3	8.15	1.7	16.1
CFLUID4	8.15	1.7	16.1
CHBDY	8.18		8.2
CHEXA1	8.17	1.3.9	5.12
CHEXA2	8.17	1.3.9	5.12
CIHEX1	8.21		
CIHEX2	8.21		
CIHEX3	8.21		
CMASS1	8.7	1.3.8	5.6
CMASS2	8.7	1.3.8	5.6
CMASS3	8.7	1.3.8	5.6
CMASS4	8.7	1.3.8	5.6
CMFREE	8.15	1.7	16.1
CØNM1	8.8	1.2.3	5.5
CØNM2	8.8	1.2.3	5.5
CØNRØD	8.27	1.3.3	5.2, 7.2
CQDMEM	8.4	1.3.5	5.8, 7.3
CQDMEM1	8.19	1.3.5	5.8
CQDMEM2	8.20	1.3.5	5.8
CQDPLT	8.5	1.3.5	5.8
CQUAD1	8.6	1.3.5	5.8, 7.3
CQUAD2	8.6	1.3.5	5.8, 7.3
CRØD	8.27	1.3.3	5.2, 7.2
CSHEAR	8.3	1.3.4	5.3
CSLØT3	8.16	1.8	17.1
CSLØT4	8.16	1.8	17.1
CTETRA	8.17	1.3.9	5.12
CTØRDRG	8.12	1.3.7	5.11
CTRAPAX	8.22	1.3.9	5.11
CTRAPRG	8.11	1.3.7	5.10
CTRBSC	8.5	1.3.5	5.8
CTRIA1	8.6	1.3.5	5.8, 7.3
CTRIA2	8.6	1.3.5	5.8, 7.3
CTRIAAX	8.23	1.3.9	5.11
CTRIARG	8.10	1.3.7	5.8

STRUCTURAL ELEMENT DESCRIPTIONS

Table 1 (Cont.) Structural Element References

<u>Bulk Data Connection Card Mnemonic</u>	<u>Programmer's Manual Reference</u>	<u>User's Manual Reference</u>	<u>Theoretical Manual Reference</u>
CTRMEM	8.4	1.3.5	5.8, 7.3
CTRPLT	8.5	1.3.5	5.8
CTUBE	8.1	1.3.3	5.2, 7.2
CTWIST	8.3	1.3.4	5.3
CVISC	8.13	1.3.3	5.2
CWEDGE	8.17	1.3.9	5.12

Note: The bulk data connection and property card descriptions in Section 2 of the NASTRAN User's Manual should also be consulted.

Table 2. Summary of Element Types

General Description								Matrices								Loading						Output						
Neumonic	Id	Plot	C	PH	So	Form	RF	K	H	B	K4	H _{CC}	H _C	H _R	K2	T	P	G	ED	H	A	F _R	F _C	S _R	S _C	R	H	
BDYLIST	12	-	Rc	-	-	II	1,3,7-9	K1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CAERØ1	72	AE	GA	P	-	Ad	A10	-	-	-	-	-	-	-	*	-	-	-	-	-	*	-	-	-	-	-	-	-
CAXIF2	47	A2	F	-	-	HF _A	1,3,7-9	K1	C	-	-	-	-	-	-	-	-	*	-	-	-	-	-	-	5	9	-	-
CAXIF3	48	A3	F	-	-	HF _A	1,3,7-9	K1	C	-	-	-	-	-	-	-	-	*	-	-	-	-	-	-	5	19	-	-
CAXIF4	49	A4	F	-	-	HF _A	1,3,7-9	K1	C	-	-	-	-	-	-	-	-	*	-	-	-	-	-	-	5	19	-	-
CBAR	34	BR	G	P1	*	-	ALL	KDP6	LC	-	*	*	*	-	-	1B	-	*	*	*	-	-	9	17	16	19	*	*
CCØNEAX	35	CN	R _A	P1	-	HA	ALL BUT 6,15	KD5	L	-	*	-	-	-	-	I	*	*	-	-	-	7	-	18	-	-	-	-
CDAMP1	20	-	G	P	-	-	19,15,7-9	-	-	B1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CDAMP2	21	-	G	-	-	-	14,15,7-9	-	-	B1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CDAMP3	22	-	G	P	-	-	14,15,7-9	-	-	B1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CDAMP4	23	-	G	-	-	-	14,15,7-9	-	-	B1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CDUM1	53-61	D1	G	U	-	-	ALL	U	U	U	U	U	U	-	-	-	-	U	-	-	-	10	9	10	9	-	-	-
CELAS1	11	-	G	P	-	-	1-3,7-12,14,15	K1	-	-	*	-	-	-	-	-	-	-	-	-	-	2	3	2	3	*	-	-
CELAS2	12	-	G	-	-	-	1-3,7-12,14,15	K1	-	-	-	-	-	-	-	-	-	-	-	-	-	2	3	2	3	*	-	-
CELAS3	13	-	G	P	-	-	1-3,7-12,14,15	K1	-	-	*	-	-	-	-	-	-	-	-	-	-	2	3	2	3	*	-	-

STRUCTURAL ELEMENT DESCRIPTIONS

Table 2. Summary of Element Types (continued)

General Description								Matrices								Loading						Output					
Neumonic	Id	Plot	C	PM	So	Form	RF	K	M	B	K4	H _{CC}	H _C	H _R	K2	T	P	G	ED	H	A	F _R	F _C	S _R	S _C	R	H
CELAS4	14	-	G	-	-	-	1-3,7-12, 14,15	K1	-	-	-	-	-	-	-	-	-	-	-	-	-	2	3	0	3	*	-
CFLUID2	43	F2	R _f	-	-	F	3,7-9	K1	C	-	-	-	-	-	-	-	-	*	-	-	-	-	-	-	-	-	-
CFLUID3	44	F3	R _f	-	-	F	3,7-9	K1	C	-	-	-	-	-	-	-	-	*	-	-	-	-	-	-	-	-	-
CFLUID4	45	F4	R _C	-	-	F	3,7-9	K1	C	-	-	-	-	-	-	-	-	*	-	-	-	-	-	-	-	-	-
CHBDY	52	HB	G	P4	-	-	H1-H9	-	-	-	-	*	*	*	-	-	-	-	-	-	-	-	-	-	-	-	*
CHEXA1	41	H1	G	145	-	-	1-3,7-12	K3	L	-	-	*	*	-	-	*	-	*	-	*	-	-	-	9	13	-	*
CHEXA2	42	H2	G	145	-	-	1-3,7-12 14,15	K3	L	-	-	*	*	-	-	*	-	*	-	*	-	-	-	9	13	-	*
CIHEX1	65	XL	G	P1245	-	I	A11 but 6	K3	C	-	*	*	*	-	-	*	*	*	-	*	-	-	-	22	*	-	*
CIHEX2	66	XG	G	P1245	-	I	A11 but 6	DK3	C	-	*	*	*	-	-	*	*	*	-	*	-	-	-	22	*	-	*
CIHEX3	67	XC	G	P1245	-	I	A11 but 6	DK3	C	-	*	*	*	-	-	*	*	*	-	*	-	-	-	22	*	-	*
CMASS1	25	-	G	P	-	-	A11	-	L1	-	-	-	-	-	-	-	-	*	-	-	-	-	-	-	-	-	-
CMASS2	26	-	G	-	-	-	A11	-	L1	-	-	-	-	-	-	-	-	*	-	-	-	-	-	-	-	-	-
CMASS3	27	-	G	P	-	-	A11	-	L1	-	-	-	-	-	-	-	-	*	-	-	-	-	-	-	-	-	-
CMASS4	28	-	G	-	-	-	A11	-	L1	-	-	-	-	-	-	-	-	*	-	-	-	-	-	-	-	-	-
CQNM1	29	-	G	-	-	-	A11	-	L1	-	-	-	-	-	-	-	-	*	-	-	-	-	-	-	-	-	-

STRUCTURAL ELEMENT DESCRIPTIONS

Table 2. Summary of Element Types (continued)

General Description								Matrices								Loading						Output						
Neumonic	Id	Plot	C	PM	So	Form	RF	K	M	B	K4	H _{CC}	H _C	H _R	K2	T	P	G	ED	H	A	F _R	F _C	S _R	S _C	R	H	
CØNM2	30	-	G	-	-	-	A11	-	LC	-	-	-	-	-	-	-	-	*	-	-	-	-	-	-	-	-	-	-
CØNRØD	10	CR	G	145	-	-	A11	KDP2	LC	-	*	*	*	-	-	*	-	*	*	*	-	3	5	5	5	*	*	*
CQDNEM	16	QM	G	P1245	*	-	A11	KDP2	L	-	*	*	*	-	-	I	*	*	-	*	-	-	-	8	7	*	*	*
CQDMEM1	62	M1	G	P	-	I	14,15 1-3,7-12	K3	L	-	*	-	-	-	-	I	*	*	-	-	-	0	0	8	7	-	-	-
CQDMEM2	63	M2	G	P12	-	-	1-3,7-12 14,15	KDP2	L	-	*	-	-	-	-	I	*	*	-	-	-	17	33	8	7	-	-	-
CQDPLT	15	QP	G	P12	*	-	14,15 1-3,7-12	K3	LC	-	*	-	-	-	-	B	*	*	-	-	-	6	11	17	15	*	-	-
CQUAD1	19	Q1	G	P1245	*	-	A11	KDP5	LC	-	*	*	*	-	-	IB	*	*	-	*	-	6	11	17	15	*	*	*
CQUAD2	18	Q2	G	P1245	*	-	A11	KDP5	LC	-	*	*	*	-	-	IB	*	*	-	*	-	6	11	17	15	*	*	*
CRIGD1	-	-	G	-	-	R	A11	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
CRIGD2	-	-	G	-	-	R	A11	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
CRØD	1	RD	G	P1245	*	-	A11	KDP2	LC	-	*	*	*	-	-	*	-	*	*	*	-	3	5	5	5	*	*	*
CSHEAR	4	SH	G	P1	*	-	A11	K2	L	-	*	-	-	-	-	-	*	*	-	-	-	17	33	4	5	*	-	-
CSLØT	50	S3	S	-	-	F	1,3,7-9	K1	C	-	-	-	-	-	-	-	-	-	-	-	-	-	-	6	11	-	-	-
CSLØT4	51	S4	S	-	-	F	1,3,7-9	K1	C	-	-	-	-	-	-	-	-	-	-	-	-	-	-	7	13	-	-	-
CTETRA	39	TE	G	1	-	-	1,3,7-12 14,15	K3	L	-	*	*	*	-	-	*	-	*	-	*	-	-	-	9	13	-	*	*
CTØDRG	38	TR	G	P13	-	A	1-3,7-12	K5	C	-	*	-	-	-	-	*	-	*	-	-	-	13	-	16	-	-	-	-
CTRAPAX	71	T4	R _A	P13	-	HA	1,3,7-12	K3	LC	-	*	-	-	-	-	*	*	*	-	-	-	14	0	32	2			

STRUCTURAL ELEMENT DESCRIPTIONS

Table 2. Summary of Element Types (continued)

General Description								Matrices								Loading					Output				
Neumonic	Id	Plot	C	PM	So	Form	RF	K	M	B	K4	H _{CC}	H _C	H _R	K ₂	T	P	G	ED	H	F _r	F _c	S _R	S _c	R
CTRAPRG	37	TA	G	1345	-	A	1-3,7-12	K2	C	-	*	*	*	-	-	*	-	*	-	*	13	-	21	-	-
CTRBSC	7	TB	G	P12	*	-	1-3,7-12 14,15	K3	LC	-	*	-	-	-	-	B	*	*	-	-	6	11	17	15	*
CTRIAAX	70	T3	R _A	P13	-	HA	1-3,7-12	K3	LC	-	*	-	-	-	-	*	*	*	-	-	11	-	8	-	-
CTRIA1	6	T1	G	P1245	*	-	A11	KDP5	LC	-	*	*	*	-	-	IB	*	*	-	*	6	11	17	15	*
CTRIA2	17	T2	G	P1245	*	-	A11	KDP5	LC	-	*	*	*	-	-	IB	*	*	-	*	6	11	17	15	*
CTRIARG	36	TI	G	1345	-	A	1-3,7-12	K2	C	-	*	*	*	-	-	*	-	*	-	*	10	-	5	-	-
CTRMEM	9	TM	G	P1245	*	-	A11	KDP2	L	-	*	*	*	-	-	I	*	*	-	*	-	-	8	7	*
CTRPLT	8	TP	G	P12	*	-	1-3,7-12 14,15	K3	LC	-	*	-	-	-	-	B	*	*	-	-	6	11	17	15	*
CTUBE	3	TU	G	P145	*	-	A11	KDP2	LC	-	*	*	*	-	-	*	-	*	*	*	3	5	5	5	*
CTWIST	5	TW	G	P1	-	-	1-3,7-12 14,15	K2	L	-	*	-	-	-	-	-	*	*	-	-	3	5	4	5	*
CVISC	24	VS	G	P	-	-	7-9	-	-	B2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CWEDGE	40	WG	G	145	-	-	1-3,7-12 14,15	K3	L	-	*	*	*	-	-	*	-	*	-	*	0	0	9	13	-
FSLIST	46	FM	R _C	-	-	-	1-3,7-12	-	L	-	-	-	-	-	*	-	-	*	-	-	-	-	-	-	-
GENEL	-	-	G	-	-	M	A11	K6	-	-	-	-	-	-	*	-	-	-	-	-	-	-	-	-	-
PLØTEL	31	PL	G	-	-	P	A11	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

8.2 THE BAR ELEMENT

8.2.1 Input Data for the BAR Element

1. The ECPT/EST entries for the BAR are:

<u>Symbol</u>	<u>Descriptions</u>
SIL_a, SIL_b	Scalar indices of grid points a and b
$N_a, X_a, Y_a, Z_a \}$ $N_b, X_b, Y_b, Z_b \}$	Local coordinate system number and location in basic coordinates of the grid points
x_1, x_2, x_3	Orientation vector (see Figure 1 in section 1.3 of the User's Manual)
F	Flag for orientation vector definition
P_a, P_b	Pin flags for either end
Mat I. D.	Material property identification number
A	Cross-sectional area
I_1, I_2	Bending inertials in element coordinates about axes normal to reference planes 1 and 2 respectively
I_{12}	Cross-product bending inertia
J	Torsional Inertia
μ	Nonstructural mass per unit length
$a_x, a_y, a_z \}$ $b_x, b_y, b_z \}$	Vectors defining offset distances between BAR ends and grid points (see Figure 1 in section 1.3 of the User's Manual)
K_1, K_2	Shear factors
$c_1, c_2, d_1, d_2 \}$ $f_1, f_2, g_1, g_2 \}$	Positions on cross section of four points for stress calculations (see section 1.3.2 of the User's Manual)
t_μ	Temperature for material properties

2. Coordinate system data

The location (X_i, Y_i, Z_i) and local coordinate system number (N_i) of each grid point ($i = a$ or b) are used to calculate the 3 by 3 global-to-basic coordinate transformation matrices, $[T_a]$ and $[T_b]$.

STRUCTURAL ELEMENT DESCRIPTIONS

3. Material data

The material identification number "Mat I.D." and t_u are used to select the following:

- E - Modulus of elasticity
- G - Shear modulus
- ν - Poisson's ratio
- ρ - Density
- α - Thermal expansion coefficient
- T_0 - Reference temperature
- g_e - Structural damping ratio
- σ_t - Stress limit, tension
- σ_c - Stress limit, compression
- σ_s - Stress limit, shear

8.2.2 Stiffness Matrix Calculation (Subroutine KBAR of Module SMA1)

1. If the orientation flag F is nonzero, transform the given vector to basic coordinates:

$$\{v_o\} = [T_a] \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix}. \quad (1)$$

Otherwise,

$$\{v_o\} = \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix}. \quad (2)$$

2. Transfer the relative beam end locations to basic coordinates:

$$\begin{Bmatrix} \delta_{a1} \\ \delta_{a2} \\ \delta_{a3} \end{Bmatrix} = [T_a] \begin{Bmatrix} a_x \\ a_y \\ a_z \end{Bmatrix}, \quad (3)$$

THE BAR ELEMENT

$$\begin{Bmatrix} \delta_{b1} \\ \delta_{b2} \\ \delta_{b3} \end{Bmatrix} = [T_b] \begin{Bmatrix} b_x \\ b_y \\ b_z \end{Bmatrix} \quad (4)$$

3. The center axis of the beam, defined as $\{i\}$ is calculated as:

$$\{V_i\} = - \begin{Bmatrix} X_a - X_b + \delta_{a1} - \delta_{b1} \\ Y_a - Y_b + \delta_{a2} - \delta_{b2} \\ Z_a - Z_b + \delta_{a3} - \delta_{b3} \end{Bmatrix}, \quad (5)$$

$$L = (V_{i1}^2 + V_{i2}^2 + V_{i3}^2)^{1/2}, \quad (6)$$

$$\{i\} = \frac{1}{L} \{V_i\}. \quad (7)$$

4. The bending axis of the beam in plane 2 is:

$$\{k\} = \frac{\{i\} \times \{v_o\}}{|\{i\} \times \{v_o\}|} \quad (8)$$

5. The bending axis of the beam in plane 1 is:

$$\{j\} = \frac{\{k\} \times \{i\}}{|\{k\} \times \{i\}|} \quad (9)$$

6. The 6x6 matrix for transforming element displacements in the element coordinates to basic coordinate displacements is:

$$[T_{eb}] = \begin{bmatrix} \{i\} & \{j\} & \{k\} \\ 0 & 0 & 0 \end{bmatrix} \quad (10)$$

7. The 6 by 6 matrices for transforming global coordinate displacements to basic coordinate displacements are:

$$[C_a] = \begin{bmatrix} T_a & 0 \\ 0 & T_a \end{bmatrix}, \quad (11)$$

STRUCTURAL ELEMENT DESCRIPTIONS

$$[C_b] = \begin{bmatrix} T_b & 0 \\ -b & T_b \\ 0 & 0 \end{bmatrix} \quad (12)$$

8. The 6 by 6 matrices for transforming displacements of the grid points to displacements of the element ends are:

$$[E_a] = \begin{bmatrix} 1 & 0 & 0 & 0 & a_z & -a_y \\ 0 & 1 & 0 & -a_z & 0 & a_x \\ 0 & 0 & 1 & a_y & -a_x & 0 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (13)$$

$$[E_b] = \begin{bmatrix} 1 & 0 & 0 & 0 & b_z & -b_y \\ 0 & 1 & 0 & -b_z & 0 & b_x \\ 0 & 0 & 1 & b_y & -b_x & 0 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (14)$$

THE BAR ELEMENT

9. The 6 by 6 partitions of the element stiffness matrix in element coordinates are:

$$[K_{aa}^e] = \begin{bmatrix} \frac{AE}{l} & 0 & 0 & 0 & 0 & 0 \\ 0 & R_1 & B & 0 & -\frac{l}{2}B & \frac{l}{2}R_1 \\ 0 & B & R_2 & 0 & -\frac{l}{2}R_2 & \frac{l}{2}B \\ 0 & 0 & 0 & \frac{GJ}{l} & 0 & 0 \\ 0 & -\frac{l}{2}B & -\frac{l}{2}R_2 & 0 & k_2 & -\frac{l^2}{3}B \\ 0 & \frac{l}{2}R_1 & \frac{l}{2}B & 0 & -\frac{l^2}{3}B & k_1 \end{bmatrix}, \quad (15)$$

$$[K_{ab}^e] = \begin{bmatrix} -\frac{AE}{l} & 0 & 0 & 0 & 0 & 0 \\ 0 & -R_1 & -B & 0 & -\frac{l}{2}B & \frac{l}{2}R_1 \\ 0 & -B & -R_2 & 0 & \frac{l}{2}R_2 & \frac{l}{2}B \\ 0 & 0 & 0 & -\frac{GJ}{l} & 0 & 0 \\ 0 & \frac{l}{2}B & \frac{l}{2}R_2 & 0 & k_4 & -\frac{l^2}{6}B \\ 0 & -\frac{l}{2}R_1 & -\frac{l}{2}B & 0 & -\frac{l^2}{6}B & k_3 \end{bmatrix}, \quad (16)$$

$$[K_{ba}^e] = [K_{ab}^e]^T, \quad (17)$$

STRUCTURAL ELEMENT DESCRIPTIONS

$$[K_{bb}^e] = \begin{bmatrix} \frac{AE}{l} & 0 & 0 & 0 & 0 & 0 \\ 0 & R_1 & \beta & 0 & \frac{l}{2}\beta & -\frac{l}{2}R_1 \\ 0 & \beta & R_2 & 0 & \frac{l}{2}R_2 & -\frac{l}{2}\beta \\ 0 & 0 & 0 & \frac{GJ}{l} & 0 & 0 \\ 0 & \frac{l}{2}\beta & \frac{l}{2}R_2 & 0 & k_2 & -\frac{l^2}{3}\beta \\ 0 & -\frac{l}{2}R_1 & -\frac{l}{2}\beta & 0 & -\frac{l^2}{3}\beta & k_1 \end{bmatrix} \quad (18)$$

The terms are defined as:

If $I_{12} = 0$:

$$\beta = 0 \quad (19)$$

$$R_1 = \frac{12EI_1}{l^3} \left[1 + \frac{12EI_1}{K_1AGl^2} \right]^{-1} \quad (20)$$

$$R_2 = \frac{12EI_2}{l^3} \left[1 + \frac{12EI_2}{K_2AGl^2} \right]^{-1} \quad (21)$$

Note: If $K_i A G = 0$, set $\frac{1}{K_i A G} = 0$, $i = 1$ or 2

If $I_{12} \neq 0$

$$R_1 = \frac{12EI_1}{l^3} \quad (22)$$

$$R_2 = \frac{12EI_2}{l^3} \quad (23)$$

$$\beta = \frac{12EI_{12}}{l^3} \quad (24)$$

Note: In this case no shearing deformations are calculated.

For both cases

$$k_1 = \frac{\ell^2}{4} R_1 + \frac{EI_1}{\ell} , \quad (25)$$

$$k_2 = \frac{\ell^2}{4} R_2 + \frac{EI_2}{\ell} , \quad (26)$$

$$k_3 = \frac{\ell^2}{4} R_1 - \frac{EI_1}{\ell} , \quad (27)$$

$$k_4 = \frac{\ell^2}{4} R_2 - \frac{EI_2}{\ell} . \quad (28)$$

10. Process the end condition ("pin") data. The nonzero digits of the "pin flag" integers P_a and P_b specify the following:

$P = \left\{ \begin{array}{l} 1 \text{ implies no forces are transmitted to the element in the x-direction at the pinned end} \\ 2 \text{ implies no forces are transmitted to the element in the y-direction at the pinned end} \\ 3 \text{ implies no forces are transmitted to the element in the z-direction at the pinned end} \\ 4 \text{ implies no forces are transmitted to the element in the } \theta_x \text{-direction at the pinned end} \\ 5 \text{ implies no forces are transmitted to the element in the } \theta_y \text{-direction at the pinned end} \\ 6 \text{ implies no forces are transmitted to the element in the } \theta_z \text{-direction at the pinned end} \end{array} \right.$

1) Nonzero digits of the number P specify the unconnected degrees of freedom on the end of the BAR.

2) Construct the overall element matrix and perform the following operations:

a)

$$\left[\begin{array}{c|c} k_{aa}^e & k_{ab}^e \\ \hline k_{ba}^e & k_{bb}^e \end{array} \right] = \left[\begin{array}{ccc} k_{11} & k_{12} & \dots \\ k_{21} & & \\ \vdots & & k_{12,12} \end{array} \right] . \quad (29)$$

b) Convert the pin numbers to row numbers in the $[k]$ matrix. If a pin number refers to end "a", it corresponds to the row number. If it refers to end "b", the row number is obtained by adding six to the pin number.

c) For each row of the $[k]$ matrix perform the following operation to obtain the new stiffness matrix $[k']$

$$k'_{jl} = k_{jl} - \frac{k_{il}k_{ji}}{k_{ii}} \quad \begin{cases} j = 1, \dots, 12, j \neq i \\ l = 1, \dots, 12, l \neq i \end{cases} \quad (30)$$

and $k'_{jl} = 0$ for $j = i$ or $l = i$, where i is the row number obtained from the pin number as in b).

This operation causes the i^{th} row and column to be zero, and disconnects that degree of freedom from the matrix. Repeat for each pin index.

d) Repartition the matrix into the four original sections, carrying the zero rows and columns along.

11. The equations to convert the partitions to global coordinates are:

$$[k_{aa}] = \{T_{eb}^T C_a E_a\}^T [k_{aa}^e] \{T_{eb}^T C_a E_a\} \quad (31)$$

$$[k_{ab}] = \{T_{eb}^T C_a E_a\}^T [k_{ab}^e] \{T_{eb}^T C_b E_b\} \quad (32)$$

$$[k_{bb}] = \{T_{eb}^T C_b E_b\}^T [k_{bb}^e] \{T_{eb}^T C_b E_b\} \quad (33)$$

$$[k_{ba}] = [k_{ab}]^T \quad (34)$$

8.2.3 Lumped Mass Matrix Calculation (Subroutine MBAR of Module SMA2)

$$[M_a] = [M_b] = \begin{bmatrix} m/2 & & \\ & m/2 & 0 \\ & & m/2 \\ - - - & - - - & - - - \\ & 0 & 0 \end{bmatrix}, \quad (35)$$

where:

$$m = 2 (\rho A + \mu), \quad (36)$$

and:

$$[M_{aa}] = \{T_{eb}^T C_a E_a\}^T [M_a] \{T_{eb}^T C_a E_a\}, \quad (37)$$

$$[M_{bb}] = \{T_{eb}^T C_b E_b\}^T [M_b] \{T_{eb}^T C_b E_b\}, \quad (38)$$

$$[M_{ab}] = [M_{ba}] = [0]. \quad (39)$$

The equations for the generation of the "consistent" or coupled mass matrix for the BAR are given in Section 8.2.8.

8.2.4 Element Load Calculation (Subroutine BAR of Module SSG1)

- Form l , $\{i\}$, $[C_a]$, $[C_n]$, $[E_a]$, $[E_b]$, and $[k']$ as in Equation 30.
- Partition the 12×12 matrix into four 6×6 matrices

$$[k'] \Rightarrow \begin{bmatrix} K_{aa}^e & K_{ab}^e \\ K_{ba}^e & K_{bb}^e \end{bmatrix}. \quad (40)$$

c) Form the vector

$$\{u_a^t\} = \begin{Bmatrix} -\alpha\ell(\bar{T} - T_0) - \delta \\ -\frac{\alpha\ell^2}{6} [T'_{1a} + 2T'_{1b}] \\ -\frac{\alpha\ell^2}{6} [T'_{2a} + 2T'_{2b}] \\ 0 \\ -\frac{\alpha\ell}{2} [T'_{2a} + T'_{2b}] \\ \frac{\alpha\ell}{2} [T'_{1a} + T'_{1b}] \end{Bmatrix}, \quad (41)$$

where \bar{T} is the average of \bar{T}_a and \bar{T}_b , δ is the enforced deformation, and T'_i are the gradients.

d) The load vectors in global coordinates are:

$$\begin{aligned} \{P_a\} &= [E_a^T][C_a^T][T_{eb}][K_{aa}^P]\{u_a^t\}, \\ \{P_b\} &= [E_b^T][C_b^T][T_{eb}][K_{ba}^P]\{u_a^t\}. \end{aligned} \quad (42)$$

8.2.5 Element Stress Calculations (Subroutines SBAR1 and SBAR2 of Module SDR2)

The stress and force data are calculated in two phases. The first phase (subroutine SBAR1) calculates unique stress versus displacement, temperature and enforced deformation functions for each element. The second phase (subroutine SBAR2) applies the various subcase displacement vectors to product the element forces and stresses.

Phase 1 calculations are as follows:

1. Using the algorithms given in the description of the stiffness matrix calculations for the element (Section 8.2.2), calculate the following data:

$[T_{eb}]$ = 6x6 element coordinate transformation

$[E_a], [E_b]$ - Offset transformation matrices (6x6)

$[C_a], [C_b]$ - 6x6 global to basic coordinate transformations

$[K_e']$ - 12x12 stiffness matrix in element coordinates with pin joint effects

l - Length of BAR

2. Partition the stiffness matrix $[K_e']$ saving only the upper 6x6 matrices, $[k_{aa}]$ and $[k_{ab}]$.

$$[K_e'] \Rightarrow \begin{bmatrix} k_{aa} & k_{ab} \\ - & - \\ k_{ba} & k_{bb} \end{bmatrix} \quad (43)$$

3. The stress matrices are:

$$[S_a] = [k_{aa}][T_{eb}]^T[C_a][E_a] \quad (44)$$

$$[S_b] = [k_{ab}][T_{eb}]^T[C_b][E_b] \quad (45)$$

STRUCTURAL ELEMENT DESCRIPTIONS

4. The temperature and enforced deformation matrix is:

$$[S_t] = [K_{aa}] \begin{bmatrix} \alpha l & 0 & 0 & 0 & 0 \\ 0 & \frac{\alpha l^2}{6} & \frac{2\alpha l^2}{6} & 0 & 0 \\ 0 & 0 & 0 & \frac{\alpha l^2}{6} & \frac{2\alpha l^2}{6} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\alpha l}{2} & \frac{\alpha l}{2} \\ 0 & -\frac{\alpha l}{2} & -\frac{\alpha l}{2} & 0 & 0 \end{bmatrix} \quad (46)$$

Phase 2 element force calculations are as follows:

1. The static element forces are calculated by the equation:

$$\{P\} = [S_a]\{u_a\} + [S_b]\{u_b\} + [S_t] \begin{Bmatrix} \bar{T}_z \\ T'_{1a} \\ T'_{1b} \\ T'_{2a} \\ T'_{2b} \end{Bmatrix} \quad (47)$$

where $[S_a]$ and $[S_b]$ are the displacement-stress matrices, $\{u_a\}$ and $\{u_b\}$ are the displacement vectors, and \bar{T}_z , T'_{1a} , etc. are the element thermal resultants. In terms of the given temperatures at the ends, \bar{T}_a and \bar{T}_b , the equation for \bar{T}_z is:

$$\bar{T}_z = \frac{\bar{T}_a + \bar{T}_b}{2} - T_0 \quad (48)$$

2. The element axial force is:

$$F_x = -[P_1 + S_\delta \delta] \quad (49)$$

3. The element shear loads are:

$$V_1 = -P_2 , \quad (51)$$

$$V_2 = -P_3 . \quad (52)$$

4. The torque and moments are:

$$T = -P_4 , \quad (53)$$

$$M_{1a} = -P_6 , \quad (54)$$

$$M_{2a} = P_5 , \quad (55)$$

$$M_{1b} = M_{1a} - V_1 l , \quad (56)$$

$$M_{2b} = M_{2a} - V_2 l . \quad (57)$$

Phase 2 element stress calculations are as follows:

1. The stresses due to bending are:

$$k_{1a} = \frac{M_{2a} I_{12} - M_{1a} I_2}{I_1 I_2 - I_{12}^2} , \quad (58)$$

$$k_{2a} = \frac{M_{1a} I_{12} - M_{2a} I_1}{I_1 I_2 - I_{12}^2} . \quad (59)$$

$$\sigma_{ca} = k_{1a} c_1 + k_{2a} c_2 , \quad (60)$$

$$\sigma_{da} = k_{1a} d_1 + k_{2a} d_2 , \quad (61)$$

$$\sigma_{fa} = k_{1a} f_1 + k_{2a} f_2 , \quad (62)$$

$$\sigma_{ga} = k_{1a} g_1 + k_{2a} g_2 . \quad (63)$$

For σ_{cb} , σ_{db} , σ_{fb} , σ_{gb} use the above equations interchanging the subscripts for b and a.

Equation 50 is intentionally missing.

STRUCTURAL ELEMENT DESCRIPTIONS

The stresses calculated at points c, d, e, and f on the cross section will be modified by the element temperatures T_{ac} , T_{ad} , ... T_{bc} , ... if at least one of the T values is nonzero. At end a:

$$\Delta\sigma_c = -E\alpha(T_{ac} - T'_{1a} c_1 - T'_{2a} c_2 - \bar{T}_a)$$

$$\Delta\sigma_d = -E\alpha(T_{ad} - T'_{1a} d_1 - T'_{2a} d_2 - \bar{T}_a)$$

etc.

⋮

At end b:

$$\Delta\sigma_c = -E\alpha(T_{bc} - T'_{1b} c_1 - T'_{2b} c_2 - \bar{T}_b)$$

$$\Delta\sigma_d = -E\alpha(T_{bd} - T'_{1b} d_1 - T'_{2b} d_2 - \bar{T}_b)$$

etc.

⋮

(63a)

(T_a and T_b are the given average temperatures at the ends.) The above stresses are added to the stresses calculated in Equations 60 - 63.

2. The axial stress is:

$$\sigma_{ax} = \frac{F_x}{A} \quad (64)$$

3. The maxima and minima are:

$$\sigma_a \max = \sigma_{ax} + \max(\sigma_{ca}, \sigma_{da}, \sigma_{fa}, \sigma_{ga}) \quad (65)$$

$$\sigma_b \max = \sigma_{ax} + \max(\sigma_{cb}, \sigma_{db}, \sigma_{fb}, \sigma_{gb}) \quad (66)$$

$$\sigma_a \min = \sigma_{ax} + \min(\sigma_{ca}, \sigma_{da}, \sigma_{fa}, \sigma_{ga}) \quad (67)$$

$$\sigma_b \min = \sigma_{ax} + \min(\sigma_{cb}, \sigma_{db}, \sigma_{fb}, \sigma_{gb}) \quad (68)$$

4. The margins of safety in tension, $M.S._t$, and compression, $M.S._c$, are as follows:

$$M.S._t = \begin{cases} \min \left(\frac{\sigma_t}{\sigma_{a \max}}, \frac{\sigma_t}{\sigma_{b \max}} \right) - 1, & \sigma_t > 0 \\ \text{Integer "1"} & , \sigma_t \leq 0 \text{ or } \max(\sigma_{a \max}, \sigma_{b \max}) \leq 0 \end{cases} \quad (69)$$

Define $\sigma'_c = -|\sigma_c|$. Then:

$$M.S._c = \begin{cases} \min \left(\frac{\sigma'_c}{\sigma_{a \min}}, \frac{\sigma'_c}{\sigma_{b \min}} \right) - 1, & \sigma'_c \neq 0 \\ \text{Integer "1"} & , \sigma'_c = 0 \text{ or } \min(\sigma_{a \min}, \sigma_{b \min}) \geq 0 \end{cases} \quad (70)$$

8.2.6 Differential Stiffness Matrix Calculation (Subroutine DBEAM of Module DSMG1)

Many of the equations used in this calculation routine are identical to the stiffness matrix and element force calculations. Refer to Section 8.2.2 and 8.2.5 for details.

1. Calculate $[T_{eb}]$, $[C_a]$, $[C_b]$, $[E_a]$, $[E_b]$ and $[K]$, the matrices used in the BAR stiffness matrix generation, Section 8.2.2.

2. Calculate the forces in the element using the equations in Section 8.2.5.

3. The number 2, 3, 4, 5, 6, 8, 9, 10, 11 and 12th rows and columns of the 12 by 12 differential stiffness matrix are given in Figure 1. The first and seventh rows and columns are zero. The terms are identical to the element forces calculated for output (Equations 51, 52 and 54 through 57 of Section 8.2.5) with the following notational changes:

$$\begin{array}{ll} M_{ay} = M_{2a} & M_{by} = M_{2b} \\ M_{az} = M_{1a} & M_{bz} = M_{1b} \\ V_y = V_1 & V_z = V_2 \end{array}$$

$[K_e^d] =$

$\frac{6F_x}{5l}$	0	$-\frac{M_{ay}}{l}$	0	$-\frac{F_x}{10}$	$-\frac{6F_x}{5l}$	0	$-\frac{M_{by}}{l}$	0	$-\frac{F_x}{10}$
0	$\frac{6F_x}{5l}$	$-\frac{M_{az}}{l}$	$\frac{F_x}{10}$	0	0	$-\frac{6F_x}{5l}$	$-\frac{M_{bz}}{l}$	$\frac{F_x}{10}$	0
$-\frac{M_{ay}}{l}$	$-\frac{M_{az}}{l}$	$\frac{JF_x}{l^2 A}$	$-\frac{lV_y}{6}$	$-\frac{lV_z}{6}$	$\frac{M_{ay}}{l}$	$\frac{M_{az}}{l}$	$-\frac{JF_x}{l^2 A}$	$\frac{lV_y}{6}$	$\frac{lV_z}{6}$
0	$\frac{F_x}{10}$	$-\frac{lV_y}{6}$	$\frac{2lF_x}{15}$	0	0	$-\frac{F_x}{10}$	$\frac{lV_y}{6}$	$-\frac{lF_x}{30}$	0
$-\frac{F_x}{10}$	0	$-\frac{lV_z}{6}$	0	$-\frac{2lF_x}{15}$	$\frac{F_x}{10}$	0	$\frac{lV_z}{6}$	0	$-\frac{lF_x}{30}$
$-\frac{6F_x}{5}$	0	$\frac{M_{ay}}{l}$	0	$\frac{F_x}{10}$	$\frac{6F_x}{5l}$	0	$\frac{M_{by}}{l}$	0	$\frac{F_x}{10}$
0	$-\frac{6F_x}{l}$	$\frac{M_{az}}{l}$	$-\frac{F_x}{10}$	0	0	$\frac{6F_x}{5l}$	$\frac{M_{bz}}{l}$	$-\frac{F_x}{10}$	0
$-\frac{M_{by}}{l}$	$-\frac{M_{bz}}{l}$	$-\frac{JF_x}{l^2 A}$	$\frac{lV_y}{6}$	$\frac{lV_z}{6}$	$\frac{M_{by}}{l}$	$\frac{M_{bz}}{l}$	$\frac{JF_x}{l^2 A}$	$-\frac{lV_y}{6}$	$-\frac{lV_z}{6}$
0	$\frac{F_x}{10}$	$\frac{lV_y}{6}$	$-\frac{lF_x}{30}$	0	0	$-\frac{F_x}{10}$	$-\frac{lV_y}{6}$	$\frac{2lF_x}{15}$	0
$-\frac{F_x}{10}$	0	$\frac{lV_z}{6}$	0	$-\frac{lF_x}{30}$	$\frac{F_x}{10}$	0	$-\frac{lV_z}{6}$	0	$\frac{2lF_x}{15}$

Figure 1. - Differential stiffness matrix for a BAR element, rows and columns 1 and 7 deleted.

THE BAR ELEMENT

4. The effects of "pin joints" are added by applying the elastic stiffness constraints. The elastic stiffness matrix, $[K^e]$, with no pin joints is equivalent to the matrix $[k]$ in Equation 29. If coordinate number i is released by a pin flag the differential stiffness matrix must be modified as follows:

a. If $i \neq j$ and $l \neq j$, $i = 1, 2, \dots, 12$ and $l = 1, 2, \dots, 12$:

$$(K_{il}^{d*})_m = (K_{il}^{d*})_{m-1} - \frac{K_{lj}^e (K_{ji}^{d*})_{m-1}}{K_{jj}^e} - \frac{K_{ji}^e (K_{lj}^{d*})_{m-1}}{K_{jj}^e} + \frac{K_{lj}^e K_{ji}^e + (K_{jj}^{d*})_{m-1}}{(K_{jj}^e)^2}, \quad (71)$$

where m is the (row) index of the pin joint number, $1 \leq m \leq 12$. For $m = 0$, define

$$(K_{il}^{d*})_0 = (K_{il}^d). \quad (72)$$

b. If i or $j = l$

$$K_{ij}^{d*} = 0 \quad i = 1, \dots, 12, \quad (73)$$

$$K_{jl}^{d*} = 0 \quad j = 1, \dots, 12. \quad (74)$$

5. The 12 by 12 matrix $[K^{d*}]$ is now partitioned into 6 by 6 matrices related to each grid point

$$[K^{d*}] = \begin{bmatrix} K_{aa}^{d*} & K_{ab}^{d*} \\ K_{ba}^{d*} & K_{bb}^{d*} \end{bmatrix}. \quad (75)$$

6. If point "p" is the pivot point ($p = a$ or b), the matrices generated in global coordinates are:

$$[K_{pa}^d] = ([T_{eb}]^T [C_p] [E_p])^T [K_{pa}^{d*}] ([T_{eb}] [C_a] [E_a]), \quad (76)$$

$$[K_{pb}^d] = ([T_{eb}]^T [C_p] [E_p])^T [K_{pb}^{d*}] ([T_{eb}]^T [C_b] [E_b]). \quad (77)$$

8.2.7 Piecewise Linear Analysis Calculations (Subroutine PSBAR of Module PLA3 and Subroutine PKBAR of Module PLA4)

The additional ECPTNL and ESTNL data block entries for a BAR element are:

- | | |
|----------------|--|
| ϵ_0^* | - The previously computed axial strain value once removed. |
| ϵ^* | - The previously computed axial strain value. |
| E^* | - The previously computed modulus of elasticity. |
| V_1^* | - The previously computed element forces and moments. |
| V_2^* | |
| T^* | |
| M_{1a}^* | |
| M_{2a}^* | |

All of the above values are initially zero with the exception of E^* , which is initially the original modulus of elasticity present on a MAT1 bulk data card.

For both stress (subroutine PSBAR) and stiffness matrix (subroutine PKBAR) calculations, the following data are generated:

ϵ , $[T_{eb}]$, $[C_a]$, $[C_b]$, $[E_a]$, $[E_b]$, $[k_{aa}^e]$, and $[k_{ab}^e]$ as in Equations 6, 10, 11, 12, 13, 14 and 30 in section 4.87.2.1. Note that: a) $[k_{aa}^e]$ and $[k_{ab}^e]$ are the partitions of the stiffness matrix with pin joint effects taken into account; b) for stress calculations, E^* is used to compute $[k_{aa}^e]$ and $[k_{ab}^e]$; and c) for stiffness matrix calculations E_1 (see Equation 84 below) is used to compute $[k_{aa}^e]$ and $[k_{ab}^e]$.

Using the incremental displacement vectors, $\{\Delta u_a\}$ and $\{\Delta u_b\}$, calculate the incremental strain:

$$\Delta \epsilon = \frac{1}{2} \{T_{eb1}\}^T \left[[C_b][E_b]\{\Delta u_b\} - [C_a][E_a]\{\Delta u_a\} \right], \quad (78)$$

where $\{T_{eb1}\}$ is the first column of $[T_{eb}]$. If coordinate "1" of either P_a or P_b (the pin flags) is "on", the element is treated as linear. This determination is made in the Piecewise Linear

Analysis pre-processor module, PLA1.

Calculate the extensional strains:

$$\Delta \epsilon^* = \epsilon^* - \epsilon_0^* , \quad (79)$$

$$\epsilon_1 = \epsilon^* + \Delta \epsilon , \quad (80)$$

$$\epsilon_2 = \epsilon_1 + \gamma(\Delta \epsilon) , \quad (81)$$

where γ is the ratio of the next load increment to the present load increment.

The stresses

$$\sigma_1 = f(\epsilon_1) , \quad (82)$$

$$\sigma_2 = f(\epsilon_2) , \quad (83)$$

are computed, where f is the tabular stress-strain function. (When $\epsilon^* = 0$, define $\sigma_1 = E_0 \epsilon_1$, where E_0 is the modulus of elasticity on the MAT1 card)

For stiffness matrix generation the new material properties are:

$$E_1 = \begin{cases} \frac{\sigma_2 - \sigma_1}{\epsilon_2 - \epsilon_1} , & \text{if } \epsilon_2 \neq \epsilon_1 \\ E^* , & \text{if } \epsilon_2 = \epsilon_1 \end{cases} ; \quad (84)$$

and

$$G_1 = \frac{E_1}{E_0} G_0 . \quad (85)$$

where E_0 and G_0 are elastic moduli obtained from the MAT1 bulk data card via subroutine MAT. Note that E_1 is calculated in PSBAR only to predict the next value of E , i.e., to update the ESTNL entry.

STRUCTURAL ELEMENT DESCRIPTIONS

For plastic element stresses and forces, the values are calculated in a fashion similar to that found in the phase 2 subroutine, SBAR2, (Section 8.2.5) of the SDR module.

They are:

$$\{\Delta P\} = [k_{aa}][T_{eb}]^T[C_a][E_a]\{\Delta u_a\} + [k_{ab}][T_{eb}]^T[C_b][E_b]\{\Delta u_b\}, \quad (86)$$

$$F_x = A\sigma_1, \quad (87)$$

$$V_1 = -\Delta P_2 + V_1^*, \quad (88)$$

$$V_2 = -\Delta P_3 + V_2^*, \quad (89)$$

$$T = -\Delta P_4 + T^*, \quad (90)$$

$$M_{1a} = -\Delta P_6 + M_{1a}^*, \quad (91)$$

$$M_{2a} = \Delta P_5 + M_{2a}^*, \quad (92)$$

$$M_{1b} = M_{1a} - V_1\ell, \quad (93)$$

$$M_{2b} = M_{2a} - V_2\ell. \quad (94)$$

The stresses due to bending, the axial stress, the minimum and maximum stresses, and the margins of safety are computed as in Equations 58 through 70.

The new ESTNL and ECPTNL entries are:

$$\epsilon_0^* = \epsilon^*, \quad (95)$$

$$\epsilon^* = \epsilon_1, \quad (96)$$

$$E^* = E_1, \quad (97)$$

THE BAR ELEMENT

$$V_1^* = V_1 , \quad (98)$$

$$V_2^* = V_2 , \quad (99)$$

$$T^* = T , \quad (100)$$

$$M_{1a}^* = M_{1a} , \quad (101)$$

$$M_{2a}^* = M_{2a} . \quad (102)$$

STRUCTURAL ELEMENT DESCRIPTIONS

8.2.8 "Consistent" Mass Matrix Calculation (Subroutine MCBAR of Module SMA2)

1. Generate the 12 by 12 matrix:

$$[M^e] = \frac{m}{420} \begin{bmatrix} 175 & 0 & 0 & 0 & 0 & 0 & 35 & 0 & 0 & 0 & 0 & 0 \\ & 156 & 0 & 0 & 0 & 22\ell & 0 & 54 & 0 & 0 & 0 & -13\ell \\ & & 156 & 0 & -22\ell & 0 & 0 & 0 & 54 & 0 & 13\ell & 0 \\ & & & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & & 4\ell^2 & 0 & 0 & 0 & -13\ell & 0 & -3\ell^2 & 0 \\ & & & & & 4\ell^2 & 0 & 13\ell & 0 & 0 & 0 & -3\ell^2 \\ & & & & & & 175 & 0 & 0 & 0 & 0 & 0 \\ & & & & & & & 156 & 0 & 0 & 0 & -22\ell \\ & & & & & & & & 156 & 0 & 22\ell & 0 \\ & & & & & & & & & 0 & 0 & 0 \\ & & & & & & & & & & 4\ell^2 & 0 \\ & & & & & & & & & & & 4\ell^2 \end{bmatrix} \quad (103)$$

where

$$m = (\rho A + \mu)\ell \quad (104)$$

2. If "pin joints" exist (P_a or P_b nonzero), generate the "unpinned" 12 by 12 stiffness matrix in element coordinates, $[K^e]$, as in Equation 29, Section 8.2.2.

For each pin joint of index j , perform the operations for $i = 1, \dots, 12$ and $\ell = 1, \dots, 12$:

$$M_{i\ell}^p = M_{i\ell}^e - \frac{K_{\ell j}^e M_{ji}^e}{K_{jj}^e} - \frac{K_{ji}^e M_{\ell j}^e}{K_{jj}^e} + \frac{K_{\ell j}^e K_{ji}^e M_{jj}^e}{(K_{jj}^e)^2} \quad (105)$$

After each pin joint j operation, replace $[M^e]$ by the "pinned" matrix $[M^p]$.

4. Partition the matrix into 6 by 6 submatrices:

THE BAR ELEMENT

$$[M] = \begin{bmatrix} M_{aa} & M_{ab} \\ M_{ba} & M_{bb} \end{bmatrix} \quad (106)$$

5. The matrices are converted to global coordinates by the equation:

$$[M_{ij}^g] = [T_{eb}]^T [C_i] [E_i] \quad [M_{ij}] \quad [T_{eb}]^T [C_j] [E_j] \quad (107)$$

where i is the pivot point (a or b) and j is used twice (for both a and b).

8.2.9 Thermal Analysis Calculations for the BAR Element

If a "stiffness" matrix for heat transfer analysis is to be generated, word 56 in COMMON data block SYSTEM is +1. The length, ℓ , of the element is calculated with the structure analysis code, described in Section 8.2.2. The thermal conductivity coefficient, k , is obtained by calling subroutine HMAT, rather than MAT. The matrix terms are:

For the pivot point i ,

$$K_{ii} = \frac{k A}{\ell}.$$

For $j \neq i$,

$$K_{ij} = -\frac{k A}{\ell}.$$

The "mass" matrix for heat transfer analysis is generated in subroutine MBAR. The capacity coefficient C_p is determined by subroutine HMAT and the matrix terms placed in the BGG matrix are:

$$B_{ii} = \frac{C_p A \ell}{2}.$$

The "stress" recovery is performed by subroutines SDHTF1, SDHTFF, and SDHTF2 in module SDR2. In Phase 1, the value k is extracted with subroutine HMAT and the output is:

$$K_1 = k,$$

$$[C] = \frac{1}{\ell} \begin{bmatrix} -1 & 1 \end{bmatrix}.$$

In Phase 2, the gradient and flux are:

$$\Delta T = [C] \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix}$$

$$Q = -K_1 \Delta T$$

8.3 THE SHEAR PANEL AND TWIST PANEL ELEMENTS

8.3.1 Input Data for SHEAR and TWIST Panels

1. The ECPT/EST entries for shear (SHEAR) and twist (TWIST) panel elements are:

<u>Symbol</u>	<u>Description</u>
$SIL_i, i=1,2,3,4$	Scalar indices for the connected points
$N_i, X_i, Y_i, Z_i \}$ $i = 1,2,3,4$	Local coordinate system number and basic co-ordinate location for each of the connected points.
Mat I.D.	Material identification number
t	Panel thickness
μ	Nonstructural mass per unit area
t_μ	Temperature for material properties

2. Coordinate system data

Using $N_i, X_i, Y_i, Z_i, i = 1,2,3,4$ the program constructs $[T_i], i = 1,2,3,4$, the 3 by 3 global-to-basic transformation matrix for each point.

3. Material data

MAT I.D. and t_μ are used, by utility routine MAT, to produce the following terms from the MPT and DIT data blocks:

<u>Symbol</u>	<u>Description</u>
E	Modulus of elasticity
G	Shear modulus
ν	Poisson's ratio
ρ	Density
α	Thermal expansion coefficient
T_0	Reference temperature
g_e	Structural damping coefficient
$\sigma_t, \sigma_c, \sigma_b$	Stress limits

8.3.2 Definition of Element Geometry

A mean plane is defined as parallel to the two diagonal lines and halfway between them. The projections of the points at the corners of the element onto the plane and the normal to the plane define the element coordinate system. Using standard vector algebra, the steps are:

1. Define:

$$\{V_{01}\} = \begin{Bmatrix} X_1 \\ Y_1 \\ Z_1 \end{Bmatrix}, \quad \{V_{02}\} = \begin{Bmatrix} X_2 \\ Y_2 \\ Z_2 \end{Bmatrix}, \text{ etc.} \quad (1)$$

2. Define diagonal vectors:

$$\{v_{d1}\} = \{V_{03}\} - \{V_{01}\}, \quad (2)$$

$$\{v_{d2}\} = \{V_{04}\} - \{V_{02}\}. \quad (3)$$

3. Define normal vector (x denotes cross product):

$$\{k_n\} = \{v_{d1}\} \times \{v_{d2}\}, \quad (4)$$

$$\{k\} = \frac{\{k_n\}}{|\{k_n\}|}, \quad (5)$$

$$A = \frac{1}{2} |\{k_n\}| \quad (\text{the projected area of the element}). \quad (6)$$

4. Define the vectors along the side of the element (see Figures 2 and 3)

$$\{v_{12}\} = \{V_{02}\} - \{V_{01}\}, \quad (7)$$

$$\{v_{41}\} = \{V_{01}\} - \{V_{04}\}. \quad (8)$$

5. Define transformation matrix $[T_e]$, which transforms element coordinate to basic coordinates, using unit vectors:

$$\{v_{12}^p\} = \{v_{12}\} - (\{v_{12}\}^T \{k\}) \{k\} , \quad (9)$$

$$\{i\} = \frac{\{v_{12}^p\}}{|\{v_{12}^p\}|} , \quad (10)$$

$$\{j\} = \{k\} \times \{i\} , \quad (11)$$

$$[T_e] = \begin{bmatrix} i_1 & j_1 \\ i_2 & j_2 \\ i_3 & j_3 \end{bmatrix} . \quad (12)$$

6. Transform the four corner point of the element from basic coordinates to the element system:

$$\{r_1\} = \begin{Bmatrix} x_1 \\ y_1 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} , \quad (13)$$

$$\{r_2\} = \begin{Bmatrix} x_2 \\ y_2 \end{Bmatrix} = \begin{Bmatrix} |v_{12}^p| \\ 0 \end{Bmatrix} , \quad (14)$$

$$\{r_3\} = \begin{Bmatrix} x_3 \\ y_3 \end{Bmatrix} = [T_e]^T \{v_{d1}\} , \quad (15)$$

$$\{r_4\} = \begin{Bmatrix} x_4 \\ y_4 \end{Bmatrix} = - [T_e]^T \{v_{d1}\} . \quad (16)$$

The four corners of the element are now projected onto the mean plane.

STRUCTURAL ELEMENT DESCRIPTIONS

7. The following conditions should be met. Otherwise, the interior angle at the indicated point is not valid.

$$y_3 > 0 \quad (\text{If not, the interior angle at point 2} > 180^\circ) \quad , \quad (17)$$

$$x_3 > \frac{y_3}{y_4} x_4 \quad (\text{If not, the interior angle at point 4} > 180^\circ) \quad , \quad (18)$$

$$y_4 > 0 \quad (\text{If not, the interior angle at point 1} > 180^\circ) \quad , \quad (19)$$

$$x_4 < x_2 - (x_2 - x_3) \frac{y_4}{y_3} \quad (\text{If not, the interior angle at point 3} > 180^\circ) \quad . \quad (20)$$

8.3.3 Coefficient Generation

The shape of the panel may be a parallelogram, a trapezoid, or a general quadrilateral, and the equations will be different for each case. The slopes of the opposite sides are checked for parallel effects, and the correct routine is used for each possibility.

1. Check for parallel effects:

If

$$\left| \frac{y_3 - y_4}{x_3 - x_4} \right| < \epsilon, \quad (21)$$

sides 1 and 3 of the panel are parallel ($\epsilon = 10^{-1}$).

If

$$\left| \frac{y_4(x_3 - x_2) - y_3x_4}{x_4(x_3 - x_2) + y_4y_3} \right| < \epsilon, \quad (22)$$

sides 2 and 4 are parallel. If both terms are less than ϵ , (i.e., the panel is a parallelogram), go to step (4). If both terms are greater than ϵ , go to step (5). If the one pair of parallel sides is 1 and 3, go to step (2); if the one pair of parallel sides is 2 and 4, go to step (3).

2. In this case the line connecting points 3 and 4 is approximately parallel to the line connecting points 1 and 2. The equations are:

$$y_p = \frac{x_2 y_3 y_4}{y_3 x_4 - y_4 (x_3 - x_2)} , \quad (23)$$

$$p_i = y_p - y_i \quad (i = 1, 2, 3, 4) , \quad (24)$$

$$x_p = \frac{x_2 y_3 x_4}{y_3 x_4 - y_4 (x_3 - x_2)} , \quad (25)$$

$$a = \left(\frac{x_2 - x_p}{y_p} \right) , \quad (26)$$

$$c = \left(\frac{x_1 - x_p}{y_p} \right) , \quad (27)$$

$$Z = \frac{p_1 p_2}{p_3 p_4} \frac{A}{2Gt} \left\{ 1 + \frac{2}{3(1+\nu)} (a^2 + ac + c^2) \right\} . \quad (28)$$

3. In this case the line connecting points 1 and 4 is approximately parallel to the line connecting points 2 and 3. The equations are:

$$d = -\frac{1}{2} \left[\frac{x_4}{y_4} + \frac{x_3 - x_2}{y_3} \right] , \quad (29)$$

$$x_q = x_4 - \frac{x_3 - x_4}{y_3 - y_4} y_4 , \quad (30)$$

STRUCTURAL ELEMENT DESCRIPTIONS

$$p_i = [(x_q - x_i) - y_i d] \frac{1}{\sqrt{1 + d^2}} \quad (i = 1, 2, 3, 4) \quad , \quad (31)$$

$$b = \frac{(x_q - x_4)d + y_4}{(x_q - x_4) - y_4 d} \quad , \quad (32)$$

$$Z = \frac{p_1 p_2}{p_3 p_4} \frac{A}{2Gt} \left\{ 1 + \frac{2}{3(1+\nu)} (b^2 + bd + d^2) \right\}. \quad (33)$$

4. In this case the panel approximates a parallelogram. The equations to solve are:

$$p_i = 1, \quad (i = 1, 2, 3, 4), \quad (34)$$

$$d = -\frac{1}{2} \left(\frac{x_4}{y_4} + \frac{x_3 - x_2}{y_3} + \frac{y_3 - y_4}{x_3 - x_4} \right) \quad , \quad (35)$$

$$Z = \frac{A}{2Gt} \left(1 + \frac{2d^2}{1+\nu} \right). \quad (36)$$

5. In this case no parallel effects exist. The equations are:

$$x_q = x_4 - \frac{(x_3 - x_4)}{(y_3 - y_4)} y_4 \quad , \quad (37)$$

$$x_p = \frac{x_2 x_4 y_3}{y_3 x_4 - y_4 (x_3 - x_2)} , \quad (38)$$

$$y_p = \frac{x_2 y_3 y_4}{y_3 x_4 - y_4 (x_3 - x_2)} , \quad (39)$$

$$l = \sqrt{(x_q - x_p)^2 + y_p^2} , \quad (40)$$

$$d = \frac{x_q - x_p}{y_p} , \quad (41)$$

$$p_i = \frac{y_p}{l} [(x_q - x_i) - y_i d] \quad (i = 1, 2, 3, 4) , \quad (42)$$

$$c = \frac{l}{p_1} - d , \quad (43)$$

$$b = \frac{l}{p_4} - c , \quad (44)$$

$$a = \frac{l}{p_2} - d . \quad (45)$$

STRUCTURAL ELEMENT DESCRIPTIONS

Let:

$$\begin{aligned}
 F = & \frac{p_1 p_2 p_3 p_4}{2g^2} \left[[(a+b) + \frac{2}{3} (a^3+b^3) + \frac{1}{5} (a^5+b^5)] \log_e |a+b| \right. \\
 & + [(c+d) + \frac{2}{3} (c^3+d^3) + \frac{1}{5} (c^5+d^5)] \log_e |c+d| \\
 & - [(b+c) + \frac{2}{3} (b^3+c^3) + \frac{1}{5} (b^5+c^5)] \log_e |b+c| \\
 & - [(d+a) + \frac{2}{3} (d^3+a^3) + \frac{1}{5} (d^5+a^5)] \log_e |d+a| \\
 & + \frac{1}{10} [(a^2-c^2) (b^3-d^3) + (b^2-d^2) (a^3-c^3)] \\
 & \left. - \frac{1}{5} [(a-c) (b^4-d^4) + (b-d) (a^4-c^4)] \right] .
 \end{aligned}
 \tag{46}$$

Then:

$$Z = \frac{p_1 p_2}{p_3 p_4} \frac{1}{2Gt} \left\{ A + \frac{4}{1+v} \left[F - \frac{2}{3} A \right] \right\} .
 \tag{47}$$

8.3.4 Stiffness Matrix Formulation For a SHEAR Panel (Subroutine KPANEL of Module SMA1)

1. Calculate the lengths of the diagonals:

$$l_{13} = \sqrt{x_3^2 + y_3^2}, \quad (48)$$

$$l_{24} = \sqrt{(x_4 - x_2)^2 + y_4^2}, \quad (49)$$

2. Calculate the unit vectors along the diagonals:

$$u_1 = u_3 = \frac{x_3}{l_{13}}, \quad (50)$$

$$v_1 = v_3 = \frac{y_3}{l_{13}}, \quad (51)$$

$$u_2 = u_4 = \frac{x_4 - x_2}{l_{24}}, \quad (52)$$

$$v_2 = v_4 = \frac{y_4}{l_{24}}. \quad (53)$$

3. The loads along the diagonals in terms of the average shear stress along side 1 are:

$$A_1 = - \frac{x_2 y_4 l_{13}}{2(x_4 y_3 - x_3 y_4)}, \quad (54)$$

STRUCTURAL ELEMENT DESCRIPTIONS

$$A_2 = \frac{x_2 y_3^2 - x_4 y_3^2}{2(x_4 y_3 - x_3 y_4 - x_2(y_3 - y_4))} , \quad (55)$$

$$A_3 = -A_1 , \quad (56)$$

$$A_4 = -A_2 . \quad (57)$$

4. The loads at grid point i in terms of the displacements at grid point j may be expressed in terms of a (3×3) matrix $[k_{ij}]$ where

$$[k_{ij}] = \frac{A_i A_j}{2L} [T_i]^T [T_e] \begin{Bmatrix} u_i \\ v_i \end{Bmatrix} \{u_j v_j\}^T [T_e]^T [T_j] , \quad (58)$$

$$i = 1, 2, 3, 4 ,$$

$$j = 1, 2, \dots, i .$$

5. The 3×3 matrices are related only to deflections and forces. The terms in the 6×6 matrices, $[K_{ij}]$, corresponding to rotations are zero. Expand the matrices to 6×6 :

$$[K_{ij}] = \begin{bmatrix} k_{ij} & 0 \\ 0 & 0 \end{bmatrix} . \quad (59)$$

The element structural damping matrix is equal to g_e , the structural damping coefficient, multiplied by the stiffness matrix, $[K_{ij}]$.

8.3.5 TWIST Element Stiffness Matrix Generation (Subroutine KPANEL of Module SMA1)

The following data for the SHEAR panel element are used for generation of the element stiffness matrix for the TWIST panel.

$[T_e]$	The 3x2 transformation matrix (Equation 12).
x_2, x_3, x_4, y_3, y_4	The locations of the corners in the element system (Equations 14, 15 and 16).
Z	The energy coefficient (Equation 28, 33, 36, or 47).
$\begin{Bmatrix} u_1 u_2 u_3 u_4 \\ v_1 v_2 v_3 v_4 \end{Bmatrix}$	Unit vector coefficients at the corners (Equations 50 through 53)
A_1, A_2, A_3, A_4	Load coefficients for the corners (Equations 54 through 57)
$[T_1], [T_2], [T_3], [T_4]$	3x3 global-to-basic transformation matrices

1. Generate the three by three matrices relating the moments at point i to the rotations at point j :

$$[q_{ij}] = \frac{A_i A_j t^2}{24Z} [T_i]^T [T_e] \begin{Bmatrix} -v_i \\ u_i \end{Bmatrix} \{-v_j u_j\}^T [T_e]^T [T_j] . \quad (60)$$

These are generated only for one point i (the pivot point) and $j = 1, 2, 3$ and 4.

2. The 3x3 matrices $[q_{ij}]$ are expanded to 6x6 matrices $[K_{ij}]$ having zeros in the translational displacement rows and columns.

$$[K_{ij}] = \begin{bmatrix} 0 & r & 0 \\ \vdots & \vdots & \vdots \\ 0 & q_{ij} & 0 \end{bmatrix} . \quad (61)$$

8.3.6 Mass Matrix Generation (Subroutine MASSTQ of Module SMA2)

The mass at each point is determined by cutting the quadrilateral into four overlapping triangles. Each triangle is defined by three of the four points as follows:

STRUCTURAL ELEMENT DESCRIPTIONS

Triangle No.	Connected Points		
<u>K</u>	<u>j1</u>	<u>j2</u>	<u>j3</u>
I	4	1	2
II	1	2	3
III	2	3	4
IV	3	4	1

The area of each triangle is determined by the equation :

$$A_K = \frac{1}{2} | (\{V_{oj2}\} - \{V_{oj1}\}) \times (\{V_{oj3}\} - \{V_{oj1}\}) | , \quad (62)$$

where $\{V_{oj1}\}$ is the location vector of the first point defining the triangle, $\{V_{oj2}\}$ the second point, and $\{V_{oj3}\}$ the third point.

The mass of each triangle is divided equally among its connected points. The mass at each point is :

$$m_1 = \frac{(\mu + \rho t)}{3} (A_4 + A_1 + A_2) , \quad (63)$$

$$m_2 = \frac{(\mu + \rho t)}{3} (A_1 + A_2 + A_3) , \quad (64)$$

$$m_3 = \frac{(\mu + \rho t)}{3} (A_2 + A_3 + A_4) , \quad (65)$$

$$m_4 = \frac{(\mu + \rho t)}{3} (A_3 + A_4 + A_1) . \quad (66)$$

For each point a six by six diagonal mass matrix is constructed. The matrix is:

$$[M_{ij}] = \begin{bmatrix} m_i & & & & & \\ & m_i & & & & \\ & & m_i & & & \\ & & & 0 & & \\ & & & & 0 & \\ & & & & & 0 \end{bmatrix} \quad (67)$$

8.3.7 SHEAR Element Stress and Force Calculations (Subroutines SPANL1 and SPANL2 of Module SDR2)

The stress and force calculations are performed in two phases: phase 1 in SPANL1; phase 2 in SPANL2.

PHASE 1

1. Calculate the 1 by 3 matrices $[S_i]$, $i = 1, 2, 3, 4$:

$$[S_i] = -\frac{A_i}{2Zt} \{u_i \quad v_i\} [T_e]^T [T_i], \quad (68)$$

where A_i , Z , t , u_i , v_i , $[T_e]$ and $[T_i]$ are as given in Sections 8.3.2 and 8.3.3.

2. The $[S]$ terms and the following parameters:

$$A_1, A_2, t, \frac{p_2}{p_1}, \frac{p_1 p_2}{p_3^2}, \frac{p_1 p_2}{p_4^2},$$

are saved on a scratch file for phase 2 calculations. p_i , where $i = 1, 2, 3$ and 4 are calculated using the equations in Section 8.3.3.

PHASE 2

1. The average stress along side 1 is:

$$\bar{s}_1 = \sum_{i=1}^4 [s_i] \{u_i^t\}. \quad (69)$$

$\{u_i^t\}$ are the translational vectors where:

$$\{u_{gi}\} \rightarrow \begin{pmatrix} u_i^t \\ \cdots \cdots \cdots \\ u_i^r \end{pmatrix}. \quad (70)$$

2. The stresses on the corners are:

$$\tau_1 = \frac{p_2}{p_1} \bar{s}_1, \quad (71)$$

$$\tau_2 = \frac{p_1}{p_2} \bar{s}_1, \quad (72)$$

$$\tau_3 = \frac{p_1 p_2}{p_3^2} \bar{s}_1, \quad (73)$$

$$\tau_4 = \frac{p_1 p_2}{p_4^2} \bar{s}_1. \quad (74)$$

3. The average and maximum stresses are defined as:

$$\tau_{avg} = \frac{1}{4} (\tau_1 + \tau_2 + \tau_3 + \tau_4), \quad (75)$$

$$\tau_{max} = \max (|\tau_1|, |\tau_2|, |\tau_3|, |\tau_4|). \quad (76)$$

4. The margin of safety in shear is defined by :

$$M.S._s = \begin{cases} \frac{\sigma_s}{\tau_{\max}} - 1, & \text{if } \sigma_s > 0 \\ \text{Integer "1",} & \text{if } \sigma_s \leq 0 \text{ or } \tau_{\max} = 0 \end{cases} \quad (77)$$

5. The net loads on the corners in the diagonal direction are :

$$P_{13} = A_1 \bar{s}_1 t, \quad (78)$$

$$P_{24} = A_2 \bar{s}_1 t. \quad (79)$$

8.3.8 TWIST Element Stress and Force Calculations (Subroutines SPANL1 and SPANL2 of Module SDR2)

The stress and force calculations are performed in two phases, as for the SHEAR panel element.

PHASE 1

1. Calculate for $i = 1, 2, 3, 4$

$$[S_i] = -\frac{A_i}{4Z} \{-v_i : u_i\} [T_e]^T [T_i]. \quad (80)$$

2. The $[S_i]$ terms and the following data:

$$A_1, A_2, t, \frac{p_2}{p_1}, \frac{p_1 p_2}{p_3^2}, \frac{p_1 p_2}{p_4^2}.$$

are saved on a scratch file for phase 2 calculations.

PHASE 2

1. The mean outer fibre shear stress along side 1 is:

$$\tau_1 = \frac{4}{\sum_{i=1}^4} [S_i] \{u_i^r\}, \quad (81)$$

STRUCTURAL ELEMENT DESCRIPTIONS

where $\{u_i^r\}$ are the three rotational displacements:

$$\{u_{gi}\} \Rightarrow \begin{Bmatrix} u_i^t \\ -\frac{u_i^r}{t} \end{Bmatrix}.$$

2. The stresses are :

$$\sigma_1 = \frac{p_2}{p_1} \tau_1, \quad (82)$$

$$\sigma_2 = \frac{\tau_1 p_1}{p_2}, \quad (83)$$

$$\sigma_3 = \frac{p_1 p_2}{p_3^2} \tau_1, \quad (84)$$

$$\sigma_4 = \frac{p_1 p_2}{p_4^2} \tau_1, \quad (85)$$

$$\sigma_{avg} = \frac{1}{4} (\sigma_1 + \sigma_2 + \sigma_3 + \sigma_4), \quad (86)$$

$$\sigma_{max} = \max(|\sigma_1|, |\sigma_2|, |\sigma_3|, |\sigma_4|). \quad (87)$$

3. The margin of safety in shear is defined by :

$$M.S._s = \begin{cases} \frac{\sigma_s}{|\sigma_{max}|} - 1, & \sigma_s > 0 \\ \text{Integer "1",} & \sigma_s \leq 0 \text{ or } \sigma_{max} = 0 \end{cases} \quad (88)$$

4. The moments are :

$$M_{13} = \frac{A_1 t^2}{6} \tau_1. \quad (89)$$

$$M_{24} = \frac{A_2 t^2}{6} \tau_1 \quad (90)$$

8.3.9 SHEAR Panel Differential Stiffness Calculations (Subroutine DSHEAR of Module DSMG1)

1. Data

The data necessary for analysis are included in the ECPT, CSTM, MPT and UGV data blocks. The following data are generated as in Sections 8.3.2, 8.3.3, and 8.3.4.

- a. $[T_i]$, $i = 1, 2, 3, 4$, the 3×3 transformation matrices between global and basic coordinates, at the four corners of the shear panel.
- b. $[T_e]$, the 3×2 transformations between basic and element coordinates.
- c. $\{k\}$, the unit vector normal to the plane in basic coordinates.
- d. u_i, v_i , $i = 1, 2, 3, 4$, the unit vectors along the diagonals in element coordinates.
- e. A_i , $i = 1, 2, 3, 4$, the load coefficients for the corners.
- f. Z , the energy coefficient for the panel.
- g. l_{13} and l_{24} , the lengths of the diagonals.

2. Algorithm

- a. The load in the diagonal between points 1 and 3 is :

$$F_{13} = - \frac{A_1}{2Z} \sum_{i=1}^4 A_i \{u_i \mid v_i\}^T [T_e]^T [T_i] \begin{Bmatrix} x_{1i} \\ x_{2i} \\ x_{3i} \end{Bmatrix} \quad (91)$$

STRUCTURAL ELEMENT DESCRIPTIONS

where $\{x_i\}$ is the vector of the three translations in global coordinates for point (i).

b. The load in the other diagonal is :

$$F_{24} = \frac{A_2}{A_1} F_{13} . \quad (92)$$

c. Construct a perpendicular, which is defined in basic coordinates, to each diagonal vector in the plane of the panel.

$$\{j_1\} = [T_e] \begin{Bmatrix} -v_1 \\ u_1 \end{Bmatrix} , \quad (93)$$

$$\{j_2\} = [T_e] \begin{Bmatrix} -v_2 \\ u_2 \end{Bmatrix} . \quad (94)$$

d. The nonzero partitions of the overall differential stiffness matrix for the displacements are:

$$[k_{11}^d] = F_{13}' [T_1]^T [\{j_1\}\{j_1\}^T + \{k\}\{k\}^T] [T_1] , \quad (95)$$

$$[k_{13}^d] = -F_{13}' [T_1]^T [\{j_1\}\{j_1\}^T + \{k\}\{k\}^T] [T_3] , \quad (96)$$

$$[k_{33}^d] = F_{13}' [T_3]^T [\{j_1\}\{j_1\}^T + \{k\}\{k\}^T] [T_3] , \quad (97)$$

$$[k_{31}^d] = [k_{13}^d]^T , \quad (98)$$

$$[k_{22}^d] = F_{24}' [T_2]^T [\{j_2\}\{j_2\}^T + \{k\}\{k\}^T] [T_2] , \quad (99)$$

$$[k_{24}^d] = -F_{24}' [T_2]^T [\{j_2\}\{j_2\}^T + \{k\}\{k\}^T] [T_4] , \quad (100)$$

$$[k_{44}^d] = F_{24}' [T_4]^T \left[\{j_2\} \{j_2\}^T + \{k\} \{k\}^T \right] [T_4], \quad (101)$$

$$[k_{42}^d] = [k_{24}^d]^T, \quad (102)$$

where :

$$F_{13}' = \frac{F_{13}}{x_{13}}, \quad (103)$$

$$F_{24}' = \frac{F_{24}}{x_{24}}. \quad (104)$$

5. The actual 6x6 partitions are

$$[K_{ij}^d] = \begin{bmatrix} k_{ij}^d & 0 \\ - & - \\ 0 & 0 \end{bmatrix}, \quad (105)$$

and

$$[K_{ji}^d] = [K_{ij}^d]^T. \quad (106)$$

STRUCTURAL ELEMENT DESCRIPTIONS

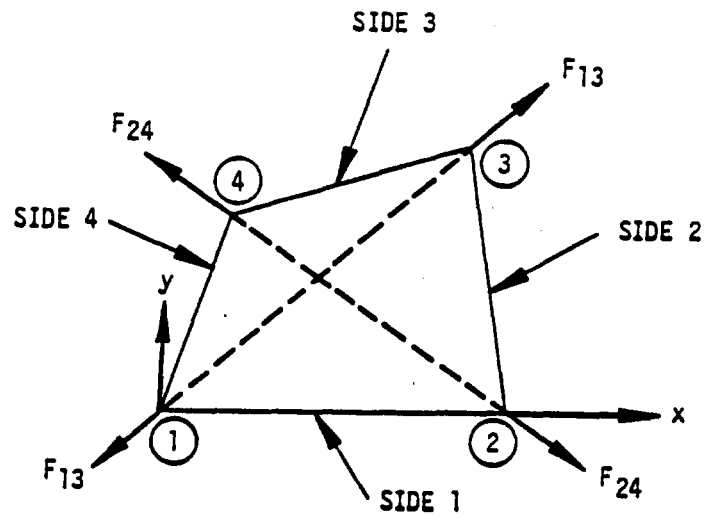


Figure 2. Shear panel element coordinate system and element forces.

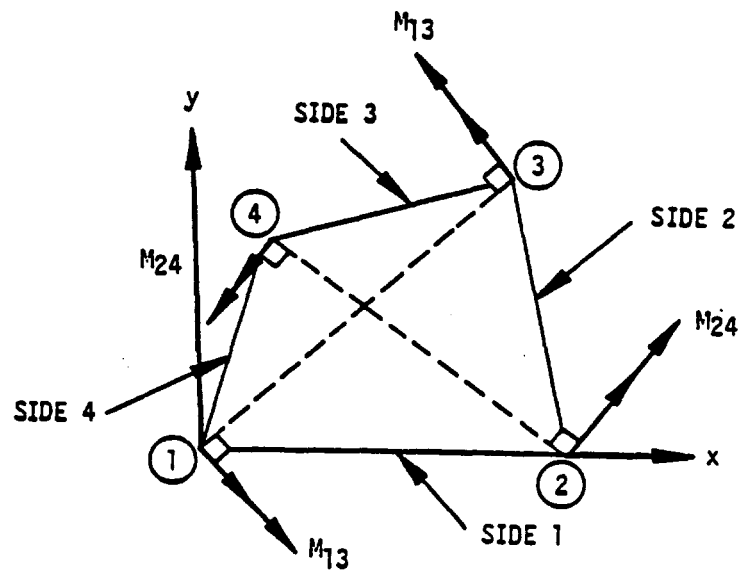


Figure 3. Twist panel element coordinate system and element forces.

8.4 TRMEM AND QDMEM ELEMENTS

8.4.1 Input Data for the TRMEM and QDMEM Elements

1. ECPT entries for the TRMEM and QDMEM are:

<u>Symbol</u>	<u>Description</u>
<u>TRMEM</u>	<u>QDMEM</u>
SIL ₁	SIL ₁
SIL ₂	SIL ₂
SIL ₃	SIL ₃
	SIL ₄
	Scalar indices of the connected grid points.
N _i	N _i
X _i	X _i
i = 1,3	i = 1,4
Y _i	Y _i
Z _i	Z _i
	Local coordinate system numbers and location coordinates in the basic system for the connected grid points.
θ	Anisotropic material orientation angle
Mat I. D.	Material identification number
t	Thickness
μ	Nonstructural mass per unit area
t _μ	Temperature for material properties

2. Coordinate system data

The numbers N_i, X_i, Y_i and Z_i are used to calculate the 3 by 3 global-to-basic coordinate transformation matrices [T_i] for points i = 1, 2, 3, and 4.

STRUCTURAL ELEMENT DESCRIPTIONS

3. Material data

<u>Symbol</u>	<u>Description</u>
$[G_e]$	3x3 stress-strain matrix
ρ	Mass density
$\alpha_x, \alpha_y, \alpha_{xy}$	Three thermal expansion coefficients
T_0	Reference temperature
g_e	Structural damping coefficient
$\sigma_t, \sigma_c, \sigma_s$	Stress limits for tension, compression and shear

8.4.2 Basic Equations For TRMEM

1. The element coordinate system is defined by the following equations (see Figure 4)

$$\{V_{12}\} = \begin{Bmatrix} x_2 - x_1 \\ y_2 - y_1 \\ z_2 - z_1 \end{Bmatrix}, \quad (1)$$

$$\{V_{13}\} = \begin{Bmatrix} x_3 - x_1 \\ y_3 - y_1 \\ z_3 - z_1 \end{Bmatrix}, \quad (2)$$

$$\{i\} = \frac{\{V_{12}\}}{|\{V_{12}\}|}, \quad (3)$$

$$\{k\} = \frac{\{i\} \times \{V_{13}\}}{|\{i\} \times \{V_{13}\}|}, \quad (4)$$

$$\{j\} = \{k\} \times \{i\}. \quad (5)$$

2. The displacement transformation matrix from basic coordinates to in-plane coordinates is :

$$[E]^T = \begin{bmatrix} i_1 & i_2 & i_3 \\ j_1 & j_2 & j_3 \end{bmatrix}, \quad (6)$$

3. The coordinates of the points in the element coordinate system are :

$$x_1 = y_1 = y_2 = 0, \quad (7)$$

$$x_2 = |\{V_{12}\}|, \quad (8)$$

$$x_3 = \{V_{13}\}^T \{i\}, \quad (9)$$

$$y_3 = |\{i\} \times \{V_{13}\}|. \quad (10)$$

The area is :

$$A = \frac{1}{2} x_2 y_3. \quad (11)$$

4. The transformations from displacements at the points to strains are :

$$[C_1] = \begin{bmatrix} -\frac{1}{x_2} & 0 \\ 0 & \frac{1}{y_3} \left(\frac{x_3}{x_2} - 1 \right) \\ \frac{1}{y_3} \left(\frac{x_3}{x_2} - 1 \right) & -\frac{1}{x_2} \end{bmatrix}, \quad (12)$$

$$[C_2] = \begin{bmatrix} \frac{1}{x_2} & 0 \\ 0 & -\frac{x_3}{x_2 y_3} \\ -\frac{x_3}{x_2 y_3} & \frac{1}{x_2} \end{bmatrix}, \quad (13)$$

$$[C_3] = \begin{bmatrix} 0 & 0 \\ 0 & \frac{1}{y_3} \\ \frac{1}{y_3} & 0 \end{bmatrix}. \quad (14)$$

8.4.3 Stiffness Matrix Calculation for TRMEM (Subroutine KTRMEM of Module SMA1)

1. The equation used in the stiffness matrix generation in global coordinates is:

$$[k_{ij}] = At([C_i][E]^T[T_i])^T[G_e]([C_j][E]^T[T_j]), \quad (15)$$

where "i" is the pivot point number, and $j = 1, 2, 3$ are the three connected points. $[k_{ij}]$ is a 3x3 matrix.

2. For use in the overall structural matrix, the matrices are expanded to 6x6 to form:

$$[K_{ij}] = \begin{bmatrix} k_{ij} & 0 \\ 0 & 0 \end{bmatrix}. \quad (16)$$

8.4.4 Mass Matrix Calculation for the TRMEM Element (Subroutine MASSTQ of Module SMA2)

The mass is generated by the following algorithm.

The vectors defining the sides are :

$$\{V_{12}\} = \begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \\ z_2 - z_1 \end{pmatrix}, \quad (17)$$

$$\{V_{13}\} = \begin{pmatrix} x_3 - x_1 \\ y_3 - y_1 \\ z_3 - z_1 \end{pmatrix}. \quad (18)$$

The area is :

$$A = \frac{1}{2} |\{V_{12}\} \times \{V_{13}\}|. \quad (19)$$

The mass at each point is :

$$m = \frac{A}{3} (\rho t + \mu), \quad (20)$$

which is one-third of the total mass.

For each point the diagonal mass matrix is :

$$[M_{ij}] = \begin{bmatrix} m & & & \\ & m & & \\ & & m & \\ & & & 0 \end{bmatrix}, \quad i = 1, 2, 3. \quad (21)$$

8.4.5 Element Load Calculations For The TRMEM Element (Subroutine TRIMEM of Module SSG1)

Using the loading temperature on the element, \bar{T} , given in the GPTT data block, the triangular membrane routine generates force vectors by the equation:

$$\{P_i\} = At[T_i]^T[E][C_i]^T[G_e]\{\alpha\}(\bar{T} - T_0), \quad i = 1, 2, 3 \quad (22)$$

where $\{P_i\}$ is a 3x1 vector.

The forces are placed in the PG load vector data block.

8.4.6 Element Stress Calculations For The TRMEM Element (Subroutines STRME1 and STOME2 of Module SDR2)

1. Calculations performed in STRME1 (Phase 1 calculations).

a. Using the formulae given in Section 8.4.2, calculate the following terms:

$[C_i]$	$i = a, b, c$	(3x2)
$[T_i]$	$i = a, b, c$	(3x3)
$[E]$		(3x2)
$[G_e]$		(3x3)

The transformations from displacements to stress are :

$$[S_i] = [G_e][C_i][E]^T[T_i] \quad (24)$$

Equation 23 is intentionally missing.

The temperature to stress relation is:

$$\{S_t\} = - [G_e]\{\alpha\} \quad (25)$$

where

$$\{\alpha\} = \alpha \begin{Bmatrix} 1 \\ 1 \\ 0 \end{Bmatrix}, \quad (26)$$

for isotropic materials. $\{\alpha\}$ is input by the user for anisotropic materials and corrected for material angle by $\alpha = [v]\{\alpha_m\}$.

2. Calculations performed by STQME2 (Phase 2 calculations)

The equation for stress is:

$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \sigma_{xy} \end{Bmatrix} = \left[\sum_{i=a,b,c} [S_i]\{u_{gi}\} \right] + \{S_t\} [\bar{T} - T_0], \quad (27)$$

where \bar{T} is the loading temperature obtained from the GPTT data block.

The principal stresses are:

$$\sigma_1 = \left(\frac{\sigma_x + \sigma_y}{2} \right) + \sqrt{\left(\frac{\sigma_x - \sigma_y}{2} \right)^2 + \sigma_{xy}^2}, \quad (28)$$

$$\sigma_2 = \left(\frac{\sigma_x + \sigma_y}{2} \right) - \sqrt{\left(\frac{\sigma_x - \sigma_y}{2} \right)^2 + \sigma_{xy}^2}, \quad (29)$$

$$\theta = \frac{1}{2} \arctan \left(\frac{2\sigma_{xy}}{\sigma_x - \sigma_y} \right) \quad (\text{in degrees}), \quad (30)$$

where θ is limited to : $-90^\circ \leq \theta \leq 90^\circ$

The maximum shear is :

$$\tau = \sqrt{\left(\frac{\sigma_x - \sigma_y}{2}\right)^2 + \sigma_{xy}^2} \quad (31)$$

8.4.7 Differential Stiffness Matrix Calculations for the TRMEM Element (Subroutine DTRMEM of Module DSMG1)

1. Input Data.

ECPT for element

i - Pivot point scalar index

$\{u_1\}$, $\{u_2\}$, $\{u_3\}$ - Displacements of pivots on triangle (UGV)

\bar{T} - Average loading temperature of the grid points of the element (GPTT)

CSTM - coordinate systems

MPT - Material properties table

2. Output Data.

$[K_{11}^d]$, $[K_{12}^d]$, $[K_{13}^d]$ - partitions of the differential stiffness matrix.

3. Solution Algorithm.

a. The planar stresses in the element, σ_x , σ_y and σ_{xy} , are calculated as in the SDR2 (Stress Data Recovery) module. The following data are saved for use in the differential stiffness calculation :

$\{i\}$, $\{j\}$, $\{k\}$ - Unit vectors defining the element coordinate system.

A, t - Area and thickness

x_2 , x_3 , y_3 - Locations of the points, element coordinates

$[T_1]$, $[T_2]$, $[T_3]$ - Global-to-basic coordinate transformations

σ_x , σ_y , σ_{xy} - Stresses in element system

TRMEM AND QDMEM ELEMENTS

b. The differential stiffness matrix in terms of the six generalized coordinates ($\omega_x, \omega_y, \omega_z, \epsilon_{xx}, \epsilon_{yy},$ and ϵ_{xy}) is:

$$[K_g^d] = At \begin{bmatrix} \sigma_y & -\sigma_{xy} & 0 & 0 & 0 & 0 \\ -\sigma_{xy} & \sigma_x & 0 & 0 & 0 & 0 \\ 0 & 0 & (\sigma_x + \sigma_y) & -\sigma_{xy} & \sigma_{xy} & (\sigma_x - \sigma_y) \\ 0 & 0 & -\sigma_{xy} & 0 & 0 & 0 \\ 0 & 0 & \sigma_{xy} & 0 & 0 & 0 \\ 0 & 0 & (\sigma_x - \sigma_y) & 0 & 0 & 0 \end{bmatrix} \quad (32)$$

If the subroutine is called from the DTRIA or DQUAD routines the following terms are set to zero:

$$[K_{g11}^d] = [K_{g12}^d] = [K_{g21}^d] = [K_{g22}^d] = 0$$

c. The transformation matrices from displacements at the points to generalized coordinates are:

$$[C_1^d] = \begin{bmatrix} 0 & 0 & \gamma_3 - \gamma_2 \\ 0 & 0 & \gamma_1 \\ \frac{\gamma_2 - \gamma_3}{2} & -\frac{\gamma_1}{2} & 0 \\ -\gamma_1 & 0 & 0 \\ 0 & \gamma_3 - \gamma_2 & 0 \\ \frac{\gamma_2 - \gamma_3}{2} & -\frac{\gamma_1}{2} & 0 \end{bmatrix} \quad (33)$$

STRUCTURAL ELEMENT DESCRIPTIONS

$$[c_2^d] = \begin{bmatrix} 0 & 0 & -\gamma_3 \\ 0 & 0 & -\gamma_1 \\ \frac{\gamma_3}{2} & \frac{\gamma_1}{2} & 0 \\ \gamma_1 & 0 & 0 \\ 0 & -\gamma_3 & 0 \\ \frac{\gamma_3}{2} & \frac{\gamma_1}{2} & 0 \end{bmatrix} \quad (34)$$

$$[c_3^d] = \begin{bmatrix} 0 & 0 & \gamma_2 \\ 0 & 0 & 0 \\ -\frac{\gamma_2}{2} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \gamma_2 & 0 \\ \frac{\gamma_2}{2} & 0 & 0 \end{bmatrix} \quad (35)$$

where

$$\gamma_1 = \frac{1}{x_2}, \quad (36)$$

$$\gamma_2 = \frac{1}{y_3}, \quad (37)$$

$$\gamma_3 = \frac{x_3}{x_2 y_3}. \quad (38)$$

d. The partitions (3x3) of the differential stiffness matrix in global coordinates are :

$$[K_{ij}] = ([C_i^d] [E^d]^T [T_i])^T [K_\omega^d] [C_j^d] [E^d]^T [T_j], \quad (39)$$

i = pivot point

j = 1, 2 and 3

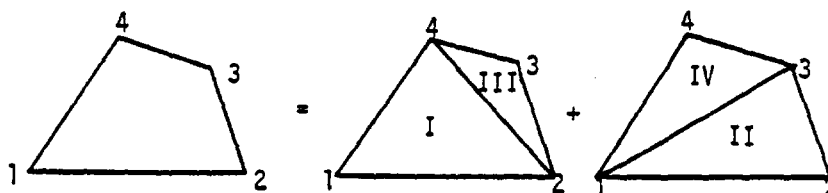
where

$$[E^d]^T = \begin{bmatrix} \{i\}^T \\ \{j\}^T \\ \{k\}^T \end{bmatrix} \quad (3 \times 3). \quad (40)$$

8.4.8 General Calculations for the QDMEM by the QDMEM Driver Routines (Subroutines KQDMEM of Module SMA1, SQDMEM of Module SDR2, DQDMEM of Module DSMG1).

1. The quadrilateral is divided into four triangles as shown in the figure below:

STRUCTURAL ELEMENT DESCRIPTIONS



The thickness used for each triangle is one-half that given for the quadrilateral. Since no special calculation time is saved by generating a unique element coordinate system, the basic locations of the points are used to calculate individual coordinate systems for the triangles.

An integer mapping matrix $[M]$ containing the quadrilateral point numbers is used to convert point numbers for the triangles to point numbers for the quadrilateral.

$$[M] = \begin{matrix} & \begin{matrix} \text{Triangle Point No.} \\ \begin{matrix} \underline{a} & \underline{b} & \underline{c} \end{matrix} \\ \begin{matrix} 1 & 2 & 4 \\ 2 & 3 & 1 \\ 3 & 4 & 2 \\ 4 & 1 & 3 \end{matrix} \end{matrix} & \begin{matrix} \text{Triangle No.} \\ \begin{matrix} \text{(I)} \\ \text{(II)} \\ \text{(III)} \\ \text{(IV)} \end{matrix} \end{matrix} \end{matrix} \quad (41)$$

The data corresponding to the point numbers in each row of the matrix are transferred to the triangular membrane routine. The pivot grid point 1 is also transferred.

2. Material orientation for subtriangles.

The material orientation angle for the QDMEM element must be transformed to a set of angles related to the base of each subtriangle. This requires the following steps:

1. The element coordinate system is defined as follows:

$$\{V_i\} = \begin{Bmatrix} X_i \\ Y_i \\ Z_i \end{Bmatrix} \quad i = 1, 2, 3, 4, \quad (42)$$

$$\{d_{21}\} = \{V_2\} - \{V_1\} , \quad (43)$$

$$\{i\} = \frac{\{d_{21}\}}{|\{d_{21}\}|} , \quad (44)$$

$$\{d_{41}\} = \{V_4\} - \{V_1\} , \quad (45)$$

$$\{k\} = \frac{\{i\} \times \{d_{41}\}}{|\{i\} \times \{d_{41}\}|} , \quad (46)$$

$$\{j\} = \{k\} \times \{i\} . \quad (47)$$

The material is oriented for each triangle as follows:

$$s_1 = \sin(\theta) , \quad (48)$$

$$c_1 = \cos(\theta) , \quad (49)$$

$$\{p\} = c_1 \{i\} + s_1 \{j\} , \quad (50)$$

$$\{V_{II}^t\} = \{V_3\} - \{V_2\} , \quad (51)$$

$$\{V_{III}^t\} = \{V_4\} - \{V_3\} , \quad (52)$$

$$\{V_{IV}^t\} = \{V_1\} - \{V_4\} , \quad (53)$$

$$c_i = \frac{\{V_i^t\}^T \{p\}}{|\{V_i^t\}|} = \cos(\theta_i), \quad (54)$$

$i = \text{II, III and IV}$

$$s_i = \frac{(\{V_i^t\} \times \{p\})^T \{k\}}{|\{V_i^t\}|} = \sin(\theta_i). \quad (55)$$

The values s_i and c_i may be passed to the triangular membrane subroutines in lieu of the angles θ_i .

8.4.9 Stiffness Matrix Calculations for the QDMEM.

Three stiffness matrices, which the triangular membrane routine calculates for each sub-triangle, are added to the four matrices which will be output. (Note: only three triangles are needed for each pivot point.) For example, consider the case where point 2 is the pivot grid point. (i.e., the second SIL value in the grid point connection list equals the pivot grid point L value). Triangle I is calculated by entering the geometry and property data for the 1, 2 and 4 points on the quadrilateral, with number 2 as the pivot point. The outputs from the stiffness matrix generation routines for the TRMEM are:

$$[K_{21}], [K_{22}], [K_{24}]$$

Data for triangles II and III are also entered, and their corresponding matrix partitions are added. Triangle number IV is not connected to point 2.

8.4.10 Element Stress Calculations for the QDMEM (Subroutine SQDME1 and STQME2 of Module SDR2).

The solution for stress in the quadrilateral involves two phases. In the first phase (SQDME1) the triangular membrane partitions are solved for their stress-displacement matrices. These matrices are modified to correspond to the element coordinate system. They

are added together to form four 3x3 matrices relating displacements in global coordinates to element stress. A vector is also calculated which transforms temperature to stress.

The second phase (STQME2) involves the acquisition of the displacement and temperature data and the calculation of the net stress.

The following steps are used to set up an element coordinate system and obtain triangle to element stress transformations.

Phase 1

1. The following quantities are calculated:

$$\{V_i\} = \begin{Bmatrix} X_i \\ Y_i \\ Z_i \end{Bmatrix} \quad i = 1, 2, 3, 4, \quad (56)$$

$$\{d_1\} = \{V_3\} - \{V_1\}, \quad (57)$$

$$\{d_2\} = \{V_4\} - \{V_2\}, \quad (58)$$

$$\{k\} = \frac{\{d_1\} \times \{d_2\}}{|\{d_1\} \times \{d_2\}|}, \quad (59)$$

$$\{a_{12}\} = \{V_2\} - \{V_4\}, \quad (60)$$

$$\{a_{23}\} = \{V_3\} - \{V_2\}, \quad (61)$$

$$\{a_{41}\} = \{V_1\} - \{V_4\}, \quad (62)$$

$$\{a_{34}\} = \{V_4\} - \{V_3\}, \quad (63)$$

STRUCTURAL ELEMENT DESCRIPTIONS

$$h = \{a_{12}\}^T \{k\} . \quad (64)$$

h is the perpendicular distance between the diagonals. The mean plane of the element lies halfway between the diagonals.

2. The unit vectors along the edges of the four triangles, projected on the mean plane, are calculated from:

$$x_2 = |\{a_{12}\} - h\{k\}| , \quad (65)$$

$$\{i\} = \frac{\{a_{12}\} - h\{k\}}{x_2} , \quad (66)$$

$$\{j\} = \{k\} \times \{i\} , \quad (67)$$

$$[R] = \begin{bmatrix} \{i\}^T \\ \{j\}^T \end{bmatrix} \quad (2 \times 3) , \quad (68)$$

$$\{v_{12}\} = \begin{Bmatrix} 1 \\ 0 \end{Bmatrix} , \quad (69)$$

$$\{v_{ij}\} = [R]\{a_{ij}\} \quad ij = 23, 34, 41 , \quad (70)$$

$$\{w^I\} = \begin{Bmatrix} 1 \\ 0 \end{Bmatrix} , \quad (71)$$

$$\{w^{II}\} = \frac{1}{|\{v_{23}\}|} \{v_{23}\} , \quad (72)$$

$$\{w^{III}\} = \frac{1}{|\{v_{34}\}|} \{v_{34}\} , \quad (73)$$

$$\{w^{IV}\} = \frac{1}{|\{v_{41}\}|} \{v_{41}\} , \quad (74)$$

3. For each triangular membrane, $\beta = I, II, III, IV$, of the quadrilateral, the subroutine STRME1 is called to calculate the three stress functions $[S_i]$ where

$$\begin{pmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_{12} \end{pmatrix}_{\text{triangle}} = \sum_{i=1}^3 [S_i] \{u_{gdi}\}, \quad (75)$$

where $\{u_{gdi}\}$ are the displacements in global coordinates of the points on the triangle.

$[S_i^\beta]$ is calculated using the full thickness of the panel for triangle β .

4. The stress functions, Equation 24, are transformed to the element coordinates by the matrix $[T^\beta]$:

$$[S_i^\beta]_{\text{element}} = [T^\beta][S_i^\beta], \quad (76)$$

where

$$[T^\beta] = \begin{bmatrix} w_1^2 & w_2^2 & -2w_1w_2 \\ w_2^2 & w_1^2 & 2w_1w_2 \\ w_1w_2 & -w_1w_2 & w_2^2 - w_1^2 \end{bmatrix}, \quad (77)$$

and

$$\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \{w^\beta\} \quad (78)$$

for triangle $\beta = \text{I, II, III, IV}$.

5. Using the mapping matrix $[M]$, Equation 41, the matrices are added. The actual equations are:

$$[\bar{S}_1] = \frac{1}{4}([S_a^I] + [S_c^{II}] + [S_b^{IV}]) \quad (79)$$

$$[\bar{S}_2] = \frac{1}{4}([S_b^I] + [S_a^{II}] + [S_c^{III}]) \quad (80)$$

$$[\bar{S}_3] = \frac{1}{4}([S_b^{II}] + [S_a^{III}] + [S_c^{IV}]) , \quad (81)$$

$$[\bar{S}_4] = \frac{1}{4}([S_c^I] + [S_b^{III}] + [S_a^{IV}]) , \quad (82)$$

$$\{\bar{S}_t\} = \frac{1}{4}(\{S_t^I\} + \{S_t^{II}\} + \{S_t^{III}\} + \{S_t^{IV}\}) , \quad (83)$$

where $[S_i^B]$ $i = a, b, c$, are the stress matrices in Equation 76 and $\{S_t^B\}$ is the vector in Equation 25.

Phase 2

1. In phase 2 the stresses are calculated from:

$$\begin{pmatrix} \sigma_x \\ \sigma_y \\ \sigma_{xy} \end{pmatrix} = \{\bar{S}_t\} \bar{T} + \sum_{i=1}^4 [\bar{S}_i] \{u_{qdi}\} , \quad (84)$$

where

$$\bar{T} = \frac{1}{4} \sum_{i=1}^4 (t_i - T_0) , \quad (85)$$

and t_i are the loading temperatures at the grid points, obtained from the GPTT data block.

2. The principal stresses and the angles are calculated in exactly the same manner as for the TRMEM element.

8.4.11 Mass Matrix Generation for the QDMEM Element (Subroutine MASSTQ of Module SMA2).

The mass is generated by the following algorithm.

The vectors defining the sides and the diagonals are:

$$\{V_{ij}\} = \begin{pmatrix} X_j - X_i \\ Y_j - Y_i \\ Z_j - Z_i \end{pmatrix} , \quad ij = 12, 23, 34, 41, 13, 24 \quad (86)$$

The area of the subtriangles defined by the integer mapping matrix [M], Equation 41, is:

$$A_I = \frac{1}{2} |\{V_{14}\} \times \{V_{12}\}| , \quad (87)$$

$$A_{II} = \frac{1}{2} |\{V_{12}\} \times \{V_{23}\}| , \quad (88)$$

$$A_{III} = \frac{1}{2} |\{V_{23}\} \times \{V_{34}\}| , \quad (89)$$

$$A_{IV} = \frac{1}{2} |\{V_{34}\} \times \{V_{14}\}| . \quad (90)$$

The area of the quadrilateral is:

$$A_q = \frac{1}{2} |\{V_{13}\} \times \{V_{24}\}| . \quad (91)$$

The mass at each point is:

$$m_1 = \frac{(A_I + A_q)}{3} (\mu + pt) , \quad (92)$$

$$m_2 = \frac{(A_{II} + A_q)}{3} (\mu + pt) , \quad (93)$$

$$m_3 = \frac{(A_{III} + A_q)}{3} (\mu + pt) , \quad (94)$$

$$m_4 = \frac{(A_{IV} + A_q)}{3} (\mu + pt) . \quad (95)$$

For each point, the diagonal mass matrix is:

$$[M_{ii}] = \begin{bmatrix} \overline{m}_1 & & & \\ & m_1 & & \\ & & m_i & \\ & & & 0 \\ 0 & & & & 0 \\ & & & & & 0 \end{bmatrix} , \quad i = 1, 2, 3, 4 . \quad (96)$$

STRUCTURAL ELEMENT DESCRIPTIONS

8.4.12 Thermal Load Computation For The QDMEM.

The thermal loads are calculated using the triangular membrane routine. The EST data are rearranged to correspond to each of the four subtriangles, and each triangle produces a load in global coordinates.

8.4.13 Differential Stiffness Computations For The QDMEM (Subroutines DODMEM and DTRMEM of Module DSMG1).

The differential stiffness matrices for the QDMEM element are generated by rearranging the ECPT data into four sets of TRMEM data. The TRMEM differential stiffness routine calculates the stresses, generates the differential stiffness matrix partitions in global coordinates and inserts them in the overall matrix.

8.4.14 Piecewise Linear Analysis Calculations (Subroutines PSTRM and PSQDM of Module PLA3 and Subroutines PKTRM and PKQDM of Module PLA4)

The additional ECPTNL and ESTNL entries are:

ϵ_o^* - The previously computed strain value once removed.

ϵ^* - The previously computed strain value.

E^* - The previously computed modulus of elasticity.

$\left. \begin{array}{l} \sigma_x^* \\ \sigma_y^* \\ \sigma_{xy}^* \end{array} \right\}$ The previously computed membrane stresses

All of the above values are initially zero with the exception of E^* , which is initially the original modulus of elasticity present on a MAT1 card.

For both PLA3 and PLA4, the element stress matrix calculations are generated in the same manner as Equation 24 of Section 8.4.6 (Equations 79 through 82 of Section 8.4.6 for the QDMEM), with the exception that for all DMAP loops (of the Piecewise Linear Analysis Rigid Format) after the first, the 3 by 3 material properties matrix $[G_e]$ is replaced by a stress-dependent 3 by 3 material properties matrix $[G_p]$ defined as follows

$$[G_p] = E_o \begin{bmatrix} 1+s_x^2 F & -v+s_x s_y F & 2\sigma_{xy}^* s_x F \\ & 1+s_y^2 F & 2\sigma_{xy}^* s_y F \\ \text{(sym)} & & 2(1+v)+4F\sigma_{xy}^{*2} \end{bmatrix}^{-1}, \quad (97)$$

where

$$\tau_o = \left[\sigma_x^{*2} - \sigma_x^* \sigma_y^* + \sigma_y^{*2} + 3\sigma_{xy}^{*2} \right]^{1/2}, \quad (98)$$

$$F = \frac{9(E_o - E^*)}{4\tau_o^2 E^*}, \quad (99)$$

$$s_x = \frac{2\sigma_x^* - \sigma_y^*}{3}, \quad (100)$$

$$s_y = \frac{2\sigma_y^* - \sigma_x^*}{3}, \quad (101)$$

and E_0 and ν are the linear type 1 material properties. If $E^* = 0$, or $\tau_0 = 0$, then $[G_p] = [0]$.

Calculate the incremental element stresses:

$$\begin{pmatrix} \Delta\sigma_x \\ \Delta\sigma_y \\ \Delta\sigma_{xy} \end{pmatrix} = \left[\sum_{i=a,b,c} [S_i] \{\Delta u_{gi}\} \right], \quad (102)$$

where $[S_i]$ is given in Equation 24 of Section 8.4.6 and $\{\Delta u_{gi}\}$ are the 3 by 1 translational displacement vectors.

Define the element stresses for output and for updating the ECPTNL and ESTNL:

$$\sigma_{x1} = \sigma_x^* + \Delta\sigma_x, \quad (103)$$

$$\sigma_{y1} = \sigma_y^* + \Delta\sigma_y, \quad (104)$$

$$\sigma_{xy1} = \sigma_{xy}^* + \Delta\sigma_{xy}. \quad (105)$$

In PLA3, using the element stresses above, the principal stresses are calculated in the same manner as in Equations 28 through 31 in Section 8.4.6.

Estimate the next elastic coefficients as defined by the following equations:

$$\tau_1 = \left[\sigma_{x1}^2 - \sigma_{x1}\sigma_{y1} + \sigma_{y1}^2 + 3\sigma_{xy1}^2 \right]^{1/2}, \quad (106)$$

$$\epsilon_1 = f^{-1}(\tau_1), \quad (107)$$

where f is the tabular stress-strain function. (When τ_1 is outside the range of the function, define $E_1 = 0$, $\epsilon_1 = \epsilon^*$, and $\epsilon^* = \epsilon_0^*$).

Calculate:

$$\Delta \epsilon = \epsilon_1 - \epsilon^* \quad , \quad (108)$$

$$\Delta \epsilon^* = \epsilon^* - \epsilon_0^* \quad , \quad (109)$$

$$\epsilon_2 = \epsilon_1 + \gamma(\Delta \epsilon) \quad , \quad (110)$$

where γ is a load ratio parameter calculated by the module driver (PLA3 or PLA4).

Calculate:

$$\tau_2 = f(\epsilon_2) \quad , \quad (111)$$

where f is the tabular stress-strain function.

Then the estimated next modulus of elasticity, E_1 , is given by:

$$E_1 = \begin{cases} \frac{\tau_2 - \tau_1}{\epsilon_2 - \epsilon_1} , & \text{for } \epsilon_2 \neq \epsilon_1 \\ 0 & , \text{for } \epsilon_2 = \epsilon_1 . \end{cases} \quad (112)$$

The new ESTNL and ECPTNL entries are:

$$\epsilon_0^* = \epsilon^* \quad , \quad (113)$$

$$\epsilon^* = \epsilon_1 \quad , \quad (114)$$

$$E^* = E_1 \quad , \quad (115)$$

$$\sigma_x^* = \sigma_{x1} \quad , \quad (116)$$

$$\sigma_y^* = \sigma_{y1} \quad , \quad (117)$$

$$\sigma_{xy}^* = \sigma_{xy1} \quad . \quad (118)$$

In module PLA4, the element stiffness matrices are calculated in the same manner as Equations 15 and 16 of Section 8.4.3 (or as in Section 8.4.9 for the QDMEM), with the

STRUCTURAL ELEMENT DESCRIPTIONS

exception that $[G_e]$ matrix is replaced by the $[G_p]$ matrix (Equation 97). In the calculation for $[G_p]$, E_1 (Equation 112) is used for E^* , and the newly calculated membrane stresses (Equations 103, 104, and 105) are used in place of σ_x^* , σ_y^* , and σ_{xy}^* .

8.4.15 Thermal Analysis Calculations for the Membrane Elements TRMEM and QDMEM

If the subroutines are to be used for thermal analysis, word 56 in labeled COMMON/SYSTEM/ is +1. The geometry of the element is processed, as with a structural analysis problem, to produce the parameters x_2 , x_3 , y_3 , A , and t . For thermal analysis, the 2×2 material conductivity matrix $[G_e^t]$ is obtained by calling subroutine HMAT. The transformation matrices (2×1) between temperatures at the connected grid points and thermal gradients are:

$$[c_1^t] = \begin{bmatrix} -\frac{1}{x_2} \\ \frac{1}{y_3} \left(\frac{x_3}{x_2} - 1 \right) \end{bmatrix},$$

$$[c_2^t] = \begin{bmatrix} \frac{1}{x_2} \\ -\frac{x_3}{x_2 y_3} \end{bmatrix},$$

$$[c_3^t] = \begin{bmatrix} 0 \\ \frac{1}{y_3} \end{bmatrix}.$$

The scalar heat conduction terms generated in subroutine KTRMEM and placed in the KGG matrix are:

$$k_{ij}^t = A t [c_i^t]^T [G_e^t] [c_j^t],$$

where i corresponds to the pivot grid point, and $j = 1, 2$, and 3 . The term is placed in column SIL_i and row number SIL_j of the matrix.

The heat transfer "mass" matrix is generated by subroutine MTRQD in module SMA2. The thermal capacity coefficient C_p is generated by subroutine HMAT. For each triangle, the scalar terms placed in the BGG matrix are:

$$B_{ii} = \frac{A t C_p}{3} \quad i = SIL_1, SIL_2, SIL_3.$$

TRMEM AND QDMEM ELEMENTS

The quadrilateral is subdivided into four overlapping triangles. The mass terms for each triangle are divided by two and added to the appropriate term in the BGG matrix.

The heat transfer "stress" calculations are executed by subroutines SDHTF1, SDHTFF, and SDHTF2. SDHTF1 rearranges the EST data for all elements to a common format and calls subroutine HMAT to produce the 2×2 conductivity matrix $[G_e^t]$ in element coordinates. Subroutine SDHTFF calculates the Phase 1 output terms with the following calculations:

For each triangle, the 2×3 matrix C_e is calculated where:

$$[C_e] = [C_a^t \ C_b^t \ C_c^t] = \frac{1}{2A} \begin{bmatrix} (\bar{y}_b - \bar{y}_c) & (\bar{y}_c - \bar{y}_a) & (\bar{y}_a - \bar{y}_b) \\ (\bar{x}_c - \bar{x}_b) & (\bar{x}_a - \bar{x}_c) & (\bar{x}_b - \bar{x}_a) \end{bmatrix},$$

where \bar{x}_i and \bar{y}_i are relative coordinates in the element coordinate system. For the triangle $a = 1, b = 2, c = 3$. For the quadrilateral, the element is broken into four triangles where $(a, b, c) = (1, 2, 3), (2, 3, 4), (3, 4, 1),$ or $(4, 1, 2)$. The matrices are superimposed in the quadrilateral to produce the average 2×4 output matrix $[C_e]$ where

$$\begin{aligned} [C_e] &= \frac{1}{4} [C_a^t \ C_b^t \ C_c^t \ 0] \text{ triangle No. 1} \\ &+ \frac{1}{4} [0 \ C_a^t \ C_b^t \ C_c^t] \text{ triangle No. 2} \\ &+ \frac{1}{4} [C_c^t \ 0 \ C_a^t \ C_b^t] \text{ triangle No. 3} \\ &+ \frac{1}{4} [C_b^t \ C_c^t \ 0 \ C_a^t] \text{ triangle No. 4} \end{aligned}$$

The output of Phase 1 consists of the matrix $[C_e]$ and the material matrix $[G_e^t]$. The Phase 2 routine, SDHTF2, calculates the gradients, $\begin{Bmatrix} \Delta T_x \\ \Delta T_y \end{Bmatrix}$, and the flux vector, $\{q\}$, where

$$\begin{Bmatrix} \Delta T_x \\ \Delta T_y \end{Bmatrix} = [C_e] \begin{Bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{Bmatrix}$$

$$\{q\} = -[G_e^t] \begin{Bmatrix} \Delta T_x \\ \Delta T_y \end{Bmatrix}$$

STRUCTURAL ELEMENT DESCRIPTIONS

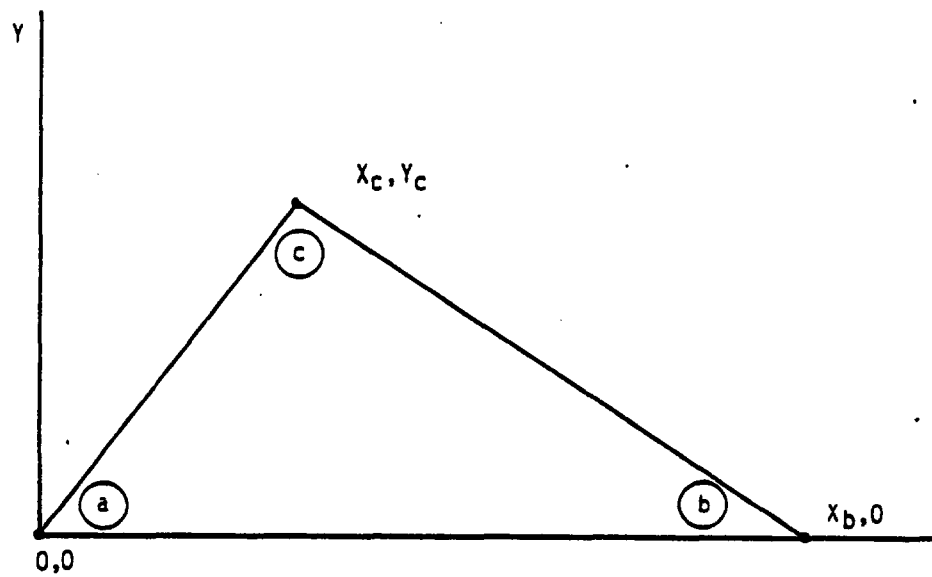


Figure 4. Triangular membrane element.

8.5 THE TRBSC, TRPLT AND QDPLT ELEMENTS

8.5.1 Input Data for the TRBSC and TRPLT Elements.

1. The ECPT/EST entries for the TRBSC and TRPLT are:

<u>Symbol</u>	<u>Description</u>
SIL_1, SIL_2, SIL_3	Scalar indices for the connected grid points
$N_i, X_i, Y_i, Z_i \}$ $i = 1, 2, 3$	Reference numbers for local coordinate system and locations in basic coordinates of the three connected grid points
I	Bending moment of inertia per unit width
t	Effective thickness for transverse shear
Mat Id. _b	Material property identification number for bending
Mat. Id. _s	Material property identification number for shear
θ	Material orientation angle
μ	Nonstructural mass per area
Z_1, Z_2	Fiber distances for stress calculations
t_μ	Temperature of element for material properties

2. ECPT entries for the QDPLT.

The entries are the same as those for the TRPLT except that four points are used.

3. Coordinate system data

Using N_i, X_i, Y_i, Z_i and the CSTM data block the 3 by 3 global-to-basic coordinate transformation matrices $[T_i]$ are produced for each point i via subroutines TRANSD or TRANSS.

4. Material data

Using the material property identification numbers, the orientation angle, the element temperature and the MPT and DIT data blocks, the following data are calculated:

STRUCTURAL ELEMENT DESCRIPTIONS

<u>Symbol</u>	<u>Description</u>
for Mat Id _b $\begin{cases} [G_b] \\ g_e \end{cases}$	3x3 elastic property matrix Structural damping coefficient
for Mat Id _s G_s	Shear coefficient

For the TRPLT and the QDPLT, the orientation of the material relative to each sub-triangle must be calculated from the geometry and the orientation given for the whole element. The details of this calculation are given in the next section.

8.5.2 General Calculation for the TRBSC Element

1. The coordinate system is defined by the three connected points a, b and c. $\{i\}$, $\{j\}$ and $\{k\}$ are the unit vectors along the x, y and z axis in basic coordinates. X_i , Y_i and Z_i are the location coordinates of the three points, $i=a, b, c$. (The element coordinate system for the basic bending triangle is shown in Figure 2 of Section 5.8 of the Theoretical Manual).

$$\{V_{ab}\} = \begin{Bmatrix} X_b - X_a \\ Y_b - Y_a \\ Z_b - Z_a \end{Bmatrix}, \quad (1)$$

$$\{V_{ac}\} = \begin{Bmatrix} X_c - X_a \\ Y_c - Y_a \\ Z_c - Z_a \end{Bmatrix}, \quad (2)$$

The x axis is defined by the unit vector:

$$\{i\} = \frac{\{V_{ab}\}}{|\{V_{ab}\}|}. \quad (3)$$

Calculate:

$$\{k\} = \frac{\{i\} \times \{V_{ac}\}}{|\{i\} \times \{V_{ac}\}|}. \quad (4)$$

The y axis is defined by the unit vector,

$$\{j\} = \{k\} \times \{i\} . \quad (5)$$

2. The locations of the points in element coordinates are:

$$x_a = y_a = y_b = 0 , \quad (6)$$

$$x_b = |\{V_{ab}\}| , \quad (7)$$

$$x_c = \{V_{ac}\}^T \{i\} , \quad (8)$$

$$y_c = \{V_{ac}\}^T \{j\} = |\{i\} \times \{V_{ac}\}| . \quad (9)$$

3. The 3 x 6 transformation matrix from the six displacements in element coordinates to the three degrees of freedom used in the plate is:

$$[E]^T = \begin{bmatrix} k_1 & k_2 & k_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & i_1 & i_2 & i_3 \\ 0 & 0 & 0 & j_1 & j_2 & j_3 \end{bmatrix} . \quad (10)$$

4. The coefficients used for the plate are:

$$[D] = I[G_b] , \quad (11)$$

$$[G_2] = t_s \begin{bmatrix} G_s & 0 \\ 0 & G_s \end{bmatrix} . \quad (12)$$

STRUCTURAL ELEMENT DESCRIPTIONS

where $[G_b]$ is the 3×3 material matrix for bending and G_s is the coefficient for transverse shear. The area of the triangle, A , the locations of the c.g., \bar{x} and \bar{y} , and the radii of gyration about the origin, ρ_x^2 , ρ_y^2 and ρ_{xy}^2 are given by:

$$A = \frac{1}{2} x_b y_c, \quad (13)$$

$$\bar{x} = \frac{1}{3} (x_c + x_b), \quad (14)$$

$$\bar{y} = \frac{1}{3} y_c, \quad (15)$$

$$\rho_x^2 = \frac{1}{6} (x_b^2 + x_b x_c + x_c^2), \quad (16)$$

$$\rho_y^2 = \frac{1}{6} y_c^2, \quad (17)$$

$$\rho_{xy}^2 = \frac{y_c}{12} (x_b + 2x_c). \quad (18)$$

5. The stiffness matrix in generalized coordinates $\{q\}$ is:

$$[K^X] = 4A \begin{bmatrix} D_{11} & D_{13} & D_{12} & 3\bar{x}D_{11} & \bar{x}D_{12} & 3\bar{y}D_{12} \\ & D_{33} & D_{23} & 3\bar{x}D_{13} & \bar{x}D_{23} & 3\bar{y}D_{23} \\ & & D_{22} & 3\bar{x}D_{12} & \bar{x}D_{22} & 3\bar{y}D_{22} \\ & & & 9\rho_x^2 D_{11} & 3\rho_x^2 D_{12} & 9\rho_{xy}^2 D_{12} \\ & & & & \rho_x^2 D_{22} + 4\rho_{xy}^2 D_{23} + 4\rho_y^2 D_{33} & 3\rho_{xy}^2 D_{22} + 6\rho_y^2 D_{23} \\ & & & & & 9\rho_y^2 D_{22} \end{bmatrix} \quad (19)$$

SYMMETRICAL

6. The transformation from generalized coordinates to grid point displacements (relative to point "a" of the triangle) with no transverse shear is:

$$[\bar{H}] = \begin{bmatrix} x_b^2 & 0 & 0 & x_b^3 & 0 & 0 \\ 0 & x_b & 0 & 0 & 0 & 0 \\ -2x_b & 0 & 0 & -3x_b^2 & 0 & 0 \\ x_c^2 & x_c y_c & y_c^2 & x_c^3 & x_c y_c^2 & y_c^3 \\ 0 & x_c & 2y_c & 0 & 2x_c y_c & 3y_c^2 \\ -2x_c & -y_c & 0 & -3x_c^2 & -y_c^2 & 0 \end{bmatrix} \quad (20)$$

where $\{q\} = [\bar{H}]\{u\}$ (no transverse shear)

7. If transverse shear effects are to be calculated ($G_s t_s \neq 0$), the following steps are followed:

a. Define

$$[J] = [G_2]^{-1}. \quad (21)$$

b. Calculate the transformation matrix of the shear coordinates:

$$[H_{Yq}] = - \begin{bmatrix} 0 & 0 & 0 & 6(J_{11}D_{11}+J_{12}D_{13}) & J_{11}(2D_{12}+4D_{33})+6J_{12}D_{23} & 6(J_{11}D_{23}+J_{12}D_{22}) \\ 0 & 0 & 0 & 6(J_{12}D_{11}+J_{22}D_{13}) & J_{12}(2D_{12}+4D_{33})+6J_{22}D_{23} & 6(J_{12}D_{23}+J_{22}D_{22}) \end{bmatrix}, \quad (22)$$

STRUCTURAL ELEMENT DESCRIPTIONS

where

$$\begin{Bmatrix} \gamma_x \\ \gamma_y \end{Bmatrix} = [H_{\gamma q}] \{q\} . \quad (23)$$

c. The stiffness matrix of the shear terms is added to the bending stiffness matrix.

$$[K^q] = [K^X] + A t_s [H_{\gamma q}]^T [G_2] [H_{\gamma q}] . \quad (24)$$

d. The effects of shear deflection on the transformation from general to displacement coordinates is calculated:

$$[H] = [\bar{H}] - \begin{bmatrix} 0 & 0 & 0 & x_b H_{\gamma q 14} & x_b H_{\gamma q 15} & x_b H_{\gamma q 16} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_c H_{\gamma q 14} & x_c H_{\gamma q 15} & x_c H_{\gamma q 16} \\ & & & +y_c H_{\gamma q 24} & +y_c H_{\gamma q 25} & +y_c H_{\gamma q 26} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} . \quad (25)$$

If no shear exists, $[H] = [\bar{H}]$

8. The matrix $[H]$ is inverted:

$$[H_{qu}] = [H]^{-1} . \quad (26)$$

9. The rigid body effects are given by the matrices $[B_b]$ and $[B_c]$ defined as follows:

$$[B_b] = \begin{bmatrix} 1 & 0 & -x_b \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad (27)$$

$$[B_c] = \begin{bmatrix} 1 & y_c & -x_c \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (28)$$

10. The 3x3 stiffness matrix partitions in element coordinates are calculated as follows:

$$[K] = [H^{-1}]^T [K^q] [H^{-1}], \quad (29)$$

$$[K] \Rightarrow \begin{bmatrix} k_{bb} & k_{bc} \\ k_{cb} & k_{cc} \end{bmatrix}, \quad (30)$$

$$[k_{ca}] = -[k_{cb}] [B_b] - [k_{cc}] [B_c], \quad (31)$$

$$[k_{ba}] = -[k_{bb}] [B_b] - [k_{bc}] [B_c], \quad (32)$$

$$[k_{aa}] = -[B_b]^T [k_{ba}] - [B_c]^T [k_{ca}], \quad (33)$$

$$[k_{ac}] = [k_{ca}]^T, \quad (34)$$

$$[k_{ab}] = [k_{ba}]^T. \quad (35)$$

8.5.3 Stiffness Matrix Calculations for the TRBSC Element (Subroutine KTRBSC of Module SMA1).

1. If the basic triangle is used by itself as a TRBSC element, the stiffness matrices are:

$$[K_{ij}] = \begin{bmatrix} T_i^T & 0 \\ 0 & T_i^T \end{bmatrix} [E] [k_{ij}] [E]^T \begin{bmatrix} T_j & 0 \\ 0 & T_j \end{bmatrix}, \quad (36)$$

where

$$i = a, b \text{ and } c$$

$$j = a, b \text{ and } c$$

2. The structural damping matrices are calculated using g_e , the structural damping coefficient, for the referenced bending material. The 6 by 6 damping matrix partitions are:

$$[K_{ij}^4] = g_e [K_{ij}] \quad (37)$$

8.5.4 Stress Calculations for the TRBSC Element.

The stress calculations involve two phases. The first phase is used to calculate the matrix relations between element forces and grid point displacements.

1. The relation between element forces and generalized coordinates is:

$$[k_s] = \begin{bmatrix} 2D_{11} & 2D_{13} & 2D_{12} & 6\bar{x}D_{11} & 2\bar{x}D_{12}^+ & 6\bar{y}D_{12} \\ 2D_{12} & 2D_{23} & 2D_{22} & 6\bar{x}D_{12} & 2\bar{x}D_{22}^+ & 6\bar{y}D_{22} \\ 2D_{13} & 2D_{33} & 2D_{23} & 6\bar{x}D_{13} & 2\bar{x}D_{23}^+ & 6\bar{y}D_{23} \\ 0 & 0 & 0 & -6D_{11} & -2D_{12}^- & -6D_{23} \\ 0 & 0 & 0 & -6D_{13} & 4D_{33} & -6D_{22} \end{bmatrix} \quad (38)$$

where

$$\begin{Bmatrix} M_x \\ M_y \\ M_{xy} \\ V_x \\ V_y \end{Bmatrix} = [K_s] \{q\} \quad (39)$$

Note: When the basic triangle routine is used for stress recovery in the TRPLT or QDPLT, the values \bar{x} and \bar{y} in the above matrix are replaced by x_c and y_c or x_q and y_q .

2. The matrix $[H]$ is calculated and inverted. The $[B]$ matrix is calculated ($[B]$ is a 6×3 matrix, the $[B_b]$ matrix (Equation 27) comprising the first three rows and the $[B_c]$ matrix, Equation 28, comprising the last three rows). The $[E]$ matrix and the global-to-basic transformation $[T_a]$, $[T_b]$, $[T_c]$ are generated. $[H]^{-1}$ is partitioned.

$$[H]^{-1} = [H_{1b} \mid H_{1c}]. \quad (40)$$

The element force - global displacement matrices are:

$$[S_a] = -[k_s][H]^{-1}[B][E]^T \begin{bmatrix} T_a & 0 \\ 0 & T_a \end{bmatrix}, \quad (41)$$

$$[S_b] = [k_s][H_{1b}][E]^T \begin{bmatrix} T_b & 0 \\ 0 & T_b \end{bmatrix}, \quad (42)$$

$$[S_c] = [k_s][H_{1c}][E]^T \begin{bmatrix} T_c & 0 \\ 0 & T_c \end{bmatrix}. \quad (43)$$

$$\{S_e\} = [D]\{\alpha\} \quad (43a)$$

where α is the vector of thermal expansion coefficients for the bending material.

3. The second stage of stress calculations involves the use of the displacement vectors $\{u_a\}$, $\{u_b\}$ and $\{u_c\}$. The element forces are:

$$\begin{Bmatrix} M_x \\ M_y \\ M_{xy} \\ V_x \\ V_y \end{Bmatrix} = \sum_{i=a,b,c} [S_i]\{u_i\} - \{M_e\}, \quad (44)$$

STRUCTURAL ELEMENT DESCRIPTIONS

where $\{M_e\}$ is the vector of thermal moments given on a TEMPP2 data field or if the gradient is given:

$$\{M_e\} = -T' \{S_t\}$$

With no thermal loads the stresses are:

$$\begin{Bmatrix} \sigma_{xi} \\ \sigma_{yi} \\ \sigma_{xyi} \end{Bmatrix} = -\frac{Z_i}{I} \begin{Bmatrix} M_x \\ M_y \\ M_{xy} \end{Bmatrix}, \quad i = 1, 2 \quad (45)$$

With direct thermal bending moments, $\{M_e\}$, given, the stresses are:

$$\begin{Bmatrix} \sigma_{xi} \\ \sigma_{yi} \\ \sigma_{xyi} \end{Bmatrix} = -\frac{Z_i}{I} \begin{Bmatrix} M_x \\ M_y \\ M_{xy} \end{Bmatrix} + \{M_e\} - \frac{1}{I} (T_i - \bar{T}) \{S_t\}, \quad i = 1, 2 \quad (45a)$$

With thermal gradient data, T' , the stresses are:

$$\begin{Bmatrix} \sigma_{xi} \\ \sigma_{yi} \\ \sigma_{xyi} \end{Bmatrix} = -\frac{Z_i}{I} \begin{Bmatrix} M_x \\ M_y \\ M_{xy} \end{Bmatrix} - \frac{1}{I} (T_i - Z_i T' - \bar{T}) \{S_t\}, \quad i = 1, 2 \quad (45b)$$

T_i is the given temperature at point i and \bar{T} is the average temperature of the element. If no T_i values are given, Equation (45) is used.

The principal stresses and their orientation are calculated in the same manner as those for the TRMEM element, Section 8.4.6.

Thermal loads are generated for this element in the SSG1 module. See Section 8.5.12 for the equations.

8.5.5 Stiffness Matrix Calculations for the TRPLT Element (Subroutine KTRPLT of Module SMA1).

The NASTRAN bending triangle (triangular plate element, TRPLT) is fabricated from three basic bending triangles. The geometry and notation are shown in Figure 5. The general approach is to calculate the stiffness matrices for all three subtriangles or basic triangles and use the constraint equation of equal slope at the midpoints of the connected edges to calculate a reduced stiffness matrix. Since only the partitions of the stiffness matrix related to one point (the pivot grid point) are used for each calculation, the extra partitions are not used. In the NASTRAN system, the basic bending triangle calculations are in subroutine form, and the variables necessary to call it are: x_b , x_c , y_c , the property and material coefficients, and the transformations for orienting the anisotropic materials. The following steps are used to calculate the overall stiffness matrix for the composite triangle.

1. The element coordinate system is defined by the location of the three grid points in basic coordinates, $\{x(1)\}$, $\{x(2)\}$ and $\{x(3)\}$:

$$\{V_2\} = \{x(2)\} - \{x(1)\}, \quad (46)$$

$$\{V_3\} = \{x(3)\} - \{x(1)\}, \quad (47)$$

$$x_2 = |\{V_2\}|, \quad (48)$$

$$\{i\} = \frac{\{V_2\}}{x_2}, \quad (49)$$

$$y_3 = |\{i\} \times \{V_3\}|, \quad (50)$$

$$\{k\} = \frac{\{i\} \times \{V_3\}}{y_3}, \quad (51)$$

STRUCTURAL ELEMENT DESCRIPTIONS

$$\{j\} = \{k\} \times \{i\}, \quad (52)$$

$$[E] = \begin{bmatrix} \{k\} & 0 & 0 \\ 0 & \{i\} & \{j\} \end{bmatrix} \cdot (6 \times 3) \quad (53)$$

The locations of the points in this coordinate system are:

$$\{R(1)\} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}, \quad (54)$$

$$\{R(2)\} = \begin{Bmatrix} x_2 \\ 0 \end{Bmatrix}, \quad (55)$$

$$x_3 = \{v_3\}^T \{i\}, \quad (56)$$

$$\{R(3)\} = \begin{Bmatrix} x_3 \\ y_3 \end{Bmatrix}, \quad (57)$$

$$\{R(4)\} = \frac{1}{3} \begin{Bmatrix} x_2 + x_3 \\ y_3 \end{Bmatrix}. \quad (58)$$

2. For use in transferring points to the subtriangles, the integer matrix [M] is formed:

	Point a	Point b	Point c	Subtriangle β	
[M] =	1	2	4	I	(59)
	2	3	4	II	
	3	1	4	III	

The Roman numerals I, II and III indicate the subtriangle numbers. Points 1, 2 and 3 are the corners of the whole triangle whose centroid is denoted by 4. Points a, b, and c are the corners of the subtriangles. Point c in the subtriangles is always the center point, 4. (see Figure 5).

Note: Steps 3 through 7 are performed for each subtriangle.

3. The location of the three points for each subtriangle, β , is defined by the vectors $\{r_i(a)\}$, $\{r_i(b)\}$, $\{r_i(c)\}$. In terms of the original vectors these are:

$$r_i(a) = R_i(M(\beta, 1)) \quad (60)$$

$$r_i(b) = R_i(M(\beta, 2)) \quad i = 1, 2, \quad (61)$$

$$r_i(c) = R_i(4) \quad (62)$$

where $M(\beta, i)$ is the (β, i) element of the $[M]$ matrix.

4. The variables necessary to calculate a basic bending triangle are x_b , x_c and y_c since the local coordinate system for each subtriangle is chosen such that the "a" point lies on the origin and the "b" point lies on the x axis.

For each triangle the following are calculated:

$$L = \sqrt{[r_1(b) - r_1(a)]^2 + [r_2(b) - r_2(a)]^2} \quad (\text{length of base}), \quad (63)$$

$$w_1 = \frac{1}{L} (r_1(b) - r_1(a)), \quad (64)$$

$$w_2 = \frac{1}{L} (r_2(b) - r_2(a)). \quad (65)$$

5. The matrix $[T]$ used for transforming the element coordinates to subtriangle coordinates is formed:

$$[T] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & w_1 & w_2 \\ 0 & -w_2 & w_1 \end{bmatrix}. \quad (66)$$

The material orientation angle, θ_m , is calculated for each subtriangle. The equations are:

$$\sin(\theta_m) = w_1 \sin(\theta) - w_2 \cos(\theta), \quad (67)$$

$$\cos(\theta_m) = w_1 \cos(\theta) + w_2 \sin(\theta). \quad (68)$$

STRUCTURAL ELEMENT DESCRIPTIONS

The displacements in the subtriangle system are equal to $[T]$ times the displacements in the original system (note $\det [T]$ is unit length, $x_b = R_1(2)$, $x_c = R_1(4)$ and $y_c = R_2(4)$ for subtriangle I).

6. The parameters x_b , x_c and y_c are then computed:

$$x_b = w_1[r_1(b) - r_1(a)] + w_2[r_2(b) - r_2(a)] , \quad (69)$$

$$x_c = w_1[r_1(c) - r_1(a)] + w_2[r_2(c) - r_2(a)] , \quad (70)$$

$$y_c = -w_2[r_1(c) - r_1(a)] + w_1[r_2(c) - r_2(a)] . \quad (71)$$

7. The stiffness matrices are formed as in the basic bending triangle (Equations 30 through 35) and give:

$$[k_{ia}], [k_{ib}], [k_{ic}], [k_{ca}], [k_{cb}], [k_{cc}] \quad i = \text{pivot grid point.}$$

They relate forces and displacements in the subtriangle coordinate system and are transformed to the overall element coordinate system (i.e., the same system as subtriangle I).

Since the stiffness matrices for each pivot grid point are calculated separately, not all of these partitions are used. For each pivot grid point, i , the following partitions are used:

$$[K_{i1}], [K_{i2}], [K_{i3}] ,$$

and

$$[K_{14}], [K_{24}], [K_{34}], [K_{44}] .$$

The integer mapping matrix $[M]$ is used to determine if and where to add the partitions. The steps used for pivot point i and triangle B are:

a) $[T]^T [k_{ac}] [T]$ is added to $[K_{m,4}]$, $m = M(\beta,1)$

$[T]^T [k_{bc}] [T]$ is added to $[K_{m,4}]$, $m = M(\beta,2)$

$[T]^T [k_{cc}] [T]$ is added to $[K_{4,4}]$

b) If $M(\beta,1) = i$:

$[T]^T [k_{aa}] [T]$ is added to $[K_{ii}]$

$[T]^T [k_{ab}] [T]$ is added to $[K_{il}]$, $l = M(\beta,2)$

or if $M(\beta,2) = i$

$[T]^T [k_{bb}] [T]$ is added to $[K_{ii}]$

$[T]^T [k_{ab}]^T [T]$ is added to $[K_{il}]$, $l = M(\beta,1)$

c) The above is repeated for each of the three subtriangles.

8. The number 4 point in the middle is a dummy point, and since the displacements at point 4 are functions of the other displacements, it will be removed from the problem by including the calculations for the point 4 displacements in the calculations for the corner displacements, as shown in steps 8a, 8b, 8c and 8d.

a. Calculate the following geometric constants:

$$l_1 = \left(x_c^2 + y_c^2 \right)^{1/2}, \quad (72)$$

$$l_2 = \left[(x_b - x_c)^2 + y_c^2 \right]^{1/2}, \quad (73)$$

$$s_1 = \frac{x_c}{l_1}, \quad (74)$$

$$s_2 = \frac{x_b - x_c}{l_2}, \quad (75)$$

$$c_1 = \frac{y_c}{l_1}, \quad (76)$$

$$c_2 = \frac{y_c}{x_2} \quad (77)$$

b. The formulas for the locations of the midpoints are:

$$x_1 = \frac{1}{2} x_c \quad (78)$$

$$y_1 = \frac{1}{2} y_c \quad (79)$$

$$x_2 = \frac{x_b + x_c}{2} \quad (80)$$

$$y_2 = \frac{1}{2} y_c \quad (81)$$

c. The $[H_\psi]$ matrix is calculated as follows:

$$[H_\psi] = \begin{bmatrix} -2x_1 c_1 (x_1 s_1 - y_1 c_1) & 2y_1 s_1 & -3x_1^2 c_1 & y_1 (2x_1 s_1 - y_1 c_1) & 3y_1^2 s_1 \\ 2x_2 c_2 (x_2 s_2 + y_2 c_2) & 2y_2 s_2 & 3x_2^2 c_2 & y_2 (2x_2 s_2 + y_2 c_2) & 3y_2^2 s_2 \end{bmatrix} \quad (82)$$

The slopes in terms of the deflections of points a, b and c are defined by the matrices:

$$[\bar{H}_{\psi b} | \bar{H}_{\psi c}] = [H_\psi][H]^{-1} \quad (83)$$

$$[\bar{H}_{\psi a}] = -[H_\psi][H]^{-1} [B] + \begin{bmatrix} 0 & s_1 & c_1 \\ 0 & s_2 & c_2 \end{bmatrix} \quad (84)$$

where $[H]^{-1}$ and $[B]$ are calculated while generating the stiffness matrices. $[H]^{-1}$ is the inverse of the 6x6 matrix in Equation 25, and $[B] = \begin{bmatrix} B_b \\ B_c \end{bmatrix}$ is a 6x3 matrix where $[B_b]$ and $[B_c]$ are calculated in Equations 27 and 28 respectively.

These are transformed to element coordinates by:

$$[H_{\psi \alpha}] = [\bar{H}_{\psi \alpha}] [T] \quad (85)$$

d. Each row of the matrix corresponds to the slope angle of the mid-point of the sides.

The first row defines the slope of the normal to the line connecting point "a" to the center point. The second row defines the slope of the normal to the line connecting point "b" to the center point. Using the $[H_{\psi\alpha}]$ matrices, four 3 by 3 matrices, $[G_M]$, are formed as follows:

The $[M]$ matrix is now used:

$$[M] = \begin{matrix} & \overbrace{\begin{matrix} a & b & c \end{matrix}}^{\alpha} \\ \begin{bmatrix} 1 & 2 & 4 \\ 2 & 3 & 4 \\ 3 & 1 & 4 \end{bmatrix} & \begin{matrix} I \\ II \\ III \end{matrix} & = \beta, \end{matrix} \quad (86)$$

For the $[H_{\psi\alpha}]^{\beta}$ matrix, ($\alpha = a, b$ and c ; $\beta = I, II$ or III), the number $M(\beta, \alpha)$ which identifies the matrix $[G_M]$ is found. The three terms in the first row of $[H_{\psi\alpha}]^{\beta}$ are added to the $M(\beta, 1)$ row of the $[G_M]$ matrix. The three numbers in the second row of $[H_{\psi\alpha}]^{\beta}$ are added to the $M(\beta, 2)$ row of the $[G_M]$ matrix. This procedure is repeated for the three $[H_{\psi\alpha}]^{\beta}$ matrices for each triangle β .

The stiffness matrix partitions of the whole plate are computed from:

$$\begin{aligned} [K_{ij}^e] &= [K_{ij}] - ([G_4]^{-1}[G_i])^T [K_{j4}]^T - [K_{i4}] [G_4]^{-1} [G_j] \\ &+ ([G_4]^{-1}[G_i])^T [K_{44}] [G_4]^{-1} [G_j] \quad \text{for } i = 1, 2, 3 \\ &\quad j = 1, 2, 3. \end{aligned} \quad (87)$$

where $[K_{ij}]$ are computed as in step 7.

9. Using the locations of the three grid points in basic coordinates, the 3x3 transformation matrices $[T_j]$, $j = 1, 2, 3$, are calculated. The 6x6 matrices $[C_j]$ are formed as:

$$[C_j] = \begin{bmatrix} [T_j] & | & 0 \\ \hline 0 & | & [T_j] \end{bmatrix} \quad j = 1, 2, 3. \quad (88)$$

10. The stiffness matrix partitions in global coordinates for pivot point i are computed from:

$$[K_{ij}^g] = [C_i]^T [E] [K_{ij}^e] [E]^T [C_j] \quad (6 \times 6), \quad (89)$$

where:

$[K_{ij}^e]$ was calculated in step 8

$[E]$ was calculated in step 1

$[C_i]$, $[C_j]$ were calculated in step 9

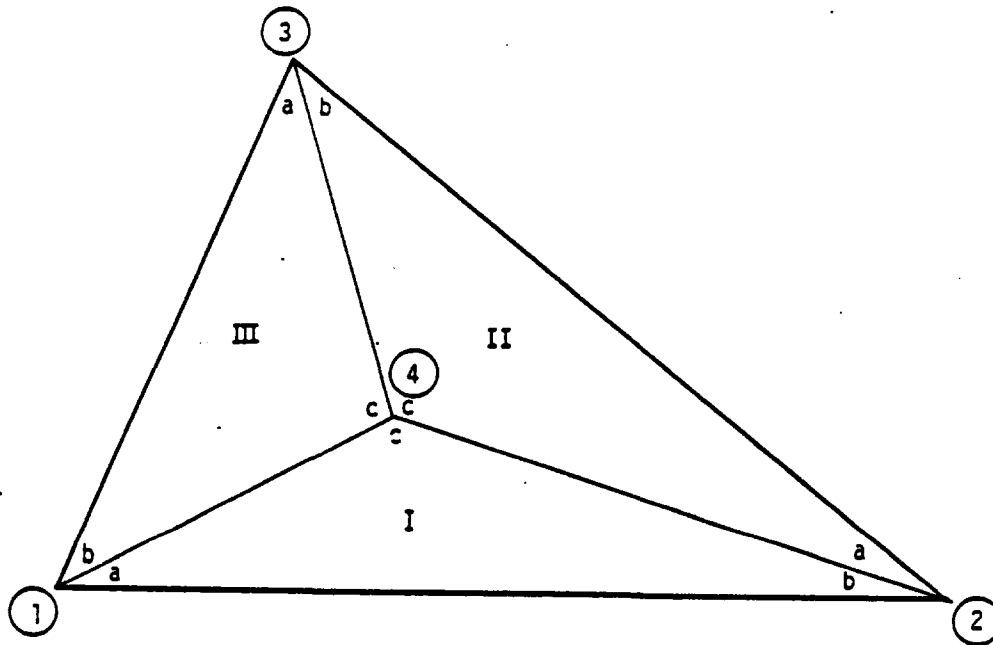


Figure 5. The triangular plate element, TRPLT.

- 1, 2, 3 = GIVEN GRID POINTS
- 4 = CENTROID OF TRIANGLE
- I, II, III = BASIC SUBTRIANGLES
- a, b, c = ORDERED VERTICES OF SUBTRIANGLES

8.5.6 Structural Damping Matrices for the TRPLT Element.

The structural damping matrices are:

$$[K_{ij}^4] = g_e [K_{ij}^g], \quad (90)$$

where g_e is the structural damping coefficient for the bending material referenced.

8.5.7 Stress and Element Force Calculations for the TRPLT Element (Subroutines STRPL1 and SBSPL2 of Module SDR2).

For stress recovery, the relationship between the center point and the corner points is used to describe the stress functions for each subtriangle. The stresses in each subtriangle at the center point are averaged to provide the final element stress and forces.

1. STRPL1 is used to calculate the phase 1 stress-displacement relations.

The following data are calculated using the same equations as those for the stiffness matrix generation routines.

$[C_i]$ - $i = 1, 2, 3$ - Global-to-basic transformations

$[E]$ - element to basic coordinate transformation

Values computed for
each subtriangle

$\left\{ \begin{array}{l} x_b, x_c, y_c - \text{subtriangle point locations} \\ \sin\theta, \cos\theta - \text{material orientation} \\ w_1, w_2 - \text{element-to-subtriangle coordinate transformation} \\ [\bar{H}_{\psi a}], [\bar{H}_{\psi b}], [\bar{H}_{\psi c}] - \text{normal slope angle relationship matrices} \end{array} \right.$

For each subtriangle, $\beta = I, II, III$, the following matrices are formed:

$$[T^\beta] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & w_1 & w_2 \\ 0 & -w_2 & w_1 \end{bmatrix}, \quad (91)$$

$$[V^\beta] = \begin{bmatrix} w_1^2 & w_2^2 & -2w_1w_2 & 0 & 0 \\ w_2^2 & w_1^2 & 2w_1w_2 & 0 & 0 \\ w_1w_2 & -w_1w_2 & w_1^2 - w_2^2 & 0 & 0 \\ 0 & 0 & 0 & w_1 & -w_2 \\ 0 & 0 & 0 & w_2 & w_1 \end{bmatrix} \quad (92)$$

For each subtriangle β , three 5×3 transformations are calculated, $[S_\alpha^\beta]$, $\alpha = a, b, c$.

These are transformed and added to the corresponding matrices for each point with the equation:

$$[S_M^*] = \frac{1}{3} \sum_{\beta} [V^\beta][S_\alpha^\beta][T^\beta], \quad (93)$$

where the α , which denotes points on the subtriangle, corresponds to the grid point M on the overall triangle.

$$[H_{\psi\alpha}] = [\bar{H}_{\psi\alpha}][T] \quad \alpha = a, b, c. \quad (94)$$

The $[H_{\psi\alpha}]$ matrices are added to the corresponding $[G_M]$ matrix with the appropriate row interchanges. When the data for all three subtriangles have been generated, the following operations are performed:

$$[S_M^e] = [S_M^*] - [S_4^*][G_4]^{-1}[G_i] \quad (95)$$

for $M = 1, 2, 3$;

$$[S_M] = [S_M^e][E]^T[C_i] \quad (96)$$

for $M = 1, 2, 3$.

2. Phase 2

a) The vector of forces is computed first.

$$\begin{Bmatrix} M_x \\ M_y \\ M_{xy} \\ V_x \\ V_z \end{Bmatrix} = \sum_{i=1}^3 [S_i] \{u_i\} - \{M_t\} \quad , \quad (97)$$

 where $\{M_e\}$ is the thermal moment vector. If a thermal gradient is given:

$$\{M_e\} = - \{S_t\} T' \quad .$$

b) With no given temperatures at the stress points, the stresses are then calculated from the equations

$$\sigma_{xi} = \frac{M_x Z_i}{I} \quad (98)$$

$$\sigma_{yi} = \frac{M_y Z_i}{I} \quad i = 1, 2 \quad . \quad (99)$$

$$\sigma_{xyi} = \frac{M_{xy} Z_i}{I} \quad (100)$$

 If temperature values T_i at the stress points are given the following equations are used.

$$\begin{Bmatrix} \sigma_{xi} \\ \sigma_{yi} \\ \sigma_{xyi} \end{Bmatrix} = - \frac{Z_i}{I} \begin{Bmatrix} M_x \\ M_y \\ M_{xy} \end{Bmatrix} + \{M_e\} - \frac{1}{I} (T_i - \bar{T}) \{S_t\} \quad , \quad i = 1 \text{ or } 2$$

or

STRUCTURAL ELEMENT DESCRIPTIONS

$$\begin{Bmatrix} \sigma_{xi} \\ \sigma_{yi} \\ \sigma_{xyi} \end{Bmatrix} = -\frac{z_i}{I} \begin{Bmatrix} M_x \\ M_y \\ M_{xy} \end{Bmatrix} - \frac{1}{I} (T_i - z_i T' - \bar{T}) \{S_t\}, \quad i = 1 \text{ or } 2$$

where T' is the gradient or $\{M_e\}$ is the thermal moment vector. \bar{T} is the average temperature for the element. The principal stresses and angles are calculated using the same formula as those for the membrane element (see Section 8.4.6, Equations 28, 29 and 30).

Thermal loads are generated for this element in the SSG1 module. See Section 8.5.11 for the algorithm.

8.5.8 Stiffness Matrix Calculations for the QDPLT Element (Subroutine QDPLT of Module SMA1)

The quadrilateral plate element uses two sets of overlapping triangles as shown in Figure 6. The logic is the same as that for the quadrilateral membrane except that the order of the points of the triangles is chosen to place the triangle coordinate systems along the diagonals.

1. The following equations are used to calculate the three unit vectors, $\{i\}$, $\{j\}$ and $\{k\}$, which define the element coordinate system.

$$\{V_i\} = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix}, \quad i = 1, 2, 3, 4. \quad (101)$$

The diagonals are:

$$\{d_1\} = \{V_3\} - \{V_1\}, \quad (102)$$

$$\{d_2\} = \{V_4\} - \{V_2\}. \quad (103)$$

The area is calculated from:

$$A = \frac{1}{2} |(\{d_1\} \times \{d_2\})|. \quad (104)$$

The normal to the plane is calculated from:

$$\{k\} = \frac{\{d_1\} \times \{d_2\}}{|\{d_1\} \times \{d_2\}|}, \quad (105)$$

$$\{a_1\} = \{V_2\} - \{V_1\}, \quad (106)$$

$$h = \frac{1}{2} \{a_1\}^T \{k\}, \quad (107)$$

The vectors lying in the new plane are computed from:

$$\{f\} = \frac{\{a_1\} - 2h\{k\}}{|\{a_1\} - h\{k\}|}, \quad (108)$$

$$\{f\} = \{k\} \times \{f\}. \quad (109)$$

The nonzero positions of the points in the plane are computed from:

STRUCTURAL ELEMENT DESCRIPTIONS

$$x_2 = \{a_i\}^T \{i\} , \quad (110)$$

$$x_3 = \{d_i\}^T \{i\} , \quad (111)$$

$$y_3 = \{d_i\}^T \{j\} , \quad (112)$$

$$x_4 = x_2 + (\{d_2\}^T \{i\}) , \quad (113)$$

$$y_4 = \{d_2\}^T \{j\} , \quad (114)$$

$$\{R(i)\} = \begin{Bmatrix} x_i \\ y_i \end{Bmatrix} . \quad (115)$$

2. Element interior angle tests.

The interior angles of the quadrilateral must be less than 180°. The following checks accomplish this task.

	<u>Test</u>	<u>Point with angle greater than 180°</u>
If	$y_4 < 0$	1
If	$y_3 < 0$	2
If	$x_4 > x_2 - (x_2 - x_3) \frac{y_4}{y_3}$	3
If	$x_3 < \frac{y_3}{y_4} x_4$	4

3. The relative data for each subtriangle must be calculated and passed to the matrix calculation subroutine. The integer mapping matrix [M] denotes which points, 1, 2, 3, and 4 of the quadrilateral, are used in the subtriangle. The row position indicates the subtriangle to which the point belongs, and the column position indicates the corresponding point in that subtriangle.

	<u>Point a</u>	<u>Point b</u>	<u>Point c</u>	<u>Triangle No.</u>
[M] =	2	4	1	I
	3	1	2	II
	4	2	3	III
	1	3	4	IV

4. For each triangle the stiffness matrix is calculated in its own coordinate system. This system has its origin at point a, point b lies on the x axis, and point c has a positive y component. $\{R(i)\}$ values are transferred to $\{r(\alpha)\}$ values ($\alpha = a, b, c$), and the following calculations are performed:

$$\{v(b)\} = \{r(b)\} - \{r(a)\}, \quad (116)$$

$$\{v(c)\} = \{r(c)\} - \{r(a)\}, \quad (117)$$

$$\begin{Bmatrix} w_1 \\ w_2 \end{Bmatrix} = \frac{\{v(b)\}}{|\{v(b)\}|}, \quad (118)$$

$$x_b = |\{v(b)\}|, \quad (119)$$

$$\begin{Bmatrix} x_c \\ y_c \end{Bmatrix} = \begin{bmatrix} w_1 & w_2 \\ -w_2 & w_1 \end{bmatrix} \{v(c)\}. \quad (120)$$

x_b , x_c and y_c are the point locations needed by the subroutine.

$$[T] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & w_1 & w_2 \\ 0 & -w_2 & w_1 \end{bmatrix}. \quad (121)$$

w_1 is the x component of the new x axis and w_2 is its y component, $[T]$ transforms the z displacement and the two angles from the quadrilateral system to the triangle system.

In order to calculate the material matrices in the basic triangle routine, the material orientation angle, θ_m , is :

$$\sin \theta_m = w_1 \sin \theta - w_2 \cos \theta , \quad (122)$$

$$\cos \theta_m = w_1 \cos \theta + w_2 \sin \theta . \quad (123)$$

w_1 and w_2 are the cosine and sine of the angle made between the base of the triangle and the material orientation axis.

5. The output of the basic bending triangle routine are the 3x3 matrices:

$$[k_{aa}], [k_{ab}], [k_{bb}], [k_{ac}], [k_{bc}], [k_{cc}] .$$

To transform these to the quadrilateral system the following equation is used:

$$[K_{ij}^e] = \frac{1}{2} [T]^T [k_{ij}] [T] . \quad (124)$$

These matrices are added to the current positions in the quadrilateral matrix partitions using the $[M]$ matrix in step 3.

6. For each pivot point i the following 3x3 partitions are formed:

$$[k_{ij}^e], \text{ for } i = 1, 2, 3, 4 .$$

7. Using the geometry data, the 3x3 global-to-basic transformations $[T_j]$ are formed for $j = 1, 2, 3, 4$. These are expanded to the 6x6 matrices $[C_j]$:

$$[C_j] = \begin{bmatrix} T_j & | & 0 \\ \hline 0 & | & T_j \end{bmatrix}. \quad (125)$$

8. The stiffness matrix partitions in global coordinates are found from:

$$[K_{ij}^g] = [C_i]^T [E] [k_{ij}^e] [E]^T [C_j], \quad (126)$$

where $[E]$ is defined in Equation 53.

8.5.9 Stress and Element Force Calculations for QDPLT Element (Subroutines SODPL1 and SBSPL2 of Module SDR2)

1. SQDPL1 calculates the phase 1 stress-displacement relations. A coordinate system matrix $[E]$, the subtriangle base vectors $\{w_1, w_2\}^T$ and the global-to-basic transformation matrices are calculated with the same equations as those used in the plate element stiffness matrix calculations. For each subtriangle, β , the following matrices are formed:

$$[T^\beta] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & w_1 & w_2 \\ 0 & -w_2 & w_1 \end{bmatrix}, \quad (127)$$

$$[V^\beta] = \begin{bmatrix} w_1^2 & w_2^2 & -2w_1w_2 & 0 & 0 \\ w_2^2 & w_1^2 & 2w_1w_2 & 0 & 0 \\ w_1w_2 & -w_1w_2 & (w_1^2 - w_2^2) & 0 & 0 \\ 0 & 0 & 0 & w_1 & -w_2 \\ 0 & 0 & 0 & w_2 & w_1 \end{bmatrix}. \quad (128)$$

STRUCTURAL ELEMENT DESCRIPTIONS

For each subtriangle the stresses must be calculated at the intersection point of the diagonals. In the quadrilateral system:

$$x_5 = \frac{x_2 x_3 y_4}{x_3 y_4 + (x_2 - x_4) y_3} , \quad (129)$$

$$y_5 = \frac{y_3}{x_3} x_5 . \quad (130)$$

For each subtriangle:

$$\{V_5\} = \{r(a)\} - \begin{Bmatrix} x_5 \\ y_5 \end{Bmatrix} ; \quad x_q = |\{V_5\}| ; \quad y_q = 0 ; \quad (131)$$

where x_q and y_q denote the location of the intersection of the diagonals of the quadrilateral in the subtriangle coordinate system. The stresses are calculated at this point.

$[T^\beta]$ transforms the translation and two rotations in the element system to the subtriangle system. $[V^\beta]$ transforms the three moments and two shears of the subtriangle to the element system. For each subtriangle β , three 5×3 transformations are calculated, $[S_\alpha^\beta]$, $\alpha = a, b, c$. These are transformed and added to the corresponding matrices for each point with the equation:

$$[S_M^e] = \frac{1}{4} \sum_{\beta} [V^\beta] [S_\alpha^\beta] [T^\beta] , \quad (132)$$

where the α , which denotes points on the subtriangle, corresponds to the grid point M on the quadrilateral.

2. Using the basic-to-element and global-to-basic transformations, the stress matrices $[S_M]$ in global coordinates are formed from:

$$[S_M] = [S_M^e] [E]^T \begin{bmatrix} [T_i] & 0 \\ 0 & [T_i] \end{bmatrix} . \quad (133)$$

The thermal load vector is:

$$\{S_t\} = [D]\{\alpha\}$$

3. The additional data for phase 2 calculations are

I, Z_1, Z_2 from the ECPT data.

4. Phase 2 (Subroutine SBSPL2)

The 5 by 6 $[S_M]$ matrices are used in the same manner as the TRPLT elements (see Equations 97 through 100).

8.5.10 Lumped Mass Matrix Generation for the TRBSC, TRPLT, and QDPLT Elements (Subroutine MASSTQ of Module SMA2).

The lumped mass matrices are calculated in the same manner as the triangular or quadrilateral membrane except that the material density is not used (see Equations 86 through 96 in Section 8.4.10). The only contribution to the mass matrix from these elements is the nonstructural mass, μ .

8.5.11 Coupled Mass Matrix Calculations for the TRBSC Element (Subroutine MTRBSC of Module SMA2).

The mass properties of the two-dimensional elements are defined by their geometry, the mass density given by the material, the thickness of the element and the nonstructural mass. The normal execution of NASTRAN will result in the calculation of the total mass of the element and distribute it as lumped masses at the connected grid points (subroutine MASSTQ of module SMA2). If the parameter C0UPBAR is set by the user, the elements with bending properties will have their mass distributed according to their elastic properties. This results in element mass matrices with directional properties and coupling terms between points in an element. Since the thickness of the TRBSC element is zero, a coupled mass matrix for this element does not exist. The MTRBSC subroutine is used exclusively by subroutines MTRPLT and MQDPLT.

1. The mass matrix $[\tilde{M}]$ in generalized coordinates is calculated in the following steps.

a. Integral values I_{ij} used in the mass matrices are calculated from the formulas:

$$A_{0j} = \left[\frac{x_b}{j+1} - \frac{x_b - x_c}{j+2} \right] y_c^{j+1}, \quad j = 0, 1, \dots, 6 \quad (134)$$

$$A_{ij} = \frac{(x_c)^{i+1} (y_c)^{j+1}}{(i+1)(i+j+2)} + \frac{ix_b}{(i+j+2)} A_{i-1,j}, \quad \begin{matrix} i = 1, 2, \dots, 6 \\ j = 0, 1, \dots, 6 \end{matrix} \quad (135)$$

$$B_{ij} = - \frac{(x_c)^{i+1}}{(y_c)^{i+1} (i+1)(i+j+2)} (y_c)^{i+j+2}, \quad \begin{matrix} i = 0, 1, \dots, 6 \\ j = 0, 1, \dots, 6 \end{matrix} \quad (136)$$

$$I_{ij} = m[A_{ij} + B_{ij}], \quad \begin{matrix} i = 0, 1, \dots, 6 \\ j = 0, 1, \dots, 6 \end{matrix} \quad (137)$$

where m is the nonstructural mass, and where x_b , x_c and y_c are computed in Equations 7, 8, and 9 respectively.

b. Partition $[\tilde{M}]$ into submatrices.

$$[\tilde{M}] \Rightarrow \begin{bmatrix} \bar{M}_{aa} & M_{ar} \\ M_{ar}^T & M_{rr} \end{bmatrix}, \quad (138)$$

where $[\bar{M}_{aa}]$ is a 3 by 3 matrix, $[M_{ar}]$ is a 3 by 6 matrix and $[M_{rr}]$ is a 6 by 6 matrix.

c. The mass matrix $[\bar{M}_{aa}]$ is given by:

$$[\bar{M}_{aa}] = \begin{bmatrix} I_{00} & I_{01} & -I_{10} \\ I_{01} & I_{02} & -I_{11} \\ -I_{10} & -I_{11} & I_{20} \end{bmatrix}. \quad (139)$$

d. The other matrices, $[M_{ar}]$ and $[M_{rr}]$, are less simple. The algebraic expressions for the elements of these matrices are given in Tables 1a and 1b below.

Table 1a. Elements of the 3 by 6 Matrix $[M_{ar}]$.

Notes: H_{Yqij} is contracted to H_{ij} , where $[H_{Yq}]$ is computed in Equation 22.

M_{arij} is contracted to M_{ik} , where M_{ik} represents an element of $[\tilde{M}]$ given in Equation 138.

$$M_{14} = I_{20}$$

$$M_{15} = I_{11}$$

$$M_{16} = I_{02}$$

$$M_{17} = I_{30} + H_{14}I_{10} + H_{24}I_{01}$$

STRUCTURAL ELEMENT DESCRIPTIONS

Table 1a (con'd). Elements of the 3 by 6 Matrix $[M_{ar}]$.

$$M_{18} = I_{12} + H_{15}I_{10} + H_{25}I_{01}$$

$$M_{19} = I_{03} + H_{16}I_{10} + H_{26}I_{01}$$

$$M_{24} = I_{21}$$

$$M_{25} = I_{12}$$

$$M_{26} = I_{03}$$

$$M_{27} = I_{31} + H_{14}I_{11} + H_{24}I_{02}$$

$$M_{28} = I_{13} + H_{15}I_{11} + H_{24}I_{02}$$

$$M_{29} = I_{04} + H_{16}I_{11} + H_{26}I_{02}$$

$$M_{34} = -I_{30}$$

$$M_{35} = -I_{21}$$

$$M_{36} = -I_{12}$$

$$M_{37} = -I_{40} - H_{14}I_{20} - H_{24}I_{11}$$

$$M_{38} = -I_{22} - H_{15}I_{20} - H_{25}I_{11}$$

$$M_{39} = -I_{13} - H_{16}I_{20} - H_{26}I_{11}$$

Table 1b. Elements of the 6 by 6 Matrix $[M_{rr}]$.

Notes: $H_{Yq_{ij}}$ is contracted to H_{ij} ; $M_{rr_{ij}}$ is contracted to M_{ik} , where M_{ik} represents an element of $[\tilde{M}]$ given in Equation 138; $M_{ij} = M_{ji}$.

$$M_{44} = I_{40}$$

$$M_{45} = I_{31}$$

$$M_{46} = I_{22}$$

$$M_{47} = I_{50} + H_{14}I_{30} + H_{24}I_{21}$$

$$M_{48} = I_{32} + H_{15}I_{30} + H_{25}I_{21}$$

$$M_{49} = I_{23} + H_{16}I_{30} + H_{26}I_{21}$$

$$M_{55} = I_{22}$$

$$M_{56} = I_{13}$$

$$M_{57} = I_{41} + H_{14}I_{21} + H_{24}I_{12}$$

$$M_{58} = I_{23} + H_{15}I_{21} + H_{25}I_{12}$$

$$M_{59} = I_{14} + H_{16}I_{21} + H_{26}I_{12}$$

$$M_{66} = I_{04}$$

$$M_{67} = I_{32} + H_{14}I_{12} + H_{24}I_{03}$$

$$M_{68} = I_{14} + H_{15}I_{12} + H_{25}I_{03}$$

$$M_{69} = I_{05} + H_{16}I_{12} + H_{26}I_{03}$$

$$M_{77} = I_{60} + 2H_{14}I_{40} + 2H_{24}I_{31} + (H_{14})^2I_{20} + 2H_{14}H_{24}I_{11} + (H_{24})^2I_{02}$$

STRUCTURAL ELEMENT DESCRIPTIONS

Table 1b (con'd). Elements of the 6 by 6 Matrix $[M_{rr}]$.

$$M_{78} = I_{42} + H_{15}I_{40} + H_{25}I_{31} + H_{14}I_{22} + H_{14}H_{15}I_{20} + H_{14}H_{25}I_{11} + H_{24}I_{13} \\ + H_{24}H_{15}I_{11} + H_{24}H_{25}I_{02}$$

$$M_{79} = I_{33} + H_{16}I_{40} + H_{26}I_{31} + H_{14}I_{13} + H_{14}H_{16}I_{20} + H_{14}H_{26}I_{11} + H_{24}I_{04} \\ + H_{24}H_{16}I_{11} + H_{24}H_{26}I_{02}$$

$$M_{88} = I_{24} + 2H_{15}I_{22} + 2H_{25}I_{13} + (H_{15})^2I_{20} + 2H_{15}H_{25}I_{11} + (H_{25})^2I_{02}$$

$$M_{89} = I_{15} + H_{16}I_{22} + H_{26}I_{13} + H_{15}I_{13} + H_{15}H_{16}I_{20} + H_{15}H_{26}I_{11} + H_{25}I_{04} \\ + H_{25}H_{16}I_{11} + H_{25}H_{26}I_{02}$$

$$M_{99} = I_{06} + 2H_{16}I_{13} + 2H_{26}I_{04} + (H_{16})^2I_{20} + 2H_{16}H_{26}I_{11} + (H_{26})^2I_{02}$$

2. The mass matrix $[M_{\text{mass}}]$ in element coordinates is calculated from the following equation:

$$[M_{\text{mass}}] = \begin{bmatrix} I & 0 \\ -H^{-1}B & H^{-1} \end{bmatrix}^T \begin{bmatrix} \bar{M}_{aa} & M_{ar} \\ M_{ar}^T & M_{rr} \end{bmatrix} \begin{bmatrix} I & 0 \\ -H^{-1}B & H^{-1} \end{bmatrix}, \quad (140)$$

where $[I]$ is a 3×3 identity matrix, $[H]^{-1}$ is calculated as in Equation 25, $[B] = \begin{bmatrix} B_b \\ B_c \end{bmatrix}$ is calculated as in Equations 27 and 28.

The calculations are broken down into the following steps where:

$$[M_{\text{mass}}] = \begin{bmatrix} M_{aa} & M_{ab} & M_{ac} \\ M_{ba} & M_{bb} & M_{bc} \\ M_{ca} & M_{cb} & M_{cc} \end{bmatrix}, \quad (141)$$

and $[M_{ij}]$, $i = a, b, c$ and $j = a, b, c$ are 3 by 3 matrices.

a) Compute:

$$[M] = [H^{-1}]^T [M_{rr}] [H^{-1}]. \quad (142)$$

b) Partition:

$$[M] \Rightarrow \begin{bmatrix} M_{bb} & M_{bc} \\ M_{cb} & M_{cc} \end{bmatrix}. \quad (143)$$

c) Compute:

$$[M_{ai}] = [M_{ar}] [H^{-1}]. \quad (144)$$

d) Partition:

$$[M_{ai}] \Rightarrow [\bar{M}_{ab} \mid \bar{M}_{ac}]. \quad (145)$$

e) Calculate:

$$[M_{ab}] = [\bar{M}_{ab}] - [B_b]^T [M_{bb}] - [B_c]^T [M_{cb}] \quad , \quad (146)$$

$$[M_{ac}] = [\bar{M}_{ac}] - [B_b]^T [M_{bc}] - [B_c]^T [M_{cc}] \quad , \quad (147)$$

$$[M_{aa}] = [\bar{M}_{aa}] - [B_b]^T [M_{ab}]^T - [B_c]^T [M_{ac}]^T \\ - [\bar{M}_{ab}] [B_b] - [\bar{M}_{ac}] [B_c] \quad , \quad (148)$$

$$[M_{ba}] = [M_{ab}]^T \quad , \quad (149)$$

$$[M_{ca}] = [M_{ac}]^T \quad . \quad (150)$$

8.5.12 Mass Matrix Calculations for the TRPLT Element (Subroutine MTRPLT of Module SMA2)

1. The general calculations for the mass matrix of the triangular plate element, TRPLT, are the same as those for the stiffness matrix calculations (See Equations 46 through 71).
2. For each subtriangle the output from the basic bending triangle subroutine are the nine 3 x 3 matrices given in Equation 141:

M_{aa}	M_{ab}	M_{ac}
M_{ba}	M_{bb}	M_{bc}
M_{ca}	M_{cb}	M_{cc}

They relate forces and accelerations in the subtriangle coordinate system and must be transformed to the overall element coordinate system (i.e., the same system as subtriangle I).

The matrix partitions in the subtriangles are added to the correct matrix partition for the whole triangle. For example, for subtriangle number II, $[M_{aa}]$ is transformed and added to $[\bar{M}_{22}]$, $[M_{ab}]$ is transformed and added to $[\bar{M}_{23}]$, $[M_{ac}]$ is transformed and added to $[\bar{M}_{24}]$, $[M_{ba}]$ is transformed and added to $[\bar{M}_{32}]$, etc.

Since the mass matrices for each pivot grid point are calculated separately, not all of these partitions are used. For each pivot grid point, i , the matrices which are used will be:

$$[\bar{M}_{i1}], [\bar{M}_{i2}], [\bar{M}_{i3}] \quad , \quad i = \text{point 1, 2 or 3 of composite triangle}$$

and

$$[\bar{M}_{14}], [\bar{M}_{24}], [\bar{M}_{34}], [\bar{M}_{44}] \quad .$$

3. The number 4 point in the middle is a dummy point and is removed from the problem in the same manner as in the computation of the stiffness matrix (see Equations 72 through 86).

The mass matrix partitions of the whole plate are:

$$\begin{aligned} [M_{ij}^e] &= [\bar{M}_{ij}]' - ([G_4]^{-1}[G_i])^T [\bar{M}_{j4}]^T - [\bar{M}_{i4}][G_4]^{-1}[G_j] \\ &+ ([G_4]^{-1}[G_i])^T [\bar{M}_{44}][G_4]^{-1}[G_j] \quad \begin{matrix} i = 1, 2, 3 \\ j = 1, 2, 3 \end{matrix} \end{aligned} \quad (151)$$

Notice that if i and j were interchanged the matrix would be transposed; this indicates that the whole mass matrix is symmetric.

4. Using the locations of the three grid points in basic coordinates, calculate the 3×3 transformation matrices $[T_j]$, $j = 1, 2, 3$. Form the 6×6 matrices:

$$[C_j] = \begin{bmatrix} T_j & 0 \\ 0 & T_j \end{bmatrix} \quad . \quad (152)$$

5. The 3×3 mass matrix partitions are expanded to 6×6 size and transformed to global coordinates using the logic:

$$M_3 = \frac{m}{6} x_2 y_3 \quad . \quad (153)$$

where m is the nonstructural mass, and x_2, y_3 are the x -coordinate and y -coordinate of points 2 and 3 respectively.

STRUCTURAL ELEMENT DESCRIPTIONS

$$[\bar{M}_{ij}^e] = \begin{bmatrix} M_3 & 0 & 0 & 0 & 0 & 0 \\ 0 & M_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & M_{11} & M_{12} & M_{13} & 0 \\ 0 & 0 & M_{21} & M_{22} & M_{23} & 0 \\ 0 & 0 & M_{31} & M_{32} & M_{33} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad (154)$$

for $i = j = 1, 2$ or 3 ; and

$$[\bar{M}_{ij}^e] = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & M_{11} & M_{12} & M_{13} & 0 \\ 0 & 0 & M_{21} & M_{22} & M_{23} & 0 \\ 0 & 0 & M_{31} & M_{32} & M_{33} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad (155)$$

for $i \neq j$.

The mass matrix partitions in global coordinates are:

$$[M_{ij}^g] = [C_i]^T [E] [\bar{M}_{ij}^e] [E]^T [C_j], \quad (156)$$

where

$$[C_i] = \begin{bmatrix} T_i & 0 \\ 0 & T_i \end{bmatrix} \quad (157)$$

is the global-to-basic transformation matrix,

and

$$[E] = \begin{bmatrix} \{i\} & \{j\} & \{k\} & 0 & 0 & 0 \\ 0 & 0 & 0 & \{i\} & \{j\} & \{k\} \end{bmatrix} \quad (158)$$

is the 6 x 6 element-to-basic transformation matrix.

8.5.13 Mass Matrix Calculations for the QDPLT Element (Subroutine MQDPLT of Module SMA2).

1. The general calculations for the mass matrix of the quadrilateral plate element, QDPLT, are the same as those for the stiffness matrix calculations (see Equations 101 through 123).
2. For each subtriangle, the output from the basic bending triangle subroutine are the nine 3 x 3 matrices:

$$\begin{bmatrix} M_{aa} & M_{ab} & M_{ac} \\ M_{ba} & M_{bb} & M_{bc} \\ M_{ca} & M_{cb} & M_{cc} \end{bmatrix}$$

These are transformed to the quadrilateral system by the following equation:

$$[\bar{M}_{ij}] = \frac{1}{2} [T]^T [M_{ij}] [T] \quad (159)$$

where $[T]$ is given in Equation 121.

These matrices are added to their corresponding positions in the quadrilateral matrix partitions $[M_{ij}^e]$, in the same manner as that for the stiffness matrix partitions of the quadrilateral (See step 3 of Section 8.5.8).

3. For each pivot point i , the following 3 x 3 partitions are formed:

$$[M_{ij}^e], \text{ for } j = 1, 2, 3, 4$$

4. The mass at each point in the plane of the element is due to the mass of the attached triangles. The masses of the triangles are:

$$m^{\beta} = \frac{m}{4} x_b^{\beta} y_c^{\beta} , \quad \beta = I, II, III \text{ and } IV , \quad (160)$$

where m is the nonstructural mass, and x_b^{β} and y_c^{β} are the x-coordinate and y-coordinate of points b and c respectively of subtriangle β .

The masses at the points are:

$$m_1 = \frac{1}{3} [m^I + m^{II} + m^{IV}] , \quad (161)$$

$$m_2 = \frac{1}{3} [m^I + m^{II} + m^{III}] , \quad (162)$$

$$m_3 = \frac{1}{3} [m^{II} + m^{III} + m^{IV}] , \quad (163)$$

$$m_4 = \frac{1}{3} [m^I + m^{III} + m^{IV}] . \quad (164)$$

5. The 3×3 mass matrix partitions are expanded to 6×6 matrices and the in-plane mass effects are added using the logic:

$$m_z^i = \frac{m_i h}{2} , \quad (165)$$

$$I_z^i = \frac{m_i h^2}{4} , \quad (166)$$

$$[\overline{M}_{ij}^e] =$$

$$\begin{bmatrix} m_i & 0 & 0 & 0 & -m_z^i & 0 \\ 0 & m_i & 0 & m_z^i & 0 & 0 \\ 0 & 0 & M_{11} & M_{12} & M_{13} & 0 \\ 0 & m_z^i & M_{21} & M_{22} + I_z^i & M_{23} & 0 \\ -m_z^i & 0 & M_{31} & M_{32} & M_{33} + I_z^i & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(167)

for $i = j = 1$ or 3 ; and

$$[\overline{M}_{ij}^e] =$$

$$\begin{bmatrix} m_i & 0 & 0 & 0 & m_z^i & 0 \\ 0 & m_i & 0 & -m_z^i & 0 & 0 \\ 0 & 0 & M_{11} & M_{12} & M_{13} & 0 \\ 0 & -m_z^i & M_{21} & M_{22} + I_z^i & M_{23} & 0 \\ m_z^i & 0 & M_{31} & M_{32} & M_{33} + I_z^i & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(168)

for $i = j = 2$ or 4 ; and

$$[\bar{M}_{ij}^e] = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & M_{11} & M_{12} & M_{13} & 0 \\ 0 & 0 & M_{21} & M_{22} & M_{23} & 0 \\ 0 & 0 & M_{31} & M_{32} & M_{33} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad (169)$$

for $i \neq j$.

6. Using the geometry data, the 3×3 global-to-basic transformations $[T_j]$ are formed for $j = 1, 2, 3, 4$. These are expanded to 6×6 matrices:

$$[C_j] = \begin{bmatrix} T_j & 0 \\ 0 & T_j \end{bmatrix}. \quad (170)$$

7. The 6×6 element-to-basic transformation matrix is:

$$[E] = \begin{bmatrix} \{i\} & \{j\} & \{k\} & 0 & 0 & 0 \\ 0 & 0 & 0 & \{i\} & \{j\} & \{k\} \end{bmatrix}. \quad (171)$$

8. The 6×6 matrices are transformed to global coordinates using the equation:

$$[M_{ij}^g] = [C_i]^T [E] [\bar{M}_{ij}^e] [E]^T [C_j] \quad (172)$$

8.5.14 Thermal Load Equations for the Bending Elements (Subroutines TRBSC, TRPLT, and QDPLT of Module SSG1).

1. The phase I SDR2 routines for all bending elements generate for connected points g_1, g_2, g_3 , (and g_4) the 5x6 matrices $[S_1], [S_2], [S_3]$ (and $[S_4]$). These matrices are generated in the SSG1 module with a minor change in the basic triangle routine.
2. The matrix $[K_s]$ is described on page 8.5-8. In the SDR2 routine the values x and y depend on which element is actually being used. For SSG1 module, calculate the matrix:

$$[K_s^t] = A[K_s(\bar{x}, \bar{y})] \quad , \quad (173)$$

the definition of $[K_s]$ is:

$$[K_s] = \begin{bmatrix} DH_{xq} \\ \hline G_2 H_{yq} \end{bmatrix} \quad , \quad (174)$$

where A is the area of the basic triangle and (\bar{x}, \bar{y}) is the center location of the basic triangle. The lower partition will not be used.

3. For each type of element the vector, $\{x\}$, is generated where:

$$\{x_e\} = - \begin{Bmatrix} \alpha_1 T' \\ \alpha_2 T' \\ \alpha_{12} T' \\ 0 \\ 0 \end{Bmatrix} \quad \text{or} \quad \{x_e\} = \begin{Bmatrix} [D]^{-1} \{M_e\} \\ 0 \\ 0 \end{Bmatrix} \quad , \quad (175)$$

where T' is given on a TEMPP1 data card or calculated from a TEMPP3 card, $\alpha_1, \alpha_2, \alpha_{12}$ are the material thermal coefficient vector components, $[D]$ is the 3 by 3 material matrix, and $\{M_e\}$ are the thermal moments given on a TEMPP2 card.

STRUCTURAL ELEMENT DESCRIPTIONS

4. The 6x1 thermal load vectors in basic coordinates are:

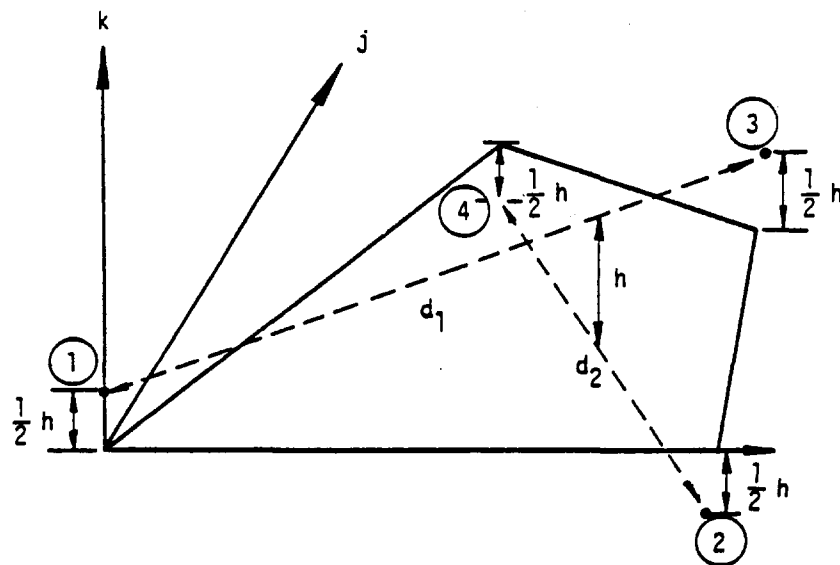
$$\{P_j\} = N[S_j]^T \{x_e\} \quad , \quad j = 1, 2, 3 \text{ (and 4)} \quad (176)$$

where N = 1: TRBSC

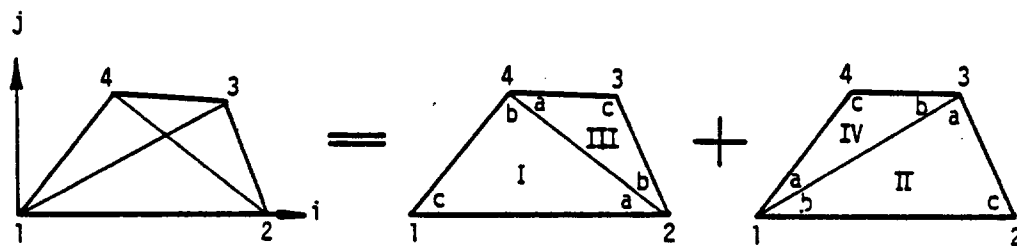
N = 3: TRPLT, TRIA1, TRIA2

N = 2: QDPLT, QUAD1, QUAD2

The $[S_j]^T$ matrix is calculated from the $[K_s]$ matrices of the various basic triangles forming the element. These $[K_s]$ matrices are transformed and combined to produce relations $[S_j]$ between the average element moments and the displacements of the connected grid points, j.



a) Definition of plane



b) Subtriangles

Figure 6. Geometry of the quadrilateral plate element, QDPLT.

8.6 THE TRIA1, TRIA2, QUAD1 and QUAD2 ELEMENTS

These elements have the properties of both membranes and bending plates. The TRIA1 and QUAD1 elements are triangles and quadrilaterals which may use separate thicknesses and materials for membrane, bending and transverse shear action. The TRIA2 and QUAD2 elements are triangles and quadrilaterals which use one thickness and one material to simulate a homogeneous plate with consistent membrane, bending and transverse shear properties.

If these elements use anisotropic materials (defined on a MAT2 bulk data card), the material is oriented with respect to the element coordinate system. The definition of the coordinate system is as follows: the vector from the first point to the second point defines the base or x axis. The z axis is normal to the average plane of the elements, and the third and fourth points have positive y values.

Mass matrices, thermal loads, and differential stiffness matrices for these elements use only the membrane properties.

8.6.1 Input Data for the TRIA1, TRIA2, QUAD1 and QUAD2 Elements

1. The ECPT/EST entries for the TRIA1 or QUAD1 elements are:

<u>Symbol</u>	<u>Description</u>
$SIL_i, \quad \left. \begin{array}{l} i = 1,2,3 \\ \text{or } i = 1,2,3,4 \end{array} \right\}$	Scalar indices of the connected grid points.
$\left. \begin{array}{l} N_i \\ X_i \\ Y_i \\ Z_i \end{array} \right\} \quad \left. \begin{array}{l} i = 1,2,3 \text{ or} \\ i = 1,2,3,4 \end{array} \right\}$	Referenced local coordinate system and location in basic coordinates of connected grid points.
θ	Material orientation angle.
MAT ID _m	Material identification number for membrane properties.
t_m	Membrane thickness.
MAT ID _b	Material identification number for bending properties.

STRUCTURAL ELEMENT DESCRIPTIONS

<u>Symbol</u>	<u>Description</u>
I	Bending inertia
MAT ID _s	Material identification number for transverse shear properties.
t _s	Transverse shear thickness
μ	Nonstructural mass per area
Z ₁ , Z ₂	Outer fiber distances for stress calculations.
t _μ	Temperature for material properties

2. ECPT Entries for the TRIA2 or QUAD2 Elements.

<u>Symbol</u>	<u>Description</u>
SIL _i i = 1,2,3 or i = 1,2,3,4	Scalar indices of connected grid points.
$\left. \begin{array}{l} N_i \\ X_i \\ Y_i \\ Z_i \end{array} \right\} \begin{array}{l} i = 1,2,3, \text{ or} \\ i = 1,2,3,4 \end{array}$	Referenced local coordinate system and location vector in basic coordinates of connected grid points.
θ	Material orientation angle.
MAT ID	Material identification number
t _m	Element thickness
μ	Nonstructural mass per area.
t _μ	Temperature for material properties

8.6.2 Stiffness Matrix Calculations (Subroutine KTRI0D of Module SMA1).

The TRIA1 or QUAD1 element ECPT data are rearranged to correspond to the (TRMEM or QDMEM) membrane ECPT form, and the routine of the TRMEM or QDMEM element is used. The ECPT data are then rearranged to correspond to the ECPT data of a TRPLT or QDPLT element and the respective plate routine is used. Each routine is entirely independent.

The TRIA2 and QUAD2 elements are treated in the same manner except that the arrangement of the ECPT data is different. The type "2" element uses the single material for all three material properties of the type "1" element. The membrane and transverse shear thickness equal the single thickness of the type "2" element. The bending inertia, I_b , for the plate property is:

$$I_b = \frac{t^3}{12} \quad (1)$$

8.6.3 Lumped Mass Matrix Generation (Subroutine MASSTQ of Module SMA2)

The bending properties are disregarded for the lumped mass matrix calculations, and the element mass matrices are computed exactly as the ones for the TRMEM and QDMEM elements.

8.6.4 Thermal Load Calculations (Subroutine of Module SSG1)

The TRPLT and QDPLT element routines are used to generate loads due to thermal gradients or moments. The TRMEM and QDMEM routines are used to calculate in-plane loads due to uniform thermal expansion.

8.6.5 Element Stress and Force Calculations (Subroutines STRQD1 and STRQD2 of Module SDR2)

As with the stiffness matrix calculations, the data are rearranged and the stresses for both the membrane and plate deformations are calculated. The element forces are calculated for the plate only.

1. For the TRIA2 and QUAD2 elements the outer fiber distances Z_1 and Z_2 are:

$$Z_1 = \frac{t}{2} \quad (2)$$

$$Z_2 = -\frac{t}{2} \quad (3)$$

The membrane and plate stresses are added together as follows for Z_1 :

$$\begin{pmatrix} \sigma_{x1} \\ \sigma_{y1} \\ \sigma_{xy1} \end{pmatrix}_{\text{composite}} = \begin{pmatrix} \sigma_x \\ \sigma_y \\ \sigma_{xy} \end{pmatrix}_{\text{membrane}} + \begin{pmatrix} \sigma_{x1} \\ \sigma_{y1} \\ \sigma_{xy1} \end{pmatrix}_{\text{bending}} \quad (4)$$

STRUCTURAL ELEMENT DESCRIPTIONS

and for Z_2 ,

$$\begin{pmatrix} \sigma_{x2} \\ \sigma_{y2} \\ \sigma_{xy2} \end{pmatrix}_{\text{composite}} = \begin{pmatrix} \sigma_x \\ \sigma_y \\ \sigma_{xy} \end{pmatrix}_{\text{membrane}} + \begin{pmatrix} \sigma_{x2} \\ \sigma_{y2} \\ \sigma_{xy2} \end{pmatrix}_{\text{bending}} \quad (5)$$

2. The principal stresses and their orientation are calculated from the above results, as in Equations 28, 29 and 30 of Section 8.4.6.

8.6.6 Coupled Mass Matrix Calculations (Subroutine MTRIQQ of Module SMA2)

In the lumped mass case these elements are processed using the membrane mass calculation routines (subroutine MASSTQ of module SMA2). When coupled mass is requested, the plate mass calculations will be used instead. The ECPT data are rearranged to the appropriate TRPLT or QDPLT format, and the respective plate routine is used. The mass per area is now calculated using the material mass density ρ and the thickness t_m for the membrane definition of the element and added to the nonstructural mass:

$$m = \mu + \rho t_m, \quad (6)$$

and m is now used instead of μ for the plate calculations.

8.6.7 Piecewise Linear Analysis Calculations (Subroutines PSTRI1, PSTRI2, PSQAD1, and PSQAD2 of Module PLA3 and PKTRI1, PKTRI2, PKQAD1 and PKQAD2 of Module PLA4).

The additional ECPTNL and ESTNL entries are:

ϵ_0^* - The previously computed strain value once removed.

ϵ^* - The previously computed strain value.

E^* - The previously computed modulus of elasticity.

σ_x^*
 σ_y^*
 σ_{xy}^*

} The previously computed membrane stresses.

M_x^*
 M_y^*
 M_{xy}^*
 V_x^*
 V_y^*

} The previously computed element forces (ESTNL only).

All of the above values are initially zero with the exception of E^* , which is initially the original modulus of elasticity present on a MAT1 card.

In module PLA3, the incremental element stress matrix is calculated by first rearranging the ESTNL data to correspond to the ESTNL data for a TRMEM or QDMEM, and then the membrane stresses are calculated in the same manner as Equations 103 through 105 of Section 8.4.14. Then the ESTNL data are rearranged to correspond to the EST data for a TRPLT (or QDPLT) and the incremental bending forces for the TRPLT (or QDPLT) element are calculated in the same manner as in Equation 97 of section 8.5.7. However, if the bending material properties are the same as the membrane material properties, then the 3 by 3 bending material properties matrix ($[G_b]$ in Equation 11 of Section 8.5.2 is replaced by the matrix given in Equation 97 of Section 8.4.14. In addition the displacement vector u_i in Equation 44 of Section 8.5.4 (or

STRUCTURAL ELEMENT DESCRIPTIONS

Equation 97 of Section 8.5.7) are replaced by an incremental displacement vector $\{\Delta u_i\}$.

The results are incremental stresses and forces for the membrane and bending properties defined as follows

$$\sigma_{x1} = \sigma_x^* + \Delta\sigma_x \quad (7)$$

$$\sigma_{y1} = \sigma_y^* + \Delta\sigma_y \quad (8)$$

$$\sigma_{xy1} = \sigma_{xy}^* + \Delta\sigma_{xy} \quad (9)$$

$$M_{x1} = M_x^* + \Delta M_x \quad (10)$$

$$M_{y1} = M_y^* + \Delta M_y \quad (11)$$

$$M_{xy1} = M_{xy}^* + \Delta M_{xy} \quad (12)$$

$$V_{x1} = V_x^* + \Delta V_x \quad (13)$$

$$V_{y1} = V_y^* + \Delta V_y \quad (14)$$

The total bending stresses are calculated using the total bending forces given in Equations 10 through 14 in conjunction with Equations 98 through 100 of Section 8.5.7.

The membrane and bending stresses are added together as follows for Z_1 :

$$\begin{Bmatrix} \sigma_{x1} \\ \sigma_{y1} \\ \sigma_{xy1} \end{Bmatrix}_{\text{composite}} = \begin{Bmatrix} \sigma_{x1} \\ \sigma_{y1} \\ \sigma_{xy1} \end{Bmatrix}_{\text{membrane}} + \begin{Bmatrix} \sigma_{x1} \\ \sigma_{y1} \\ \sigma_{xy1} \end{Bmatrix}_{\text{bending}} ; \quad (15)$$

and for Z_2 :

$$\begin{Bmatrix} \sigma_{x2} \\ \sigma_{y2} \\ \sigma_{xy2} \end{Bmatrix}_{\text{composite}} = \begin{Bmatrix} \sigma_{x1} \\ \sigma_{y1} \\ \sigma_{xy1} \end{Bmatrix}_{\text{membrane}} + \begin{Bmatrix} \sigma_{x2} \\ \sigma_{y2} \\ \sigma_{xy2} \end{Bmatrix}_{\text{bending}} \quad (16)$$

The principal stresses and their orientation for output are calculated from the above results, as in Equations 28 through 30 of Section 8.4.6.

In module PLA4, the bending properties are disregarded, and the ECPTNL data are rearranged to correspond to the ECPT data for the TRMEM or QDMEM, and the stresses are calculated exactly as the ones for the TRMEM or QDMEM elements (see Section 8.4.14).

In modules PLA3 and PLA4, after the above stress calculations have been completed, the next elastic coefficients are calculated in the same manner as Equations 106 through 112 of Section 8.4.14.

The new ESTNL and ECPTNL entries are:

$$\epsilon_0^* = \epsilon^* \quad (17)$$

$$\epsilon^* = \epsilon_1 \quad (18)$$

$$E^* = E_1 \quad (19)$$

$$\sigma_x^* = \sigma_{x1} \quad (20)$$

$$\sigma_y^* = \sigma_{y1} \quad (21)$$

$$\sigma_{xy}^* = \sigma_{xy1} \quad (22)$$

$$M_x^* = M_{x1} \quad (23)$$

$$M_y^* = M_{y1} \quad (24)$$

$$M_{xy} = M_{xy1} \quad (25)$$

STRUCTURAL ELEMENT DESCRIPTIONS

$$V_{x1}^* = V_{x1} , \quad (26)$$

$$V_{y1}^* = V_{y1} , \quad (27)$$

In module PLA4, the ECPTNL data are rearranged, and the element stiffness matrices are calculated in the same manner as for the TRMEM or QDMEM elements in Section 8.4.14.

8.6.8 Differential Stiffness Matrix Calculations for the TRIA1 and TRIA2 Elements (Subroutine DTRIA of Module DSMG1)

This subroutine uses the displacement vectors and thermal load temperature from a static solution to produce a differential stiffness matrix. The DTRMEM subroutine is used to calculate in plane-stresses and in-plane differential stiffness. The triangle is subdivided into three subtriangles and the DTRBSC routine is used to calculate out-of-plane differential stiffness for each of the three subtriangles. The matrices are combined and the center point is removed in the same manner as the KTRPLT subroutine (section 8.5.5). The basic steps are as follows:

1. The TRIA2 data from the ECPT is converted to TRIA1 format while in both cases the data is moved to a protected location. The conversion is:

$$\begin{array}{cc} \text{TRIA1 DATA} & \text{TRIA2 DATA} \\ \text{MATID}_m = \text{MATID}_b = \text{MATID}_s = \text{MATID} & \end{array} \quad (28)$$

$$t_m = t_s = t_m \quad (29)$$

$$I = \frac{t_m^3}{12.0} \quad (30)$$

2. The ECPT data are rearranged to the TRMEM format (the same as the TRIA2 format).
3. The material property orientation angles are established and subroutine DTRMEM is called. This routine will insert the in-plane differential stiffness terms in the KDGG matrix and will return the stress values $\bar{\sigma}_x$, $\bar{\sigma}_y$, and $\bar{\tau}_{xy}$.
4. The element coordinate system and geometric coefficients are calculated as follows where the three location vectors are $\{x(1)\}$, $\{x(2)\}$, and $\{x(3)\}$.

$$\{V_2\} = \{x(2)\} - \{x(1)\} \quad (31)$$

$$\{V_3\} = \{x(3)\} - \{x(1)\} \quad (32)$$

THE TRIA1, TRIA2, QUAD1 AND QUAD2 ELEMENTS

$$x_2 = |\{V_2\}| \quad (33)$$

$$\{i\} = \frac{\{V_2\}}{x_2} \quad (34)$$

$$y_3 = |\{i\} \times \{V_3\}| \quad (35)$$

$$\{k\} = \frac{\{i\} \times \{V_3\}}{y_3} \quad (36)$$

$$\{j\} = \{k\} \times \{i\} \quad (37)$$

$$x_4 = \frac{x_2 + x_3}{3} \quad (38)$$

$$y_4 = \frac{y_3}{3} \quad (39)$$

The locations of the four points in this coordinate system are defined by the matrix [R] where each row defines one point

$$[R] = \begin{bmatrix} 0 & 0 \\ x_2 & 0 \\ x_3 & y_3 \\ x_4 & y_4 \end{bmatrix} \quad (40)$$

5. For use in transferring points in the element to points in the subtriangle, the integer mapping matrix [M] is used.

$$[M] = \begin{array}{c} \begin{array}{ccc} \text{Point a} & \text{Point b} & \text{Point c} \end{array} \\ \begin{bmatrix} 1 & 2 & 4 \\ 2 & 3 & 4 \\ 3 & 1 & 4 \end{bmatrix} \end{array} \quad (41)$$

Each row of the matrix corresponds to the three points connected to each subtriangle.

6. A major loop is now performed with one cycle for each of the three subtriangles. Corresponding to points a, b, and c of the M matrix, the location vectors $\{r_a\}$, $\{r_b\}$, and $\{r_c\}$ are extracted from the corresponding rows of the R matrix. For each triangle the

STRUCTURAL ELEMENT DESCRIPTIONS

following are calculated:

$$l = \sqrt{(r_{b1} - r_{a1})^2 - (r_{b2} - r_{a2})^2} \quad (42)$$

$$w_1 = \frac{1}{l} (r_{b1} - r_{a1}) \quad (43)$$

$$w_2 = \frac{1}{l} (r_{b2} - r_{a2}) \quad (44)$$

The transformation between subtriangle coordinates to element coordinates is:

$$[T] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & w_1 & w_2 \\ 0 & -w_2 & w_1 \end{bmatrix} \quad (45)$$

The material orientation data are

$$\left. \begin{aligned} \sin(\theta_m) &= w_1 \sin(\theta) - w_2 \cos(\theta) \\ \cos(\theta_m) &= w_1 \cos(\theta) + w_2 \sin(\theta) \end{aligned} \right\} \quad (46)$$

7. The locations of the three points of each subtriangle are transformed to geometric coefficients x_b, x_c, y_c where

$$x_b = w_1(r_{b1} - r_{a1}) + w_2(r_{b2} - r_{a2}) \quad (47)$$

$$x_c = w_1(r_{c1} - r_{a1}) + w_2(r_{c2} - r_{a2}) \quad (48)$$

$$y_c = -w_2(r_{c1} - r_{a1}) + w_1(r_{c2} - r_{a2}) \quad (49)$$

8. The stresses are transformed to the subtriangle system by the equations:

$$\sigma_x = w_1^2 \bar{\sigma}_x + w_2^2 \bar{\sigma}_y + 2w_1w_2\bar{\tau}_{xy} \quad (50)$$

$$\sigma_y = w_2^2 \bar{\sigma}_x + w_1^2 \bar{\sigma}_y - 2w_1w_2\bar{\tau}_{xy} \quad (51)$$

$$\tau_{xy} = -w_1w_2\bar{\sigma}_x + w_1w_2\bar{\sigma}_y + (w_1^2 - w_2^2) \bar{\tau}_{xy} \quad (52)$$

9. The differential stiffness matrix for each subtriangle is formed in subroutine DTRBSC (Section 8.6.10). The input to this routine consists of the ECPT property data; the location parameters x_b , x_c , and y_c ; and the in plane stresses σ_x , σ_y and τ_{xy} . The output from this routine consists of the following matrices:

$$[K_{ij}^d] \quad \begin{cases} i = \text{pivot point} \\ j = 1, 2, \text{ and/or } 3 \end{cases}$$

$$[K_{j4}^d] \quad j = 1, 2, \text{ and/or } 3$$

$$[H], [S]$$

10. The above matrices are combined and transformed in exactly the same manner as the triangular plate equations. The differential stiffness matrices replace the elastic stiffness matrices in the equations. See section 8.5.5, steps 7 through 10.

8.6.9 Differential Stiffness Matrix Calculation for the QUAD1 and QUAD2 Elements (Subroutine DQUAD of Module DSMG1)

The differential stiffness matrix for the QUAD1 and QUAD2 elements is constructed from the matrices produced by four subtriangles. The method used to subdivide the quadrilateral is shown in Figure 6. The stress is calculated for each triangle using the DTRMEM subroutine. The out-of-plane differential stiffness for each triangle is calculated using the DTRBSC subroutine. The element geometry and the manipulation of the matrices is done in the same manner as the elastic stiffness equations for the quadrilateral plates.

The steps followed by the subroutine are:

1. If a QUAD2 element is used, its property data is converted to the QUAD1 equivalent and the ECPT is expanded to the QUAD1 format. The property conversion is:

$$\text{MATID}_m = \text{MATID}_6 = \text{MATID}_5 = \text{MATID} \quad (53)$$

$$t_m = t_s = t_m \quad (54)$$

$$I = \frac{t_m^3}{12.0} \quad (55)$$

STRUCTURAL ELEMENT DESCRIPTIONS

With both cases the data is stored in a protected location.

2. The element coordinate system and the location of the grid points are calculated as follows:

$$\left. \begin{aligned} \{d_1\} &= \{V_3\} - \{V_1\} \\ \{d_2\} &= \{V_4\} - \{V_2\} \end{aligned} \right\} \quad (56)$$

where $\{V_1\}$, $\{V_2\}$, $\{V_3\}$, and $\{V_4\}$ are the location vectors of the connected grid points.

$$\{k\} = \frac{\{d_1\} \times \{d_2\}}{|\{d_1\} \times \{d_2\}|} \quad (57)$$

$$\{a_1\} = \{V_2\} - \{V_1\} \quad (58)$$

$$h = \{a_1\} \cdot \{k\} \quad (59)$$

$$\{i\} = \frac{\{a_1\} - h\{k\}}{|\{a_1\} - h\{k\}|} \quad (60)$$

$$\{j\} = \{k\} \times \{i\} \quad (61)$$

The locations of the points in the element plane are stored in the [R] matrix where each row corresponds to the x and y location of a point.

$$R_{11} = R_{12} = R_{22} = 0.0 \quad (62)$$

$$R_{21} = X_2 = \{a_1\} \cdot \{i\} \quad (63)$$

$$R_{31} = X_3 = \{d_1\} \cdot \{i\} \quad (64)$$

$$R_{32} = Y_3 = \{d_1\} \cdot \{j\} \quad (65)$$

$$R_{41} = X_4 = X_2 + \{d_2\} \cdot \{i\} \quad (66)$$

$$R_{42} = Y_4 = \{d_2\} \cdot \{j\} \quad (67)$$

3. At this stage the four triangles are processed in a loop. The matrix partition for the element is set to zero and the results for each triangle are added in. The following steps 4 - 8 describe this loop.

4. The three points for each triangle are selected using the mapping matrix M. The data corresponding to these points are put into the ECPT format for a TRMEM element.

5. The geometry of the triangle is calculated using the three rows of the R matrix corresponding to the three points of the triangle a, b, c. These rows correspond to vectors $\{V_a\}$, $\{V_b\}$, and $\{V_c\}$.

$$\{V\} = \{V_b\} - \{V_a\} \quad (68)$$

$$\{V_v\} = \{V_c\} - \{V_a\} \quad (69)$$

$$x_b = |\{V\}| \quad (70)$$

$$\begin{Bmatrix} w_1 \\ w_2 \end{Bmatrix} = \frac{\{V\}}{x_b} \quad (71)$$

$$x_c = w_1 V_{v1} + w_2 V_{v2} \quad (72)$$

$$y_c = -w_2 V_{v1} + w_1 V_{v2} \quad (73)$$

The transformation matrix between element and triangle coordinates is:

$$[T] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & w_1 & w_2 \\ 0 & -w_2 & w_1 \end{bmatrix} \quad (74)$$

6. The orientation angle θ_m for the subtriangle is computed in case the material is anisotropic.

$$\sin \theta_m = w_1 \sin \theta - w_2 \cos \theta \quad (75)$$

$$\cos \theta_m = w_1 \cos \theta + w_2 \sin \theta \quad (76)$$

7. The triangular membrane subroutine DTRMEM at this point will calculate and insert the in-plane differential stiffness terms and will produce the stress values σ_x , σ_y , and τ_{xy} in the triangle coordinates. The basic triangle subroutine will use the in-plane stresses and the basic locations x_b , x_c , and y_c to produce the out of plane differential stiffness terms. The output of the differential stiffness subroutine, DTRBSC, is the 3 by 3 matrix

STRUCTURAL ELEMENT DESCRIPTIONS

partitions $[K_{ia}]$, $[K_{ib}]$, and $[K_{ic}]$, where i is the pivot point. These are transformed to quadrilateral coordinates with the $[T]$ matrix.

$$[K_{ij}^e] = [T]^T [K_{ij}] [T] \quad (77)$$

8. The matrices for each triangle are added into the running sum for the quadrilateral and steps 4 - 8 are repeated.

9. The differential stiffness matrix partitions are transformed to global coordinates and inserted into the overall differential stiffness matrix in a manner identical to steps 7 and 8 of Section 8.5.8.

8.6.10 Differential Stiffness Matrix Calculations for the Basic Bending Triangle (Subroutine DTRBSC of Module DSMG1)

Unlike the case of elastic stiffness matrix generation, the basic triangle (TRBSC) may not be used by itself to produce differential stiffness matrix terms. This subroutine, however, is used for the calculation of differential stiffness for the TRIA1, TRIA2, QUAD1, and QUAD2 elements. Its purpose is analogous to the way the KTRBSC subroutine is used in the calculation of elastic stiffness matrices.

The necessary inputs to this subroutine are passed to it via the labeled common blocks DSIAET and DSIADP. The input data used are x_b , x_c , y_c (the basic geometry), σ_x , σ_y , τ_{xy} (the in-plane stresses), and the element property data.

The basic algorithm used by the routine is as follows:

1. The presence of transverse shear is tested and the subroutine selects the method of calculating the element coordinate-to-generalized coordinate transformation matrices $[H_b]$ and $[H_c]$.
2. If no transverse shear flexibility exists, the matrices $[H_b]$ and $[H_c]$ are calculated by the following equations:

$$\begin{aligned}
 r &= \frac{1}{x_b} \\
 s &= \frac{1}{y_c} \\
 t &= \frac{x_c}{y_c} \\
 u &= \frac{x_c}{x_b^2 y_c^2} = r^2 s t
 \end{aligned}
 \tag{78}$$

$$[H_b] = \begin{bmatrix} 3r^2 & 0 & r \\ 0 & r & 0 \\ -3r^2 t^2 & -rt & -rt^2 \\ -2r^3 & 0 & -r^2 \\ -6ru(x_b - x_c) & -rs & u(3x_c - 2x_b) \\ 2rtu(3x_b - 2x_c) & rst & 2tu(x_b - x_c) \end{bmatrix} \quad (6 \times 3)
 \tag{79}$$

$$[H_c] = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 3s^2 & -s & st \\ 0 & 0 & 0 \\ 0 & 0 & -s^2 \\ -2s^3 & s^2 & 0 \end{bmatrix} \quad (6 \times 3)
 \tag{80}$$

3. If both shear material and shear thickness exist, then the $[H_{yq}]$ and $[H]^{-1}$ matrices are generated as with the existing equations for the TRBSC element. See pages 8.5-5, 6. The 6x6 $[H]^{-1}$ matrix, is partitioned as follows:

$$[H]^{-1} \Rightarrow [H_b : H_c]
 \tag{81}$$

and is used instead of Equations (79) and (80).

STRUCTURAL ELEMENT DESCRIPTIONS

4. In order to form the differential stiffness matrix $[K^{dq}]$, referred to generalized coordinates, the following integral must be evaluated over the triangular area.

$$I_{nm} = h \int_A x^n y^m dA \quad (82)$$

The results are:

$$\begin{aligned} I_{00} &= hA = \frac{hx_b y_c}{2} & I_{10} &= \frac{hA}{3} (x_b + x_c) \\ I_{01} &= \frac{hAy_c}{3} & I_{11} &= \frac{hAy_c}{12} (x_b + 2x_c) \\ I_{02} &= \frac{hAy_c^2}{6} & I_{12} &= \frac{hAy_c^2}{30} (x_b + 3x_c) \\ I_{03} &= \frac{hAy_c^3}{10} & I_{13} &= \frac{hAy_c^3}{60} (x_b + 4x_c) \\ I_{04} &= \frac{hAy_c^4}{15} \\ I_{20} &= \frac{hA}{6} (x_b^2 + x_b x_c + x_c^2) & (83) \\ I_{21} &= \frac{hAy_c}{30} (x_b^2 + 2x_b x_c + 3x_c^2) \\ I_{22} &= \frac{hAy_c^2}{90} (x_b^2 + 3x_b x_c + 6x_c^2) \\ I_{30} &= \frac{hA}{10} [x_b^3 + x_b^2 x_c + x_b x_c^2 + x_c^3] \\ I_{31} &= \frac{hAy_c}{60} [x_b^3 + 2x_b^2 x_c + 3x_b x_c^2 + 4x_c^3] \\ I_{40} &= \frac{hA}{15} [x_b^4 + x_b^3 x_c + x_b^2 x_c^2 + x_b x_c^3 + x_c^4] \end{aligned}$$

THE TRIA1, TRIA2, QUAD1 AND QUAD2 ELEMENTS

5. The elements of the (8x8) differential stiffness matrix $[K^{dq}]$ are listed below. The matrix is symmetric so only the upper triangle terms are given. The superscript (^{dq}) is omitted for convenience.

$$K_{11} = \sigma_x I_{00}$$

$$K_{12} = \tau I_{00}$$

$$K_{13} = 2\sigma_x I_{10}$$

$$K_{14} = \sigma_x I_{01} + \tau I_{10}$$

$$K_{15} = 2\tau I_{01}$$

$$K_{16} = 3\sigma_x I_{20}$$

$$K_{17} = \sigma_x I_{02} + 2\tau I_{11}$$

$$K_{18} = 3\tau I_{02}$$

$$K_{22} = \sigma_y I_{00}$$

$$K_{23} = 2\tau I_{10}$$

(84)

$$K_{24} = \tau I_{01} + \sigma_y I_{10}$$

$$K_{25} = 2\sigma_y I_{01}$$

$$K_{26} = 3\tau I_{20}$$

$$K_{27} = \tau I_{02} + 2\sigma_y I_{11}$$

$$K_{28} = 3\sigma_y I_{02}$$

$$K_{33} = 4\sigma_x I_{20}$$

$$K_{34} = 2(\sigma_x I_{11} + \tau I_{20})$$

$$K_{35} = 4\tau I_{11}$$

$$K_{36} = 6\sigma_x I_{30}$$

STRUCTURAL ELEMENT DESCRIPTIONS

$$\begin{aligned}
 K_{37} &= 2(\sigma_x I_{12} + 2\tau I_{21}) \\
 K_{38} &= 6\tau I_{12} \\
 K_{44} &= \sigma_x I_{02} + 2\tau I_{11} + \sigma_y I_{20} \\
 K_{45} &= 2(\tau I_{02} + \sigma_y I_{11}) \\
 K_{46} &= 3(\sigma_x I_{21} + \tau I_{30}) \\
 K_{47} &= \sigma_x I_{03} + 3\tau I_{12} + 2\sigma_y I_{21} \\
 K_{48} &= 3(\tau I_{03} + \sigma_y I_{12}) \\
 K_{55} &= 4\sigma_y I_{02} \\
 K_{56} &= 6\tau I_{21} \\
 K_{57} &= 2(\tau I_{03} + 2\sigma_y I_{12}) \\
 K_{58} &= 6\sigma_y I_{03} \\
 K_{66} &= 9\sigma_x I_{40} \\
 K_{67} &= 3(\sigma_x I_{22} + 2\tau I_{31}) \\
 K_{68} &= 9\tau I_{22} \\
 K_{77} &= \sigma_x I_{04} + 4\tau I_{13} + 4\sigma_y I_{22} \\
 K_{78} &= 3(\tau I_{04} + 2\sigma_y I_{13}) \\
 K_{88} &= 9\sigma_y I_{04}
 \end{aligned} \tag{84}$$

6. In order to transform the matrix to the displacements of points at the corners of the triangle, the following matrices are generated.

$$[H_a] = [H_b][S_b] - [H_c][S_c] \quad (6 \times 3) \tag{85}$$

THE TRIA1, TRIA2, QUAD1 AND QUAD2 ELEMENTS

$$[C_a] = \begin{bmatrix} H_q H_a \\ \hline H_a \end{bmatrix} + \begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ \hline 0 \end{bmatrix} \quad (8 \times 3) \quad (86)$$

$$[C_b] = \begin{bmatrix} H_q H_b \\ \hline H_b \end{bmatrix} \quad (8 \times 3) \quad (87)$$

$$[C_c] = \begin{bmatrix} H_q H_c \\ \hline H_c \end{bmatrix} \quad (8 \times 3) \quad (88)$$

7. The output matrix partitions depend on the type of element using this subroutine. If the element type is a QUAD1 or QUAD2 the three output matrix partitions $[K^{de}]$ are:

$$[K_{ij}^{de}] = [C_i]^T [K^{dq}] [C_j] \quad (89)$$

where i is the pivot point and $j = a, b$, and c .

If the element type is a TRIA1 or TRIA2 the output differential stiffness matrices are calculated using equation 89 above. They are:

$$[K_{ia}] = [C_i]^T [K^{dq}] [C_a] \quad (90)$$

$$[K_{ib}] = [C_i]^T [K^{dq}] [C_b] \quad (91)$$

if i is the pivot point and $i = a$ or $i = b$.

In addition, for the TRIA1 and TRIA2, elements the following matrices are output:

$$[K_{ac}^{de}] = [C_a]^T [K^{dq}] [C_c] \quad (92)$$

$$[K_{bc}^{de}] = [C_b]^T [K^{dq}] [C_c] \quad (93)$$

$$[K_{cc}^{de}] = [C_c]^T [K^{dq}] [C_c] \quad (94)$$

The matrices $[H]^{-1}$ and $[S]$, previously calculated are also output for the TRIA1 and TRIA2 elements.

STRUCTURAL ELEMENT DESCRIPTIONS

8.6.11 Thermal Calculations for the Combination Elements

If the heat transfer system parameter equals +1, the elements are treated exactly like the membrane elements QDMEM and TRMEM. The bending calculations are bypassed and subroutines KQDMEM or KTRMEM are used for calculation of the conductivity matrix terms, subroutine MTRQD is used for the mass calculations, and subroutines SDHTF1, SDHTF2, and SDHTFF are used for stress recovery.

8.7 THE ELASi, MASSi AND DAMPi ELEMENTS

The scalar elements (ELASi, MASSi and DAMPi, $i = 1,2,3,4$) are connected to scalar components of grid points or to scalar points. The ELASi elements contribute only to:

a) the stiffness matrix, $[K_{gg}^x]$, for $i = 1,2,3,4$; and b) to the structural damping matrix, $[K_{gg}^4]$, for $i = 1,2,3$. The MASSi elements contribute only to the mass matrix, $[M_{gg}]$, and the DAMPi elements contribute only to the viscous damping matrix, $[B_{gg}]$.

The scalar elements do not require material or geometric properties in their calculations. Only the ELASi elements are used for stress or force calculations.

8.7.1 Input Data for the ELASi, MASSi and DAMPi Elements

The ECPT/EST entries for the scalar elements are:

<u>Symbol</u>	<u>Description</u>	<u>Elements</u>
SIL_1, SIL_2	Scalar indices of connected grid or scalar points	All
c_1, c_2	Component numbers corresponding to SIL_1 and SIL_2 .	Types 1 and 2
K	Spring constant	All ELASi elements
g_e	Damping factor	ELASi, 2 and 3
S	Stress coefficient	
B	Viscous damping coefficient	All DAMPi elements
m	Mass coefficient	All MASSi elements

8.7.2 ELASi Stiffness Matrix Generation (Subroutine KELAS of Module SMA1)

1. The two connected scalar indices are i_1 and i_2 given by:

$$i_1 = \begin{cases} SIL_1 + (c_1 - 1), & \text{if Point 1 is a grid point} \\ SIL_1, & \text{if Point 1 is a scalar point} \end{cases}$$

STRUCTURAL ELEMENT DESCRIPTIONS

$$i_2 = \begin{cases} \text{SIL}_2 + (c_2 - 1), & \text{if Point 2 is a grid point} \\ \text{SIL}_2, & \text{if Point 2 is a scalar point} \end{cases}$$

2. The following terms are added to the $[K_{gg}^x]$ matrix:

+K in position (i_1, i_1) and in position (i_2, i_2) ,

-K in position (i_2, i_1) and in position (i_1, i_2) .

3. If point 2 is not defined, add +K to position (i_1, i_1) .

4. The damping terms are:

$$K^4 = K g_e. \quad (1)$$

These are added to $[K_{gg}^4]$ in the same manner as the stiffness terms were added to $[K_{gg}^x]$.

8.7.3 MASSi Mass Matrix Generation (Subroutine MASSD of Module SMA2)

These elements are treated like the ELASi elements except the "m" term is added to the four positions in $[M_{gg}]$.

8.7.4 DMAPI Damping Matrix Generation (Subroutine MASSD of Module SMA2)

These elements are treated like the ELASi elements except the "B" term is added to the four positions in $[B_{gg}]$.

8.7.5 ELASi Stress and Force Recovery (Subroutines SELAS1 and SELAS2 of Module SDR2)

The element force is:

$$F = K(u_2 - u_1), \quad (2)$$

where u_1 and u_2 are the displacements at scalar index numbers i_1 and i_2 .

THE ELASi, MASSi AND DAMPi ELEMENTS

The element stress is:

$$\sigma = SF. \quad (3)$$

8.8 CONCENTRATED MASS ELEMENTS CØNM1, CØNM2

Two types of grid point mass data are available. CØNM1 defines the mass matrix directly at the point, with the axes defined by a given local coordinate system. CØNM2 defines the same matrix for a body with a mass and three inertias with a center of gravity offset from the grid point.

8.8.1 ECPT Entries for the CØNM1 Mass Element

<u>Symbol</u>	<u>Description</u>
N_m	Local coordinate system number in which the mass matrix is defined.
N_g, X, Y, Z	Local coordinate system number and basic coordinates of the point.
m_{11} m_{21}, m_{22} m_{31}, m_{32}, m_{33} $m_{41}, m_{42}, m_{43}, m_{44}$ $m_{51}, m_{52}, m_{53}, m_{54}, m_{55}$ $m_{61}, m_{62}, m_{63}, m_{64}, m_{65}, m_{66}$	Terms of mass matrix given in row form (out to the diagonal term).

8.8.2 Mass Matrix Calculations for the CØNM1 Element (Subroutine MCØNMX of Module SMA2)

- Using the symmetrical relationships, fill out the remainder of the 6x6 matrix, [m]:

$$m_{ij} = m_{ji} \quad (1)$$

- Using the basic coordinates of the point and the local coordinate system definition, the 3x3 transformation matrices $[T_g]$ and $[T_m]$ are generated, and the mass matrix in global coordinates is:

STRUCTURAL ELEMENT DESCRIPTIONS

$$[M] = \begin{bmatrix} T_g^T & 0 \\ 0 & T_g^T \end{bmatrix} \begin{bmatrix} T_m & 0 \\ 0 & T_m \end{bmatrix} [m] \begin{bmatrix} T_m^T & 0 \\ 0 & T_m^T \end{bmatrix} \begin{bmatrix} T_g & 0 \\ 0 & T_g \end{bmatrix}, \quad (2)$$

where $[T_g]$ is the transformation from global-to-basic coordinates at the point, and $[T_m]$ is the transformation from the coordinates defined by the mass local system to the basic coordinate system.

8.8.3 ECPT Entries for the C0NM2 Mass Element

<u>Symbol</u>	<u>Description</u>
N_m	Local coordinate system number in which the mass terms are defined.
N_g, X, Y, Z	Local coordinate system number and basic coordinates of the point.
\bar{m}	Concentrated mass
x, y, z	Offset of center of gravity in mass coordinate system
I_{11} I_{21}, I_{22} I_{31}, I_{32}, I_{33}	Inertias about the center of gravity given in row order out to the diagonal term

8.8.4 Mass Matrix Calculations for the C0NM2 Element (Subroutine MC0NMX of Module SMA2)

1. The transformation from the offset to the grid point is:

$$[D] = \begin{bmatrix} 1 & & & 0 & z & -y \\ & 1 & & -z & 0 & x \\ & & 1 & y & -x & 0 \\ - & - & - & - & - & - \\ & & & 1 & & \\ 0 & & & & 1 & \\ & & & & & 1 \end{bmatrix}. \quad (3)$$

2. The mass matrix referenced to the offset point is:

$$[m_o] = \begin{bmatrix} \bar{m} & & & & & \\ & \bar{m} & & & & \\ & & \bar{m} & & & \\ & & & 0 & & \\ & & & & I_{11} & -I_{21} & -I_{31} \\ & & & & -I_{21} & I_{22} & -I_{32} \\ & & & & -I_{31} & -I_{32} & I_{33} \end{bmatrix} \quad (4)$$

3. The mass matrix about the grid point along the given coordinates is:

$$[m] = [D]^T [m_o] [D]. \quad (5)$$

The actual nonzero terms of matrix $[m]$ are calculated directly from the equations:

$$m_{11} = m_{22} = m_{33} = \bar{m}, \quad (6)$$

$$m_{15} = m_{51} = -m_{24} = -m_{42} = +z\bar{m}, \quad (7)$$

$$m_{16} = m_{61} = -m_{34} = -m_{43} = -y\bar{m}, \quad (8)$$

$$m_{26} = m_{62} = -m_{35} = -m_{53} = x\bar{m}, \quad (9)$$

$$m_{44} = I_{11} + (y^2 + z^2)\bar{m}, \quad (10)$$

$$m_{55} = I_{22} + (x^2 + z^2)\bar{m}, \quad (11)$$

$$m_{66} = I_{33} + (x^2 + y^2)\bar{m}, \quad (12)$$

$$m_{45} = m_{54} = -I_{21} - xy\bar{m}, \quad (13)$$

STRUCTURAL ELEMENT DESCRIPTIONS

$$m_{46} = m_{64} = -I_{31} - xz\bar{m} \quad , \quad (14)$$

$$m_{56} = m_{65} = -I_{32} - yz\bar{m} \quad . \quad (15)$$

4. The matrix $[m]$ is transformed back to global coordinates using Equation 2.

8.9 THE CØNEAX ELEMENT

8.9.1 Input Data for the CØNEAX Element

1. The ECPT/EST entries for the CØNEAX element are:

<u>Symbol</u>	<u>Description</u>
SIL_a, SIL_b	Scalar indices of the n^{th} harmonic of the connected rings
n	Harmonic index
r_a, r_b	Radii at points a and b
z_a, z_b	Projected distances along the axis
t_m	Membrane thickness
MAT ID _m	Membrane material identification number
I	Bending coefficient
MAT ID _b	Bending material identification number
t_s	Shear thickness
MAT ID _s	Shear material identification number
Z_1, Z_2	Outer fiber distances for stress calculations
$\theta_i, i = 1 \text{ to } 14$	Angles defining points around element
t_μ	Temperature for material properties
μ	Nonstructural mass

8.9.2 Stiffness Matrix Calculations (Subroutine KCØNE of Module SMA1).

1. The shell orientation is given by:

$$l = \sqrt{(r_b - r_a)^2 + (z_b - z_a)^2}, \quad (1)$$

$$\sin\psi = \frac{r_b - r_a}{l}. \quad (2)$$

$$\cos\psi = \frac{z_b - z_a}{l}. \quad (3)$$

2. The transformation matrix $[E]$ from element coordinates to ring cylindrical coordinates is:

$$[E] = \begin{bmatrix} 0 & \sin \psi & \cos \psi & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & \cos \psi & -\sin \psi & 0 & 0 \\ 0 & 0 & 0 & 0 & \sin \psi \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & \cos \psi \end{bmatrix} \quad (4)$$

3. The serial steps for the balance of the stiffness matrix computations unique to the axisymmetric conical shell element are explicitly described in the NASTRAN Theoretical Manual section 5.9.5.7 (Summary of Procedures).

8.9.3 Mass Matrix Computation (Subroutine MCONE of Module SMA2)

$$[M_{ii}] = \begin{bmatrix} m_i & 0 & 0 & 0 & 0 & 0 \\ 0 & m_i & 0 & 0 & 0 & 0 \\ 0 & 0 & m_i & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad i = a \text{ or } b \quad (5)$$

where

$$\begin{aligned} m_a &= \pi \ell \left(\frac{r_b}{6} + \frac{r_a}{3} \right) (\rho t + \mu) \\ m_b &= \pi \ell \left(\frac{r_b}{3} + \frac{r_a}{6} \right) (\rho t + \mu) \end{aligned} \quad (6)$$

8.9.4 Element Load Calculations (Subroutine CONE of Module SSG1)

The Fourier coefficients of the temperatures are stored in the GPTT data block. The loads are generated by the elements, which reference the connected rings and harmonics indirectly by the grid point scalar indices. The scalar indices are used with the SIL (Scalar Index List) data block to obtain the temperatures. The following steps are used to generate the loads:

1. The data for a logical element are read from the EST data block. The harmonic

THE CØNEAX ELEMENT

number, n , is extracted from the element ID, N_e , by the equation:

$$N_e = 1000N + (n + 1) , \quad (7)$$

where N is the total number of harmonics plus one in the problem.

The temperatures for this particular element and harmonic (T_n^a and T_n^b) are extracted from the GPTT data block. (No default nonzero temperatures are allowed).

2. The following data are generated in the same manner as in the stiffness matrix routine, KCØNE:

- r_a, r_b, z_a, z_b - Ring locations
- l - Linear distance
- $\sin\psi, \cos\psi$ - Inclination functions
- $[E]$ - Element-to-global transformation matrix (6x5)
- t - Element thickness
- $[E_m]$ - Material matrix (3x3)
- $\alpha_s = \alpha_\phi = \alpha$ - Temperature coefficient

3. The geometry coefficients, I_{mn} , are calculated as in the stiffness matrix routine by the equation:

$$I_{mn} = \pi \int_0^l s^m r^{1-n} ds = \frac{\pi}{b} \int_{r_a}^{r_b} s^m r^{1-n} dr , \quad (8)$$

where $r = a + bs$, $a = r_a$, $b = \frac{r_b - r_a}{l}$.

4. The loads are generated in generalized coordinates, $\{P_n^q\}$, with the equations:

$$\{P_{1n}\} = n(I_{01}A_n + I_{11}B_n) , \quad (9)$$

STRUCTURAL ELEMENT DESCRIPTIONS

$$\{P_{2n}\} = n(I_{11}A_n + I_{21}B_n) , \quad (10)$$

$$\{P_{3n}\} = \sin\psi (I_{01}A_n + I_{11}B_n) , \quad (11)$$

$$\{P_{4n}\} = \sin\psi(I_{11}A_n + I_{21}B_n) + I_{00}C_n + I_{10}D_n , \quad (12)$$

$$\{P_{5n}\} = \cos\psi(I_{01}A_n + I_{11}B_n) , \quad (13)$$

$$\{P_{6n}\} = \cos\psi(I_{11}A_n + I_{21}B_n) , \quad (14)$$

$$\{P_{7n}\} = \cos\psi(I_{21}A_n + I_{31}B_n) , \quad (15)$$

$$\{P_{8n}\} = \cos\psi(I_{31}A_n + I_{41}B_n) , \quad (16)$$

$$\{P_{9n}\} = 0 , \quad (17)$$

$$\{P_{10,n}\} = 0 , \quad (18)$$

where

$$A_n = tE_{12}(\alpha_s T_n^a) + tE_{22}(\alpha_\phi T_n^a) , \quad (19)$$

$$B_n = \frac{t}{\ell} (T_n^b - T_n^a)(E_{12}\alpha_s + E_{22}\alpha_\phi) , \quad (20)$$

$$C_n = tE_{11}(\alpha_s T_n^a) + tE_{12}(\alpha_\phi T_n^a) , \quad (21)$$

$$D_n = \frac{t}{\ell} (T_n^b - T_n^a)(E_{11}\alpha_s + E_{12}\alpha_\phi) . \quad (22)$$

THE CONEAX ELEMENT

5. The transformation from generalized coordinates to element coordinates, $[G_{qu}]$, is calculated if no transverse shear material or thickness is given (i.e., $MAT ID_s = 0$ or $t_s = 0$).

$$[H_{qu}] = [H]^{-1}, \quad (23)$$

where $[H]$ is given explicitly in the stiffness matrix calculations.

6. If transverse shear exists, the shear matrix, $[B]$, is generated. Additional material terms are:

$$[D] = I_b[E_b], \text{ where } [E_b] \text{ is the bending material } 3 \times 3 \text{ matrix.}$$

$$G_{11} = G = \text{Shear coefficient of transverse shear material.}$$

The nonzero terms of $[B]$ are:

$$B_{4,1} = B_{9,1} = \frac{1}{Q} [D_{12} n \cos \psi \left(\frac{1}{r_b} - \frac{1}{r_a} \right) - \frac{n}{2} \cos \psi \sin \psi \frac{I_{03}}{\pi} (n_{33} + 2D_{22})], \quad (24)$$

$$B_{4,2} = B_{9,2} = \frac{1}{Q} [D_{12} \frac{n \ell \cos \psi}{r_b} - \frac{1}{2} n \sin \psi \cos \psi \frac{I_{13}}{\pi} (3D_{33} + D_{22}) + \frac{3}{2} n D_{33} \cos \psi \frac{I_{02}}{\pi}], \quad (25)$$

$$B_{4,3} = B_{9,3} = \frac{1}{Q} \left[\frac{1}{2} n^2 D_{33} \cos \psi \frac{I_{03}}{\pi} \right], \quad (26)$$

$$B_{4,4} = B_{9,4} = \frac{1}{Q} \left[\frac{1}{2} n^2 D_{33} \cos \psi \frac{I_{13}}{\pi} \right], \quad (27)$$

$$B_{4,5} = B_{9,5} = \frac{1}{Q} [n^2 D_{12} \left(\frac{1}{r_b} - \frac{1}{r_a} \right) - n^2 \sin \psi \frac{I_{03}}{\pi} (2D_{33} + D_{22})], \quad (28)$$

$$B_{4,6} + B_{9,6} = \frac{1}{Q} [D_{12} \frac{n^2 \ell}{r_b} - n^2 \sin \psi \frac{I_{13}}{\pi} (2D_{33} + D_{22}) + \frac{I_{02}}{\pi} (2n^2 D_{33} + \sin^2 \psi D_{22})], \quad (29)$$

$$B_{4,7} = B_{9,7} = \frac{1}{Q} [2D_{11} (r_a - r_b) + D_{12} \frac{n^2 \ell^2}{r_b} + \frac{2I_{12}}{\pi} (2n^2 D_{33} + \sin^2 \psi D_{22})] \quad (30)$$

$$- n^2 \sin \psi \frac{I_{23}}{\pi} (2D_{33} + D_{22})],$$

$$B_{4,8} = B_{9,8} = \frac{1}{Q} [-D_{11} 6\ell r_b + D_{12} \frac{n^2 \ell^3}{r_b} + \frac{3I_{22}}{\pi} (2n^2 D_{33} + \sin^2 \psi D_{22}) - n^2 \sin \psi \frac{I_{33}}{\pi} (2D_{33} + D_{22})], \quad (31)$$

$$B_{4,9} = B_{9,9} = \frac{1}{Q} [-n \sin \psi \frac{I_{02}}{\pi} (D_{22} + D_{33})], \quad (32)$$

$$B_{4,10} = B_{9,10} = \frac{1}{Q} [n\ell(D_{12} + D_{33}) - n \sin \psi \frac{I_{12}}{\pi} (D_{22} + D_{33})], \quad (33)$$

where

$$Q = \ell t_s G_{11} \frac{r_a + r_b}{2} \left[1 + \frac{I_{02}}{\pi} \frac{n^2 D_{33} + \sin^2 \psi D_{22}}{\ell t_s G_{11} r_a} \right]. \quad (34)$$

7. The transformation $[\bar{H}]$ transforming displacements in the element coordinate system to displacements in generalized coordinates of the power series is:

$$[\bar{H}] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{\cos \psi}{r_a} & 0 & 0 & 0 & \frac{n}{r_a} & 0 & 0 & 0 & 1 & 0 \\ 1 & \ell & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \ell & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \ell & \ell^2 & \ell^3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2\ell & 3\ell^3 & -1 & 0 \\ \frac{\cos \psi}{r_b} & \frac{\ell \cos \psi}{r_b} & 0 & 0 & \frac{n}{r_b} & \frac{n\ell}{r_b} & \frac{n\ell^2}{r_b} & \frac{n\ell^3}{r_b} & 1 & \ell \end{bmatrix} \quad (35)$$

THE CONEAX ELEMENT

8. The transformation $[H_{qu}]$ for nonzero transverse shear is:

$$[H_{qu}] = ([H] - [B])^{-1} \quad (10 \times 10) \quad . \quad (36)$$

9. $[H_{qu}]$ is partitioned into two 10×5 matrices

$$[H_{qu}] = [H_a \mid H_b] \quad . \quad (37)$$

10. The loads in global coordinates are calculated with:

$$\{P_a\} = [E][H_a]^T \{P_n^q\} \quad , \quad (38)$$

$$\{P_b\} = [E][H_b]^T \{P_n^q\} \quad . \quad (39)$$

8.9.5 Element Stress Calculations (Subroutines SCONE1, SCONE2, SCONE3 of Module SDR2)

1. For each element the following quantities are calculated as in section 4.87.9.2:

λ , $\sin\psi$, $\cos\psi$, $[E]$, $[H_a]$, $[H_b]$, $[H_{eq}]$, $[H_{\psi q}]$, $[H_{\chi q}]$ (using $s = \frac{\lambda}{2}$, $r = \frac{r_a + r_b}{2}$).

2. Using the material properties, the following matrices are calculated:

$$[E_m] = \frac{E_1}{(1 - \nu_1^2)} \begin{bmatrix} 1 & -\nu_1 & 0 \\ -\nu_1 & 1 & 0 \\ 0 & 0 & \frac{1 - \nu_1}{2} \end{bmatrix} \quad . \quad (40)$$

STRUCTURAL ELEMENT DESCRIPTIONS

$$[D_b] = \frac{E_2 I}{(1 - \nu_2^2)} \begin{bmatrix} 1 & -\nu_2 & 0 \\ -\nu_2 & 1 & 0 \\ 0 & 0 & \frac{1 - \nu_2}{2} \end{bmatrix}, \quad (41)$$

$$[G_s] = \begin{bmatrix} G_3 & 0 \\ 0 & G_3 \end{bmatrix}, \quad (42)$$

where $[E_m]$, $[D_b]$ and $[G_s]$ are computed for membrane, bending and shear materials respectively.

3. The stress matrices are then calculated:

$$[K_s] = \begin{bmatrix} [E_m] & [H_{eq}] \\ - & - & - & - \\ [D] & [H_{xq}] \\ - & - & - & - \\ t_s [G_s] & [H_{\alpha q}] \end{bmatrix}, \quad (8 \times 10) \quad (43)$$

$$[S_a] = [K_s][H_a][E], (8 \times 6) \quad (44)$$

$$[S_b] = [K_s][H_b][E], (8 \times 6) \quad (45)$$

$$St_1 = \alpha_1 E_{11} + \alpha_2 E_{12}$$

$$St_2 = \alpha_1 E_{12} + \alpha_2 E_{12}$$

$$(\alpha_1 = \alpha_2 \text{ for type 1 materials})$$

4. Each entry in the EST data block contains data pertaining to the n^{th} harmonic motion of the CØNEAX element. The elements in the EST are ordered by harmonic and CØNEAX I.D. number. All harmonic elements for each CØNEAX are grouped together.

a. When the harmonic, n , of an element is zero, this indicates it is the first of a group of elements. Storage space is allotted for fourteen 8 by 1 vectors defining the element forces at points. Two UGV vector data blocks must be used to calculate stresses on points. These data blocks correspond to the two subcases "C" and "S" and are solved simultaneously using the same data.

THE CØNEAX ELEMENT

b. Using the $[S_a]$ and $[S_b]$ matrices and the 6 by 1 displacement vectors, $\{u\}$, by their SIL numbers, stress and force vectors are computed.

If $n \neq 0$:

$$\begin{Bmatrix} \bar{\sigma}_{sn}^c \\ \bar{\sigma}_{\phi n}^c \\ \bar{\sigma}_{s\phi n}^c \\ M_{sn}^c \\ M_{\phi n}^c \\ M_{s\phi n}^c \\ V_{sn}^c \\ V_{\phi n}^c \end{Bmatrix} = [S_{an}] \{u_{an}^c\} + [S_{bn}] \{u_{bn}^c\}, \quad (46)$$

$$\begin{Bmatrix} \bar{\sigma}_{sn}^s \\ \bar{\sigma}_{\phi n}^s \\ \bar{\sigma}_{s\phi n}^s \\ M_{sn}^s \\ M_{\phi n}^s \\ M_{s\phi n}^s \\ V_{sn}^s \\ V_{\phi n}^s \end{Bmatrix} = [S_{an}] \{u_{an}^s\} + [S_{bn}] \{u_{bn}^s\}. \quad (47)$$

If $n = 0$:

$$\begin{Bmatrix} \bar{\sigma}_{so} \\ \bar{\sigma}_{\phi o} \\ \bar{\sigma}_{s\phi o} \\ M_{so} \\ M_{\phi o} \\ M_{s\phi o} \\ V_{so} \\ V_{\phi o} \end{Bmatrix} = [S_{ao}] (\{u_{ao}^s\} + \{u_{ao}^c\}) - [S_{bo}] (\{u_{bo}^s\} + \{u_{bo}^c\}). \quad (48)$$

c. The temperature effects are added using the GPTT data for the two subcases, (T_α^c and T_α^s)

$$(\sigma_{sn}^{c'}) = \sigma_{sn}^c + S_{t1} \bar{T} \quad (49)$$

$$(\sigma_n^{c'}) = \sigma_{\phi n}^c + S_{t2} \bar{T} \quad (50)$$

$$(\sigma_{s\phi n}^{c'}) = \sigma_{s\phi n}^c \quad (51)$$

where

$$\bar{T} = \frac{1}{2} (T_a^n + T_b^n) - T_o, \quad n = 0$$

$$\bar{T} = \frac{1}{2} (T_a^n + T_b^n) \quad n \neq 0$$

The same equations are used for the "S" set and when $n = 0$.

THE CØNEAX ELEMENT

d. The harmonic stresses are calculated by the equations
for $i = 1, 2, 3$:

$$\sigma_{sni}^C = (\sigma_{sn}^{C'}) + \frac{M_{sn}^C c_i}{I} , \quad (52)$$

$$\sigma_{\phi n}^C = (\sigma_{\phi n}^{C'}) + \frac{M_{\phi n}^C c_i}{I} , \quad (53)$$

$$\sigma_{s\phi n}^C = (\sigma_{s\phi n}^{C'}) + \frac{M_{s\phi n}^C c_i}{I} . \quad (54)$$

STRUCTURAL ELEMENT DESCRIPTIONS

The equations are repeated for the S set and when $n = 0$.

e. Principal stresses ($\sigma_1, \sigma_2, \theta, \tau_{\max}$ etc.) are calculated as with the TRIA1 or QUAD1 element, except that when $n \neq 0$ the data are calculated for both the S and C sets.

f. The incremental element stresses or forces for the points on the cone are calculated from the following equations for $j = 1, 2, \dots, 14$:

For $n \neq 0$:

$$\delta\sigma_{sj} = (\sigma_{sn}^C) \cos(n\phi_j) + (\sigma_{sn}^S) \sin(n\phi_j), \quad (55)$$

$$\delta\sigma_{\phi j} = (\sigma_{\phi n}^C) \cos(n\phi_j) + (\sigma_{\phi n}^S) \sin(n\phi_j), \quad (56)$$

$$\delta\sigma_{s\phi j} = (\sigma_{s\phi n}^C) \sin(n\phi_j) - (\sigma_{s\phi n}^S) \cos(n\phi_j), \quad (57)$$

$$\delta M_{sj} = M_{sn}^C \cos(n\phi_j) + M_{sn}^S \sin(n\phi_j), \quad (58)$$

$$\delta M_{\phi j} = M_{\phi n}^C \cos(n\phi_j) + M_{\phi n}^S \sin(n\phi_j), \quad (59)$$

$$\delta M_{s\phi j} = M_{s\phi n}^C \sin(n\phi_j) - M_{s\phi n}^S \cos(n\phi_j), \quad (60)$$

$$\delta V_{sj} = V_{sn}^C \cos(n\phi_j) + V_{sn}^S \sin(n\phi_j), \quad (61)$$

$$\delta V_{\phi j} = V_{\phi n}^C \sin(n\phi_j) - V_{\phi n}^S \cos(n\phi_j). \quad (62)$$

g. The incremental stress and force values are added to the running sums for the points. After the last element is calculated ($n = N$), the forces and stresses for the points are calculated and output. The equations are identical to steps d and e of this section except that 1) the "S" and "C" sets are not used and 2) up to 14 points may be calculated for output for each physical element.

Since the user may leave some spaces blank on the property card for this element, only one of the $\phi_j = 0$ points is used in the calculation.

8.9.6 Differential Stiffness Matrix Calculations (Subroutine DCØNE of Module DSMG1)

The data input from the ECPT to the DCØNE subroutine are the same as those given in section

8.9.1. Additional data for the generation of the differential stiffness matrix are as follows:

$\{u_a^0\}, \{u_b^0\}$ - Displacement vectors of the zero harmonic (extracted from the UGV data block)

T_a^0, T_b^0 - Element loading temperatures of the zero harmonic (extracted from the GPTT data block)

The first part of the calculations involves calculation of the element force components. The following steps are performed with harmonic number $n = 0$.

1. The 10 by 10 transformation matrix $[H_{uq}^0]$ is computed from:

$$[H_{uq}^n] = [\bar{H}_{uq}] + \{H_{uy}\} \{H_{ysq}\}^T \quad (63)$$

where $[\bar{H}_{uq}]$, a 10 by 10 matrix, and $\{H_{uy}\}$, a column vector, are derived in the NASTRAN Theoretical Manual, section 5.9.5.3, and $\{H_{ysq}\}^T$, a row vector, is explicitly written out in Equation 85 of section 5.9 of the NASTRAN Theoretical Manual.

2. The 10 generalized displacement quantities q_i are

$$\{q\} = [H_{uq}^0]^{-1} \begin{pmatrix} [E]^T \{u_a^0\} \\ [E]^T \{u_b^0\} \end{pmatrix} \quad (64)$$

where $\{u_a^0\}$ and $\{u_b^0\}$ are the 6 by 1 displacement vectors, and $[E]$ is calculated as in Equation 4 of Section 8.9.2.

3. The strain coefficients are

$$\begin{pmatrix} \Delta \epsilon_s \\ \Delta \epsilon_\phi \\ \Delta \epsilon_{s\phi} \end{pmatrix} = \{\alpha\} \frac{(T_b^c - T_a^0)}{l} \quad (65)$$

$$\begin{pmatrix} \epsilon_s \\ \epsilon_\phi \\ \epsilon_{s\phi} \end{pmatrix} = T_a^0 \{\alpha\} \quad (66)$$

where T_a^0 and T_b^0 are the loading temperatures at the grid points, the $\{\alpha\}$ vector is obtained from the MPT data block via subroutine MAT and λ is calculated as in Equation 1 of Section 8.9.2.

4. The force coefficients are calculated:

$$a_0 = t_m G_{12} (\sin \psi q_3 + \cos \psi q_5) , \quad (67)$$

$$a_1 = t_m G_{12} (\sin \psi q_4 + \cos \psi q_6) , \quad (68)$$

$$a_2 = t_m G_{12} \cos \psi q_7 , \quad (69)$$

$$a_3 = t_m G_{12} \cos \psi q_8 , \quad (70)$$

$$b_0 = t_m G_{22} (\sin \psi q_3 + \cos \psi q_5) , \quad (71)$$

$$b_1 = t_m G_{22} (\sin \psi q_4 + \cos \psi q_6) , \quad (72)$$

$$b_2 = t_m G_{22} \cos \psi q_7 , \quad (73)$$

$$b_3 = t_m G_{22} \cos \psi q_8 , \quad (74)$$

$$c_0 = t_m G_{11} (q_4 - \epsilon_s) - t_m G_{12} \epsilon_\phi , \quad (75)$$

$$c_1 = -t_m G_{11} \Delta \epsilon_s - t_m G_{12} \Delta \epsilon_\phi , \quad (76)$$

$$d_0 = t_m G_{12} (q_4 - \epsilon_s) - t_m G_{22} \epsilon_\phi , \quad (77)$$

$$d_1 = -t_m G_{12} \Delta \epsilon_s - t_m G_{22} \Delta \epsilon_\phi , \quad (78)$$

THE CØNEAX ELEMENT

where G_{11} , G_{12} and G_{22} are elements of the 3 by 3 symmetric material properties matrix, $[G]$.

5. The geometry coefficients are calculated:

$$I_{mn} = \int_0^{\ell} s^m r^{1-n} ds \quad \begin{cases} m = 0, 1, \dots, 9 \\ n = 0, 1, 2, 3 \end{cases} ; \quad (79)$$

where

$$r = r_a + \left(\frac{r_b - r_a}{\ell} \right) s . \quad (80)$$

An explicit formula for the evaluation of I_{mn} is given in the NASTRAN Theoretical Manual, section 5.9.5.8.

6. The following coefficients for the computation of the differential stiffness matrix are calculated:

$$\begin{aligned} A_{mn} = & a_0 I_{m, n+1} + a_1 I_{m+1, n+1} + a_2 I_{m+2, n+1} \\ & + a_3 I_{m+3, n+1} + c_0 I_{m, n} + c_1 I_{m+1, n} , \end{aligned} \quad (81)$$

$$\begin{aligned} B_{mn} = & b_0 I_{m, n+1} + b_1 I_{m+1, n+1} + b_2 I_{m+2, n+1} \\ & + b_3 I_{m+3, n+1} + d_0 I_{m, n} + d_1 I_{m+1, n} , \end{aligned} \quad (82)$$

$$C_{mn} = A_{mn} + B_{mn} , \quad (83)$$

where $m = 0, 1, \dots, 6$; $n = 0, 1, 2$.

Note: The index n used above is a dummy index and is not the harmonic number.

The second part of the calculations involves generating the differential stiffness matrix. The remaining steps use n as the harmonic number of the element.

1. The nonzero elements of the symmetric differential stiffness matrix $[K^{qd}]$, in generalized coordinates, are given in Table 1c below.

Table 1c. Nonzero Elements of the Differential Stiffness Matrix, $[K^{qd}]$.

$$K_{11}^{qd} = \cos^2 \psi B_{02} + \frac{1}{4} \sin^2 \psi C_{02}$$

$$K_{12}^{qd} = \cos^2 \psi B_{12} + \frac{1}{4} \sin \psi C_{01} + \frac{1}{4} \sin^2 \psi C_{12}$$

$$K_{13}^{qd} = \frac{1}{4} n \sin \psi C_{02}$$

$$K_{14}^{qd} = \frac{1}{4} n \sin \psi C_{12}$$

$$K_{15}^{qd} = n \cos \psi B_{02}$$

$$K_{16}^{qd} = n \cos \psi B_{12}$$

$$K_{17}^{qd} = n \cos \psi B_{22}$$

$$K_{18}^{qd} = n \cos \psi B_{32}$$

$$K_{22}^{qd} = \cos^2 \psi B_{22} + \frac{1}{4} C_{00} + \frac{1}{2} \sin \psi C_{11} + \frac{1}{4} \sin^2 \psi C_{22}$$

$$K_{23}^{qd} = \frac{1}{4} n C_{01} + \frac{1}{4} n \sin \psi C_{12}$$

$$K_{24}^{qd} = \frac{1}{4} n C_{11} + \frac{1}{4} n \sin \psi C_{22}$$

$$K_{25}^{qd} = n \cos \psi B_{12}$$

$$K_{26}^{qd} = n \cos \psi B_{22}$$

$$K_{27}^{qd} = n \cos \psi B_{32}$$

$$K_{28}^{qd} = n \cos \psi B_{42}$$

$$K_{33}^{qd} = \frac{1}{4} n^2 C_{02}$$

$$K_{34}^{qd} = \frac{1}{4} n^2 C_{12}$$

Table 1c (con'd). Elements of the Differential Stiffness Matrix, $[K^{qd}]$.

$$K_{44}^{qd} = \frac{1}{4} n^2 C_{22}$$

$$K_{55}^{qd} = n^2 B_{02}$$

$$K_{56}^{qd} = n^2 B_{12}$$

$$K_{57}^{qd} = n^2 B_{22}$$

$$K_{58}^{qd} = n^2 B_{32}$$

$$K_{66}^{qd} = A_{00} + n^2 B_{22}$$

$$K_{67}^{qd} = 2A_{10} + n^2 B_{32}$$

$$K_{68}^{qd} = 3A_{20} + n^2 B_{42}$$

$$K_{77}^{qd} = 4A_{20} + n^2 B_{42}$$

$$K_{78}^{qd} = 6A_{30} + n^2 B_{52}$$

$$K_{88}^{qd} = 9A_{40} + n^2 B_{62}$$

The formulas for $n = 0$ are the same as in Table 1c except that they are all multiplied by 2. The nonzero terms for $n = 0$ fall into two uncoupled sets which are

(11) (12)
 (22)

Effect on
Twisting

(66) (67) (68)
 (77) (78)
 (88)

Effect on Axisymmetric Deformation

2. The 10 by 10 transformation matrix, $[H_{uq}^n]$, from generalized coordinates to element coordinates, for the n^{th} harmonic is computed as in Equation 63. The matrix is inverted and partitioned into two 10 by 5 matrices:

$$[H_{uq}^n]^{-1} \Rightarrow [H_a \mid H_b] \quad . \quad (84)$$

3. The 6 by 6 differential stiffness matrices in global coordinates are:

$$[K_{ij}^d] = [E] [H_i]^T [K^{qd}] [H_j] [E]^T \quad , \quad (85)$$

where i = pivot grid point; $j = a, b$; and $[E]$ is computed as in Equation 4 of Section 8.9.2.

THE TRIARG ELEMENT

8.10 THE TRIARG ELEMENT

8.10.1 Input Data for the TRIARG Element

1. The ECPT/EST entries for the axisymmetric triangular ring (TRIARG) element are:

<u>Symbol</u>	<u>Descriptions</u>
SIL ₁ , SIL ₂ , SIL ₃	Scalar index numbers for the three grid points.
Y	Material property orientation angle (degrees)
Mat I.D.	Material property identification number.
$\left. \begin{array}{l} N_1, X_1, Y_1, Z_1 \\ N_2, X_2, Y_2, Z_2 \\ N_3, X_3, Y_3, Z_3 \end{array} \right\}$	Local coordinate system number and location in basic coordinates of the three grid points.
t _μ	Element temperature for material properties.

For this element, Y_i must equal zero for i = 1, 2 and 3 and, we define:

$$\{R_s\} = \begin{Bmatrix} X_1 \\ X_2 \\ X_3 \end{Bmatrix} \quad (1)$$

$$\{Z_s\} = \begin{Bmatrix} Z_{s1} \\ Z_{s2} \\ Z_{s3} \end{Bmatrix} = \begin{Bmatrix} Z_1 \\ Z_2 \\ Z_3 \end{Bmatrix} \quad (2)$$

2. Coordinate system data

The location (X_i, Y_i, Z_i) and local coordinate system number (N_i) of each grid point are used to calculate the 3 by 3 global-to-basic coordinate system transformation matrices, [T_i], i = 1, 2, 3.

3. Material data

The material property identification number, Mat I.D., and the element temperature for material properties, t_μ, are used to select the following data items. For this element, material properties may be defined on a MAT1 or MAT3 but not a MAT2 bulk data card.

STRUCTURAL ELEMENT DESCRIPTIONS

<u>Symbol</u>	<u>Description</u>
E_r, E_θ, E_z	Young's moduli in the radial, tangential and axial directions respectively.
$\nu_{r\theta}, \nu_{\theta z}, \nu_{rz}$	Poisson's ratios in the three directions indicated.
ρ	Mass density
$G_{r\theta}, G_{\theta z}, G_{rz}$	Shear moduli in the three directions indicated.
$\alpha_r, \alpha_\theta, \alpha_z$	Coefficients of thermal expansion in the three directions indicated.
T_0	Thermal expansion reference temperature.
g_e	Structural element damping coefficient.

8.10.2 General Geometric Calculations

1. Local coordinate calculations are:

$$Z_{\min} = \text{minimum of } (Z_{s1}, Z_{s2}, Z_{s3}), \quad (3)$$

$$\{R_L\} = \{R_s\}, \quad (4)$$

$$\{Z_L\} = \{Z_s\} - \begin{pmatrix} Z_{\min} \\ Z_{\min} \\ Z_{\min} \end{pmatrix}. \quad (5)$$

2. The transformation from field coordinates to grid point degrees of freedom is given by (R_{L1} and Z_{L1} are the i^{th} components of $\{R_L\}$ and $\{Z_L\}$ respectively):

$$[T_{\beta q}] = \begin{bmatrix} 1 & R_{L1} & Z_{L1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & R_{L1} & Z_{L1} \\ 1 & R_{L2} & Z_{L2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & R_{L2} & Z_{L2} \\ 1 & R_{L3} & Z_{L3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & R_{L3} & Z_{L3} \end{bmatrix} \quad 6 \times 6, \quad (6)$$

THE TRIARG ELEMENT

$$[\Gamma_{\beta q}] = [\bar{\Gamma}_{\beta q}]^{-1}. \quad (7)$$

3. The transformation matrix from two to three degrees of freedom per point is:

$$[\Gamma_{qs}] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad 6 \times 9. \quad (8)$$

8.10.3 Integral Calculations

1. The integrals over the area of the cross-section are of the form:

$$\delta_{ij} = r^i z^j dz dr, \quad (9)$$

for the values:

$$\delta_{-10}, \delta_{-11}, \delta_{00}, \delta_{10}, \delta_{20}, \delta_{30}, \delta_{01}, \delta_{11}, \delta_{21}, \delta_{02}, \delta_{12}, \delta_{-12}.$$

To accomplish this we integrate in two parts:

a. From line $Z = K_{12}r + m_{12}$ to line $Z = K_{13}r + m_{13}$

where

$$K_{ij} = \frac{Z_{Lj} - Z_{Li}}{R_{Lj} - R_{Li}}, \quad (10)$$

and

$$m_{ij} = - \frac{R_{Li}Z_{Lj} - R_{Lj}Z_{Li}}{R_{Lj} - R_{Li}}, \quad (11)$$

and from point R_{L1} to point R_{L3} .

b. From line $Z = K_{12}r + m_{12}$ to $Z = K_{32}r + m_{32}$ and from point R_{L3} to R_{L2} .

For the case where

$$R_{L1} = R_{L2} \text{ or } \left| \frac{R_{L2} - R_{L1}}{R_{L2}} \right| < 10^{-5}$$

we must integrate differently, that is, from line $Z = K_{32}r + m_{32}$ to $Z = K_{13}r + m_{13}$ and from R_{L1} to R_{L3} .

2. After the integrals are computed, a check is made to determine if an excessive amount of round-off error occurred. If round-off was excessive, an approximate integral can be calculated.

These tests are:

If any $\delta_{ij} < 0$, then approximation must be used.

If $\sigma_{12} \leq \sigma_{02}$, or $\delta_{-12} \geq \delta_{12}$, or $\delta_{-12} > \delta_{02}$, then approximation must be used.

If $\Delta r \leq \hat{r}$ or $\Delta Z \leq \hat{Z}$, then approximation must be used. The terms Δr , ΔZ , \hat{r} and \hat{Z} are:

$$\Delta r = \max.(|R_{L1} - R_{L2}|, |R_{L2} - R_{L3}|, |R_{L3} - R_{L1}|), \quad (12)$$

$$\hat{r} = [\min.(R_{L1}, R_{L2}, R_{L3})]/10, \quad (13)$$

$$\Delta Z = \max.(|Z_{L1} - Z_{L2}|, |Z_{L2} - Z_{L3}|, |Z_{L3} - Z_{L1}|), \quad (14)$$

$$\hat{Z} = [\min.(Z_{L1}, Z_{L2}, Z_{L3})]/10. \quad (15)$$

The approximation is:

$$\sigma_{ij} = (r_a)^i (z_a)^j A, \quad (16)$$

where

$$r_a = \frac{1}{3} [R_{L1} + R_{L2} + R_{L3}], \quad (17)$$

$$z_a = \frac{1}{3} [Z_{L1} + Z_{L2} + Z_{L3}], \quad (18)$$

$$A = \frac{1}{2} [R_{L1} (Z_{L2} - Z_{L3}) + R_{L2} (Z_{L3} - Z_{L1}) + R_{L3} (Z_{L1} - Z_{L2})]. \quad (19)$$

3. Form the matrix of integrals:

$$[\tilde{D}] = 2\pi \begin{bmatrix} 0 & \delta_{10} & 0 & 0 & 0 & 0 \\ \delta_{00} & \delta_{10} & \delta_{01} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \delta_{10} \\ 0 & 0 & \delta_{10} & 0 & \delta_{10} & 0 \end{bmatrix}_{4 \times 6} \quad (20)$$

8.10.4 Elastic Constants Matrix Calculations

1. Generate the transformation from material axis to element geometric axis:

$$[T_{eo}] = \begin{bmatrix} \cos^2 \gamma & 0 & \sin^2 \gamma & \sin \gamma \cos \gamma \\ 0 & 1 & 0 & 0 \\ \sin^2 \gamma & 0 & \cos^2 \gamma & -\sin \gamma \cos \gamma \\ -2 \sin \gamma \cos \gamma & 0 & 2 \sin \gamma \cos \gamma & \cos^2 \gamma - \sin^2 \gamma \end{bmatrix}_{4 \times 4} \quad (21)$$

2. Generate the matrix of elastic constants for an orthotropic body with respect to cylindrical coordinates:

STRUCTURAL ELEMENT DESCRIPTIONS

$$[E_m] = \frac{1}{\Delta} \begin{bmatrix} E_r(1-\nu_{\theta z}\nu_{z\theta}) & E_r(\nu_{\theta r}+\nu_{zr}\nu_{\theta z}) & E_\theta(1-\nu_{rz}\nu_{zr}) & 0 \\ E_r(\nu_{\theta r}+\nu_{zr}\nu_{\theta z}) & E_r(\nu_{zr}+\nu_{\theta r}\nu_{z\theta}) & E_\theta(\nu_{z\theta}+\nu_{r\theta}\nu_{zr}) & 0 \\ E_\theta(1-\nu_{rz}\nu_{zr}) & E_\theta(\nu_{z\theta}+\nu_{r\theta}\nu_{zr}) & E_z(1-\nu_{r\theta}\nu_{\theta r}) & 0 \\ 0 & 0 & 0 & G_{rz}\Delta \end{bmatrix} \begin{matrix} \text{(Symmetric)} \\ \\ \\ \end{matrix} \quad , \quad (22)$$

4x4

where

$$\nu_{\theta r} = \nu_{r\theta} E_\theta/E_r \quad , \quad (23)$$

$$\nu_{z\theta} = \nu_{\theta z} E_z/E_\theta \quad , \quad (24)$$

$$\nu_{rz} = \nu_{zr} E_r/E_z \quad , \quad (25)$$

$$\Delta = 1 - \nu_{r\theta} \nu_{\theta r} - \nu_{\theta z} \nu_{z\theta} - \nu_{zr} \nu_{rz} - \nu_{r\theta} \nu_{\theta z} \nu_{zr} - \nu_{rz} \nu_{\theta r} \nu_{z\theta} \quad . \quad (26)$$

3. Calculate the elastic constants matrix in element geometric axes:

$$[E_g] = [T_{eo}]^T [E_m] [T_{eo}] \quad . \quad (27)$$

8.10.5 Stiffness Matrix Generation (Subroutine KTRIRG of Module SMA1)

1. Generate the element stiffness matrix in field coordinates:

THE TRIANG ELEMENT

$$[\bar{K}] =$$

$$2\pi \begin{bmatrix} E_{22}\delta_{-10} & & & & & \\ (E_{12}+E_{22})\delta_{00} & (E_{11}+2E_{12}+E_{22})\delta_{10} & & & & \\ E_{22}\delta_{-11}+E_{24}\delta_{00} & (E_{12}+E_{22})\delta_{01}+(E_{14}+E_{24})\delta_{10} & E_{22}\delta_{-12}+2E_{24}\delta_{01}+E_{44}\delta_{10} & & & \\ 0 & 0 & 0 & 0 & & \\ E_{24}\delta_{00} & (E_{14}+E_{24})\delta_{10} & E_{24}\delta_{01}+E_{44}\delta_{10} & 0 & E_{44}\delta_{10} & \\ E_{23}\delta_{00} & (E_{13}+E_{23})\delta_{10} & E_{23}\delta_{01}+E_{34}\delta_{10} & 0 & E_{34}\delta_{10} & E_{33}\delta_{10} \end{bmatrix}_{6 \times 6} \quad (28)$$

where E_{ij} is an element of $[E_g]$.

2. Transform the element stiffness matrix from field coordinates to grid point degrees of freedom:

$$[\bar{K}] = [r_{\beta q}]^T [\tilde{K}] [r_{\beta q}]. \quad (29)$$

3. Transform the element stiffness matrix from two to three degrees of freedom per point.

$$[K] = [r_{qs}]^T [\bar{K}] [r_{qs}]. \quad (30)$$

4. The 3 by 3 partitions $[K_{pj}^3]$ of $[K]$ corresponding to the pivot point p are transformed:

$$[K_{pj}^3] = [T_p]^T [K_{pj}^3] [T_j], \quad j = 1, 2, 3 \quad (31)$$

5. Finally these 3 by 3 partitions are expanded to 6 by 6 partitions:

$$[K_{pj}] = \begin{bmatrix} K_{pj}^3 & & 0 \\ & \ddots & \\ 0 & & 0 \end{bmatrix} \quad (32)$$

8.10.6 Mass Matrix Calculations (Subroutine MTRIRG of Module SMA2)

1. Generate the consistent mass matrix in field coordinates:

$$[\tilde{M}] = 2\pi \begin{bmatrix} \bar{m}_{11}\delta_{10} & & & & & \\ \bar{m}_{11}\delta_{20} & \bar{m}_{11}\delta_{30} & & & & \\ \bar{m}_{11}\delta_{11} & \bar{m}_{11}\delta_{21} & \bar{m}_{11}\delta_{12} & & & \\ 0 & 0 & 0 & \bar{m}_{22}\delta_{10} & & \\ 0 & 0 & 0 & \bar{m}_{22}\delta_{20} & \bar{m}_{22}\delta_{30} & \\ 0 & 0 & 0 & \bar{m}_{22}\delta_{11} & \bar{m}_{22}\delta_{21} & \bar{m}_{22}\delta_{12} \end{bmatrix} \quad \begin{matrix} \text{(Symmetric)} \\ \\ \\ 6 \times 6 \end{matrix} \quad (33)$$

where

$$[\bar{m}] = \begin{bmatrix} \rho & 0 \\ 0 & \rho \end{bmatrix} = \begin{bmatrix} \bar{m}_{11} & 0 \\ 0 & \bar{m}_{22} \end{bmatrix} \quad (34)$$

2. Transform the mass matrix from field coordinates to grid point degrees of freedom:

$$[\tilde{M}] = [r_{\beta q}]^T [\tilde{M}] [r_{\beta q}] \quad (35)$$

3. Transform the mass matrix from two to three degrees of freedom per point:

$$[M] = [r_{qs}]^T [\tilde{M}] [r_{qs}] \quad (36)$$

4. The 6 by 6 partitions, $[M_{pj}]$, are calculated as in Equations 31 and 32.

8.10.7 Thermal Load Calculations (Subroutine TTRIRG of Module SSG1)

1. Form the vector of thermal strains:

$$\{\bar{\alpha}\} = (T_{avg} - T_o) \begin{Bmatrix} \alpha_r \\ \alpha_\theta \\ \alpha_z \\ 0 \end{Bmatrix} \quad (4 \times 1). \quad (37)$$

where T_{avg} is the average loading temperature at the grid points.

2. Compute thermal load vector in grid point degrees of freedom:

$$\{\bar{F}_T\} = [r_{\beta q}]^T [\bar{D}]^T [E_g] \{\bar{\alpha}\} \quad (6 \times 1). \quad (38)$$

3. Transform thermal load from two to three degrees of freedom per point

$$\{F_T\} = [r_{qs}]^T \{\bar{F}_T\} \quad (9 \times 1). \quad (39)$$

4. Each partition, $\{F_T^3\}$, of length 3 of $\{F_T\}$ is transformed to global coordinates by

$$\{F_T^3\}_g = [T_i]^T \{F_T^3\}. \quad (40)$$

5. These vectors are added to the overall load vector, $\{P_g\}$.

8.10.8 Element Force and Stress Calculations (Subroutines STRIR1 and STRIR2 of Module SDR2)

Element stress and force data items are calculated in two phases. The first phase (subroutine STRIR1) calculates the element stiffness and stress matrices. The second phase (subroutine STRIR2) calculates the actual element forces and stresses from the various subcase displacement vectors.

STRUCTURAL ELEMENT DESCRIPTIONS

Phase 1 calculations are as follows:

1. Form the element stiffness matrix, $[K]$, as in section 4.87.10.5.

2. Compute the constants:

$$r_o = \frac{1}{3} \sum_{i=1}^3 R_{Li} \quad (41)$$

$$z_o = \frac{1}{3} \sum_{i=1}^3 Z_{Li} \quad (42)$$

3. Form the $[D_o]$ matrix:

$$[D_o] = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{1}{r_o} & 1 & \frac{z_o}{r_o} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \quad 4 \times 6 \quad (43)$$

4. Compute the stress matrix

$$[\bar{S}] = [E_g][D_o][r_{\beta q}] \quad (44)$$

5. Transform the stress matrix from two to three degrees of freedom per point:

$$[S] = [\bar{S}][r_{qs}] \quad (45)$$

6. Transform the stress matrix from basic to local coordinates:

$$[S]_L = [S][T_{123}] \quad (46)$$

THE TRIANG ELEMENT

where

$$[T_{123}] = \begin{bmatrix} T_1 & 0 & 0 \\ 0 & T_2 & 0 \\ 0 & 0 & T_3 \end{bmatrix} . \quad (47)$$

7. Compute the thermal stress vector

$$\{T_s\} = [E_g]\{\bar{\alpha}\} . \quad (48)$$

Phase 2 calculations are as follows:

1. Extract the displacement vector, $\{\Delta\}$, at the three translational components of the grid points from the global displacement vector.

2. Calculate the element forces:

$$\{P\} = [K]\{\Delta\} . \quad (49)$$

3. Calculate the element stresses:

$$\{\sigma\} = [S]_e\{\Delta\} - T_d\{T_s\} , \quad (50)$$

where

$$T_d = T_{avg} - T_o . \quad (51)$$

8.10.9 Thermal Analysis Calculations for the TRIARG and TRAPRG Elements

If a heat transfer problem is being solved, special code and subroutines are used for these elements. The following checks are made on the geometry:

$$\left. \begin{array}{l} y = 0 \\ x_i > 0 \end{array} \right\} i = 1, 2, 3 \text{ (or 4)}.$$

If these conditions are not met, a fatal error exists. The order of the grid points is also checked. The following equation must be true:

$$(x_s - x_r)(z_t - z_s) - (x_t - x_s)(z_s - z_r) > 0,$$

where the indices r, s, t correspond to three grid points in the element where

r, s, t = 1, 2, 3 for triangles,

$$\text{or } \left. \begin{array}{l} s, t = 1, 2, 3 \\ 2, 3, 4 \\ 3, 4, 1 \\ 4, 1, 2 \end{array} \right\} \text{ for trapezoids.}$$

The conduction matrix for TRIARG is computed by calling for a TRMEM, whose thickness is calculated from

$$t = 2\pi(x_1 + x_2 + x_3)/3.$$

To compute the conduction matrix for TRAPRG, it will be divided into overlapping triangles. The mapping is exactly the same as the mapping of quadrilaterals into triangles. The material orientation angles for the triangles must be computed as was done for QDMEM. The thickness of each of the triangles is given by

$$t = \pi(x_r + x_s + x_t)/3,$$

where r, s, t, are the three vertex indices used from the mapping matrix. The KTRMEM subroutine is then called four times.

THE TRIARG ELEMENT

$$\left. \begin{array}{l} y = 0 \\ x_i > 0 \end{array} \right\} i = 1, 2, 3 \text{ (or 4)}.$$

If these conditions are not met, a fatal error exists. The order of the grid points is also checked. The following equation must be true:

$$(x_s - x_r)(z_t - z_s) - (x_t - x_s)(z_s - z_r) > 0,$$

where the indices r, s, t correspond to three grid points in the element where

$$\begin{array}{l} r, s, t = 1, 2, 3 \text{ for triangles,} \\ \text{or } r, s, t = \left. \begin{array}{l} 1, 2, 3 \\ 2, 3, 4 \\ 3, 4, 1 \\ 4, 1, 2 \end{array} \right\} \text{ for trapezoids.} \end{array}$$

The thermal "stiffness" matrices are generated by subroutine HRING of module SMA1. The conduction matrix for TRIARG is computed by calling for a TRMEM, whose thickness is calculated from

$$t = 2\pi(x_1 + x_2 + x_3)/3.$$

To compute the conduction matrix for TRAPRG, it will be divided into overlapping triangles. The mapping is exactly the same as the mapping of quadrilaterals into triangles. The material orientation angles for the triangles must be computed as was done for QDMEM. The thickness of each of the triangles is given by:

$$t = \pi(x_r + x_s + x_t)/3,$$

where r, s, t are the three vertex indices used from the mapping matrix. The KTRMEM subroutine is then called four times.

The thermal "mass" matrices are generated in a similar manner. Subroutine MRING in module SMA2 rearranges the ECPT data into the TRMEM format with the equivalent "thickness" given by:

$$t = \frac{2\pi}{3} (x_1 + x_2 + x_3).$$

STRUCTURAL ELEMENT DESCRIPTIONS

Subroutine MASSTQ generates the scalar mass terms for each triangle and inserts them into the BGG matrix. The TRIAG element is converted into a triangle once. The TRAPRG element is arranged into the TRMEM format for each of its four overlapping subtriangles; each triangle having a different equivalent "thickness," t , given by the equation above.

The thermal "stress" output data calculations are performed by subroutines SDHTF1, SDHTF2, and SDHTFF in module SDR2. The TRIARG and TRAPRG elements are processed with the same equations as the TRMEM or QDMEM elements, except that the value of the "thickness" t is calculated as above.

THE TRAPRG ELEMENT

8.11 THE TRAPRG ELEMENT

8.11.1 Input Data for the TRAPRG Element

1. The ECPT/EST entries for the axisymmetric trapezoidal ring (TRAPRG) element are

<u>Symbol</u>	<u>Description</u>
SIL ₁ , SIL ₂ , SIL ₃ , SIL ₄	Scalar index numbers for the four grid points.
γ	Material property orientation angle (degrees).
Mat. I.D.	Material property identification number.
$\left. \begin{array}{l} N_1, X_1, Y_1, Z_1 \\ N_2, X_2, Y_2, Z_2 \\ N_3, X_3, Y_3, Z_3 \\ N_4, X_4, Y_4, Z_4 \end{array} \right\}$	Local coordinate system number and location in basic coordinates of the four grid points.
t_μ	Element temperature for material properties

For this element Y_i must equal zero for $i = 1, 2, 3$ and 4 , and we define:

$$\{R_s\} = \begin{Bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{Bmatrix}, \quad (1)$$

$$Z_s = \begin{Bmatrix} Z_{s1} \\ Z_{s2} \\ Z_{s3} \\ Z_{s4} \end{Bmatrix} = \begin{Bmatrix} Z_1 \\ Z_2 \\ Z_3 \\ Z_4 \end{Bmatrix}. \quad (2)$$

2. Geometry Input

The location (X_i, Y_i, Z_i) and local coordinate system number (N_i) of each grid point are used to calculate the 3 by 3 global-to-basic coordinate system transformation matrices, $[T_i]$, $i = 1, 2, 3, 4$.

3. Material Property Input

The material property input for the TRAPRG element is the same as that for the TRIARG element (see section 4.87.10.1).

8.11.2 General Calculations

1. Local coordinate calculations are:

$$Z_{\min} = \text{minimum of } (Z_{s1}, Z_{s2}, Z_{s3}, Z_{s4}), \quad (3)$$

$$\{R_L\} = \{R_S\}, \quad (4)$$

$$\{Z_L\} = \{Z_S\} - \begin{pmatrix} Z_{\min} \\ Z_{\min} \\ Z_{\min} \\ Z_{\min} \end{pmatrix}. \quad (5)$$

Let R_{Li} and Z_{Li} be the i^{th} component of $\{R_L\}$ and $\{Z_L\}$ respectively. To insure the user has input his grid points in accordance with the restrictions set down in section 2 of the User's Manual, the following tests are made:

If $R_{L1} \geq R_{L2}$ or $R_{L4} \geq R_{L3}$ or $Z_{L4} \leq Z_{L1}$, then the user has violated the restriction that the grid points be ordered in a counterclockwise manner and a user fatal error occurs.

If $|Z_{L1} - Z_{L2}| > .001$ or $|Z_{L3} - Z_{L4}| > .001$, then the restriction that the line joining grid points 1 and 2 and the line joining grid points 3 and 4 be parallel to the r-axis has been violated and a user fatal error occurs.

2. Test for a rectangle. Define:

$$R_{M14} = (R_{L1} + R_{L4})/2, \quad (6)$$

THE TRAPRG ELEMENT

$$R_{M23} = (R_{L2} + R_{L3})/2. \quad (7)$$

$$\text{If } \left| \frac{R_{L1} - R_{L4}}{R_{M14}} \right| < 0.005 \text{ then } R_{L1} = R_{L4} = R_{M14}.$$

$$\text{If } \left| \frac{R_{L2} - R_{L3}}{R_{M23}} \right| < 0.005 \text{ then } R_{L2} = R_{L3} = R_{M23}.$$

If $R_{L1} = R_{L4}$ and $R_{L2} = R_{L3}$, then the element is rectangular.

If $R_{L1} = R_{L4} = 0$, then the element is a core element.

3. Generate the transformation matrix from field coordinates to grid point degrees of freedom:

$$[H] = \begin{bmatrix} 1 & R_{L1} & Z_{L1} & R_{L1}Z_{L1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & R_{L1} & Z_{L1} & R_{L1}Z_{L1} \\ 1 & R_{L2} & Z_{L1} & R_{L2}Z_{L1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & R_{L2} & R_{L1} & R_{L2}Z_{L1} \\ 1 & R_{L3} & Z_{L4} & R_{L3}Z_{L4} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & R_{L3} & Z_{L4} & R_{L3}Z_{L4} \\ 1 & R_{L4} & Z_{L4} & R_{L4}Z_{L4} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & R_{L4} & Z_{L4} & R_{L4}Z_{L4} \end{bmatrix} \quad (8)$$

8x8

$$[H] = [H]^{-1}. \quad (9)$$

4. Generate the transformation matrix from two to three degrees of freedom per point:

$$[r_{qs}] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (10)$$

8x12

5. If the element is a core element, then:

$$\begin{aligned} h_{1j} &= h_{3j} = 0 & j &= 1, 2, \dots, 8, \\ h_{i1} &= h_{i7} = 0 & i &= 1, 2, \dots, 8. \end{aligned} \quad (11)$$

where h_{ij} is an element of $[H]$.

8.11.3 Integral Calculations

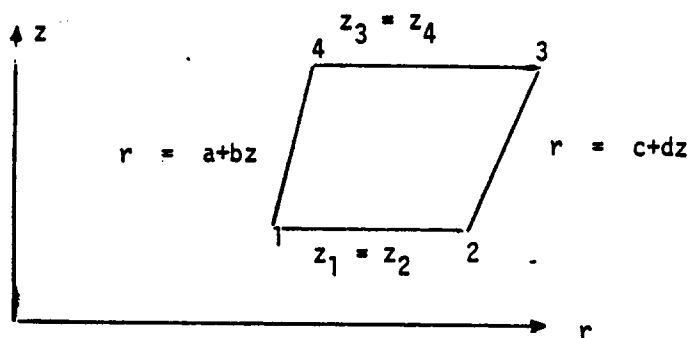
a. Compute the integrals over the cross-section of the trapezoid that are of the form:

$$I_{pq} = \iint_{rz} r^p z^q dr dz \quad (12)$$

for the values:

$$I_{-10}, I_{-11}, I_{-12}, I_{00}, I_{01}, I_{02}, I_{10}, I_{11}, I_{12}, I_{20}, I_{21}, I_{22}, I_{30}, I_{31}, I_{32}.$$

The limits of integration are chosen depending on the geometric shape of the trapezoid.



If the lines between points 1 and 4 and between points 2 and 3 are defined by the equations

$$r = a + bz, \quad (13)$$

$$r = c + dz, \quad (14)$$

respectively, then:

$$a = R_{L1} - \left(\frac{R_{L4} - R_{L1}}{Z_{L4} - Z_{L1}} \right) Z_{L1}, \quad (15)$$

$$b = \frac{R_{L4} - R_{L1}}{Z_{L4} - Z_{L1}}, \quad (16)$$

$$c = R_{L2} - \left(\frac{R_{L3} - R_{L2}}{Z_{L3} - Z_{L2}} \right) Z_{L2}, \quad (17)$$

$$d = \frac{R_{L3} - R_{L2}}{Z_{L3} - Z_{L2}}. \quad (18)$$

In general, the integration takes the form:

$$I_{pq} = \int_{z_1}^{z_4} \int_{a+bz}^{c+dz} r^p z^q dr dz. \quad (19)$$

For the case with the side $r = c + dz$ parallel to the axis of symmetry (the z axis) we have:

$$I_{pq} = \int_{z_1}^{z_4} \int_{a+bz}^c r^p z^q dr dz. \quad (20)$$

For the case with the side $r = a + bz$ parallel to the axis of symmetry we have:

$$I_{pq} = \int_{z_1}^{z_4} \int_a^{c+dz} r^p z^q dr dz. \quad (21)$$

And finally for the rectangle, the integration takes the form

$$I_{pq} = \int_{z_1}^{z_4} \int_a^c r^p z^q dr dz. \quad (22)$$

8.11.4 Elastic Constants Matrix Calculation

The elastic constants matrix in element coordinates, $[E_g]$, for the TRAPRG element is calculated identically to the elastic constants matrix for the TRIARG element (see Section 8.10.4).

8.11.5 Stiffness Matrix Generation (Subroutine KTRAPR of Module SMA1)

1. Generate the terms of the symmetric element stiffness matrix in field coordinates as shown in Table 2. Each term must be multiplied by 2π to form $[\tilde{K}]$.
2. Transform the element stiffness matrix from field coordinates to grid point degrees of freedom:

$$[K] = [H]^T [\tilde{K}] [H]. \quad (23)$$

3. Transform the element stiffness matrix from two to three degrees of freedom per point:

$$[K] = [r_{qs}]^T [K] [r_{qs}]. \quad (24)$$

4. The 3 by 3 partitions $[K_{pj}]$ of $[K]$ corresponding to the pivot point p are transformed:

$$[K_{pj}^3] = [T_p]^T [K_{pj}^3] [T_j], \quad j = 1, 2, 3, 4. \quad (25)$$

5. Finally, these 3 by 3 partitions are expanded to 6 by 6 partitions:

$$[K_{pj}] = \begin{bmatrix} K_{pj}^3 & 0 \\ 0 & 0 \end{bmatrix}. \quad (26)$$

THE TRAPRG ELEMENT

Table 2. Elements of the 8 by 8 Symmetric Stiffness Matrix for the TRAPRG Element.

	Col. 1	Col. 2	Col. 3	Col. 4
Row 1	$E_{22}I_{-10}$			
Row 2	$(E_{22}+E_{12})I_{00}$	$(E_{11}+2E_{12}+E_{22})I_{10}$		
Row 3	$E_{22}I_{-11}$	$(E_{12}+E_{22})I_{01}$	$E_{22}I_{-12}+E_{44}I_{10}$	
Row 4	$(E_{12}+E_{22})I_{01}$	$(E_{11}+2E_{12}+E_{22})I_{11}$	$(E_{12}+E_{22})I_{02}$ $+ E_{44}I_{20}$	$(E_{11}+2E_{12}+E_{22})I_{12}$ $+ E_{44}I_{30}$
Row 5	0	0	0	0
Row 6	0	0	$E_{44}I_{10}$	$E_{44}I_{20}$
Row 7	$E_{32}I_{00}$	$(E_{13}+E_{23})I_{10}$	$E_{23}I_{01}$	$(E_{13}+E_{23})I_{11}$
Row 8	$E_{32}I_{10}$	$(E_{31}+E_{32})I_{20}$	$(E_{23}+E_{44})I_{11}$	$(E_{13}+E_{23}+E_{44})I_{21}$

	Col. 5	Col. 6	Col. 7	Col. 8
Row 5	0			
Row 6	0	$E_{44}I_{10}$		
Row 7	0	0	$E_{33}I_{10}$	
Row 8	0	$E_{44}I_{11}$	$E_{33}I_{20}$	$E_{33}I_{30}+E_{44}I_{12}$

8.11.6 Mass Matrix Calculation (Subroutine MTRAPR of Module SMA2)

1. Form the coupled mass matrix in field coordinates:

$$[\bar{M}] = 2\pi \begin{bmatrix} M_1 I_{10} & & & & & & & & \\ M_1 I_{20} & M_1 I_{30} & & & & & & & \\ & & \text{Symmetric} & & & & & & \\ M_1 I_{11} & M_1 I_{21} & M_1 I_{12} & & & & & & \\ M_1 I_{21} & M_1 I_{31} & M_1 I_{22} & M_1 I_{32} & & & & & \\ 0 & 0 & 0 & 0 & M_2 I_{10} & & & & \\ 0 & 0 & 0 & 0 & M_2 I_{20} & M_2 I_{30} & & & \\ 0 & 0 & 0 & 0 & M_2 I_{11} & M_2 I_{21} & M_2 I_{12} & & \\ 0 & 0 & 0 & 0 & M_2 I_{21} & M_2 I_{31} & M_2 I_{22} & M_2 I_{32} & \end{bmatrix} \cdot \quad (27)$$

8x8

where

$$M_1 = M_2 = \rho \cdot \quad (28)$$

2. Transform the mass matrix to grid point degrees of freedom:

$$[\bar{M}] = [H]^T [\tilde{M}] [H]. \quad (29)$$

3. Transform the mass matrix from two to three degrees of freedom per point:

$$[M] = [r_{qs}] [\bar{M}] [r_{qs}]. \quad (30)$$

4. The 6 by 6 partitions,
- $[M_{pj}]$
- , are calculated as in Equations 25 and 26.

8.11.7 Thermal Load Calculations (Subroutine TTRAPR of Module SSG1)

1. Form the temperature gradient vector:

$$\{\Delta T\} = \{T\} - \begin{Bmatrix} T_o \\ T_o \\ T_o \\ T_o \end{Bmatrix}, \quad (31)$$

where $\{T\}$ is the vector of loading temperatures at the grid points.

2. Form the thermal expansion coefficient vector:

$$\{\alpha\} = \begin{Bmatrix} \alpha_r \\ \alpha_\theta \\ \alpha_z \\ 0 \end{Bmatrix}. \quad (32)$$

3. Multiply the elastic constants matrix by the thermal expansion coefficient vector:

$$\{AB\} = [E_g]\{\alpha\}. \quad (33)$$

4. Form the $[Q]$ matrix:

$$[Q] = \begin{bmatrix} AB_2 I_{00} & AB_2 I_{10} & AB_2 I_{01} & AB_2 I_{11} \\ (AB_1 + AB_2) I_{10} & (AB_1 + AB_2) I_{20} & (AB_1 + AB_2) I_{11} & (AB_1 + AB_2) I_{21} \\ AB_2 I_{01} & AB_2 I_{11} & AB_2 I_{02} & AB_2 I_{12} \\ (AB_1 + AB_2) I_{11} & (AB_1 + AB_2) I_{21} & (AB_1 + AB_2) I_{12} & (AB_1 + AB_2) I_{22} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ AB_3 I_{10} & AB_3 I_{20} & AB_3 I_{11} & AB_3 I_{21} \\ AB_3 I_{20} & AB_3 I_{30} & AB_3 I_{21} & AB_3 I_{31} \end{bmatrix} \quad 8 \times 4 \quad (34)$$

5. Partition the transformation matrix $[H]$ to form $[H']$:

$$[H'] = [h'_{ij}] \quad 4 \times 4, \quad (35)$$

where

$$h'_{ij} = h_{ik} \quad \text{for } i = 1, 2, 4; \quad j = 1, 2, \dots, 4; \quad \text{and } k = 2j-1$$

and h_{ik} are the elements of $[H]$.

6. Compute the thermal load in field coordinates:

$$\{\tilde{F}_T\} = 2\pi[Q][H']\{\Delta T\}. \quad (36)$$

7. Transform the thermal load to grid point degrees of freedom:

$$\{\bar{F}_T\} = [H]^T\{\tilde{F}_T\}. \quad (37)$$

8. Transform the thermal load from two to three degrees of freedom per point:

$$\{F_T\} = [\Gamma_{qs}]^T\{\bar{F}_T\}. \quad (38)$$

9. Each partition, $\{F_T^3\}$, of length 3 of $\{F_T\}$ is transformed to global coordinates by:

$$\{F_T^3\}_g = [T_i]^T\{F_T^3\}. \quad (39)$$

10. These vectors are added to the overall load vector, $\{P_g\}$.

8.11.8 Element Force and Stress Calculations (Subroutines STRAP1 and STRAP2 of Module SDR2).

Element stress and force data items are calculated in two phases. The first phase (subroutine STRAP1) calculates the element stiffness and stress matrices. The second phase (subroutine STRAP2) calculates the element forces and stresses from the various subcase displacement vectors.

Phase 1 calculations are as follows:

1. Form the element stiffness matrix, $[K]$, as in section 4.87.11.5.
2. Define a fifth "grid point" to be the average of the other four points:

$$R_{L5} = \frac{1}{4}(R_{L1} + R_{L2} + R_{L3} + R_{L4}). \quad (40)$$

$$Z_{L5} = \frac{1}{4}(Z_{L1} + Z_{L2} + Z_{L3} + Z_{L4}). \quad (41)$$

3. Form the matrices $[D^{(1)}], [D^{(2)}], [D^{(3)}], [D^{(4)}], [D^{(5)}]$

where

$$[D^{(i)}] = \begin{bmatrix} 0 & 1 & 0 & Z_{Li} & 0 & 0 & 0 & 0 \\ \frac{1}{R_{Li}} & 1 & \frac{Z_{Li}}{R_{Li}} & Z_{Li} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & R_{Li} \\ 0 & 0 & 1 & R_{Li} & 0 & 1 & 0 & Z_{Li} \end{bmatrix} \quad 4 \times 8 \quad (42)$$

where $i = 1$ to 5 denotes the five grid points.

4. Compute the stress matrices for each of the five grid points in field coordinates:

$$[\tilde{S}^{(i)}] = [E_g][D^{(i)}]. \quad (43)$$

STRUCTURAL ELEMENT DESCRIPTIONS

5. Transform each stress matrix to grid point degrees of freedom:

$$[\bar{S}^{(i)}] = [\tilde{S}^{(i)}][H]. \quad (44)$$

6. Transform each stress matrix from two to three degrees of freedom per point:

$$[S^{(i)}] = [\bar{S}^{(i)}][r_{qs}]. \quad (45)$$

7. Form the master stress matrix:

$$[S] = \begin{bmatrix} S^{(1)} \\ S^{(2)} \\ S^{(3)} \\ S^{(4)} \\ S^{(5)} \end{bmatrix} \quad 20 \times 12 \quad (46)$$

8. Transform the stress matrix from basic to local coordinates:

$$[S_l] = [S][T_{1234}], \quad (47)$$

where

$$[T_{1234}] = \begin{bmatrix} T_1 & 0 & 0 & 0 \\ 0 & T_2 & 0 & 0 \\ 0 & 0 & T_3 & 0 \\ 0 & 0 & 0 & T_4 \end{bmatrix}. \quad (48)$$

9. Compute the thermal stress vector:

$$\{T_s\} = [E_g]\{\alpha\}. \quad (49)$$

Phase 2 calculations are as follows:

1. Extract the displacement vector, $\{\Delta\}$, at the three translation coordinates of each of the four grid points from the global displacement vector.

THE TRAPRG ELEMENT

2. Calculate the element forces:

$$\{P\} = [K]\{\Delta\} . \quad (50)$$

3. Calculate the element stresses:

$$\{\sigma\} = [S_\ell]\{\Delta\} - T_d \{T_s\} , \quad (51)$$

where

$$T_d = \begin{cases} T_i - T_o, & \text{if } i \neq 5 \text{ (} T_i \text{ is the temperature at the } i^{\text{th}} \text{ point)} \\ T_{\text{avg}} - T_o, & \text{if } i = 5 \text{ (} T_{\text{avg}} \text{ is the average temperature over the four grid points)} \end{cases} \quad (52)$$

8.11.9 Thermal Analysis Calculations for the TRAPRG Element (Subroutine HRING by Module SMA1)

The calculations are described in the preceding description for the TRIARG element; see Section 8.10.9.

8.12 THE TØRDRG ELEMENT

8.12.1 Input Data for the TØRDRG Element

1. The ECPT/EST entries for the axisymmetric toroidal ring (TØRDRG) element are:

<u>Symbol</u>	<u>Description</u>
SIL_1, SIL_2	Scalar index numbers for the two grid points
α_1, α_2	Angles of curvature at the two grid points (degrees)
γ	Material property orientation angle (currently not used)
H_m, H_f	Membrane and flexure thickness
$\left. \begin{matrix} N_1, X_1, Y_1, Z_1 \\ N_2, X_2, Y_2, Z_2 \end{matrix} \right\}$	Local coordinate system number and location in basic coordinates of the grid points.
t_μ	Element temperature for material properties

For this element Y_i must equal zero for $i = 1$ and 2 , and we define:

$$\{R\} = \begin{Bmatrix} X_1 \\ X_2 \end{Bmatrix} \quad , \quad (1)$$

$$\{Z\} = \begin{Bmatrix} Z_1 \\ Z_2 \end{Bmatrix} \quad . \quad (2)$$

2. Coordinate system data

The location (X_i, Y_i, Z_i) and local coordinate system number (N_i) of each grid point are used to calculate the 3 by 3 global-to-basic coordinate system transformation matrices, $[T_i]$, $i = 1, 2$.

3. Material data

The material property input for the TØRDRG element is the same as that for TRIARG element (see section 4.87.10.1) with the following notational changes: ,

STRUCTURAL ELEMENT DESCRIPTIONS

$$E_p = E_r, \quad E_T = E_\theta, \quad \nu_{PT} = \nu_{r\theta}$$

$$\{ALF\} = \begin{Bmatrix} \alpha_r \\ \alpha_\theta \end{Bmatrix}$$

8.12.2 General Calculations

1. Compute the following constants used in stiffness matrix generation:

$$C = E_p/E_T, \quad (3)$$

$$D_M = E_p H_m / (C - \nu_{PT}^2), \quad (4)$$

$$D_B = E_p H_f^3 / (12 (C - \nu_{PT}^2)). \quad (5)$$

2. Determine if the element is a toroidal ring, a conical ring, a cylindrical ring, or a shell cap:

- (1) if $\alpha_1 \neq \alpha_2$, then the element is a toroidal ring.
- (2) if $\alpha_1 = \alpha_2 = 90^\circ$, then the element is a cylindrical ring.
- (3) if $\alpha_1 = \alpha_2 \neq 90^\circ$, then the element is a conical ring.
- (4) if $\alpha_1 = 0$, then the element is a shell cap.

3. Compute the local coordinate constants for a toroidal element:

$$\phi_B = \alpha_2 - \alpha_1, \quad (6)$$

$$R_p = \frac{[(R_2 - R_1)^2 + (Z_2 - Z_1)^2]^{1/2}}{2 \sin(\frac{\phi_B}{2})}, \quad (7)$$

$$\lambda_1 = \frac{1}{R_p}, \quad (8)$$

THE TØRDRG ELEMENT

$$S = (\alpha_2 - \alpha_1)R_p, \quad (9)$$

$$B_\beta = R_1 + R_p [\sin(\alpha_1 + \frac{\phi_\beta}{2}) - \sin(\alpha_1)], \quad (10)$$

$$R_T = \frac{B_\beta}{\sin(\alpha_1 + \frac{\phi_\beta}{2})}, \quad (11)$$

$$\psi_1 = \cos(\alpha_1 + \frac{\phi_\beta}{2}), \quad (12)$$

$$\psi_2 = -\frac{\sin(\alpha_1 + \frac{\phi_\beta}{2})}{R_p}. \quad (13)$$

4. Compute the local coordinate constants for a conical or cylindrical ring

$$S = [(R_2 - R_1)^2 + (Z_2 - Z_1)^2]^{1/2}, \quad (14)$$

$$B_\beta = R_1 + \frac{S \cos \alpha_1}{2}, \quad (15)$$

$$R_T = \frac{B_\beta}{\sin \alpha_1}, \quad (16)$$

$$R_p = 0, \quad (17)$$

$$\lambda_1 = 0, \quad (18)$$

$$\psi_1 = \cos \alpha_1, \quad (19)$$

$$\psi_2 = 0. \quad (20)$$

STRUCTURAL ELEMENT DESCRIPTIONS

5. Generate the transformation matrix from field coordinates to grid point degrees of freedom:

$$[r_{Bq}] = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{-10}{s^3} & \frac{-6}{s^2} & 0 & \frac{-3}{2s} & 0 & 0 & \frac{10}{s^3} & \frac{-4}{s^2} & 0 & \frac{1}{2s} \\ 0 & 0 & \frac{15}{s^4} & \frac{8}{s^3} & 0 & \frac{3}{2s^2} & 0 & 0 & \frac{-15}{s^4} & \frac{7}{s^3} & 0 & \frac{-1}{s^2} \\ 0 & 0 & \frac{-6}{s^5} & \frac{-3}{s^4} & 0 & \frac{-1}{2s^3} & 0 & 0 & \frac{6}{s^5} & \frac{-3}{s^4} & 0 & \frac{1}{2s^3} \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{-3}{s^2} & 0 & 0 & 0 & \frac{-2}{s} & 0 & \frac{3}{s^2} & 0 & 0 & 0 & \frac{-1}{s} & 0 \\ \frac{2}{s^3} & 0 & 0 & 0 & \frac{1}{s^2} & 0 & \frac{-2}{s^3} & 0 & 0 & 0 & \frac{1}{s^2} & 0 \end{bmatrix} \quad (21)$$

10x12

6. Generate the transformation matrix from local to system coordinates:

$$[r_{rs}] = \begin{bmatrix} \cos \alpha_1 & 0 & -\sin \alpha_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \sin \alpha_1 & 0 & \cos \alpha_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \cos \alpha_2 & 0 & -\sin \alpha_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \sin \alpha_2 & 0 & \cos \alpha_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad 12 \times 12 \quad (22)$$

7. Rearrange the transformation matrix $[r_{\beta q}]$ such that the membrane and flexure terms are reversed.

$$[\bar{r}_{\beta q}] = \begin{bmatrix} r_{\beta q}^{(2)} \\ r_{\beta q}^{(1)} \end{bmatrix} \quad 10 \times 12 \quad (23)$$

where $[r_{\beta q}^{(1)}]$ is the first six rows of $[r_{\beta q}]$ and $[r_{\beta q}^{(2)}]$ is the last four rows of $[r_{\beta q}]$.

8.12.3 Integral Calculations

The method used to compute the integrals depends on the geometric shape of the element.

1. Toroidal ring - basic integrals.

There are six basic integrals to be computed, of which the first three can best be evaluated by series expansion, but the last three require numerical integration.

$$I_1^j = R_p^{j+1} \int_0^{\phi_\beta} \phi^j d\phi = \frac{R_p^{j+1} \phi_\beta^{j+1}}{j+1}, \quad (24)$$

$$I_2^j = R_p^{j+1} \int_0^{\phi_\beta} \phi^j \sin\phi d\phi = R_p^{j+1} \sum_{i=0}^{\infty} \frac{(-1)^i \phi_\beta^{j+1+2i+1}}{(j+1+2i+1)(2i+1)!}, \quad (25)$$

$$I_3^j = R_p^{j+1} \int_0^{\phi_\beta} \phi^j \cos\phi d\phi = R_p^{j+1} \sum_{i=0}^{\infty} \frac{(-1)^i \phi_\beta^{j+1+2i}}{(j+1+2i)(2i)!}, \quad (26)$$

$$I_4^j = R_p^{j+1} \int_0^{\phi_\beta} \frac{\phi^j \sin^2\phi}{B} d\phi, \quad (27)$$

$$I_5^j = R_p^{j+1} \int_0^{\phi_\beta} \frac{\phi^j 2\sin\phi \cos\phi}{B} d\phi, \quad (28)$$

$$I_6^j = R_p^{j+1} \int_0^{\phi_\beta} \frac{\phi^j \cos^2\phi}{B} d\phi, \quad (29)$$

where

$$B = R_1 - R_p \sin\alpha_1 + R_p \sin\alpha_1 \cos\phi + R_p \cos\alpha_1 \sin\phi, \quad (30)$$

and $j = 0, 1, \dots, 10$.

The summations in I_2^j and I_3^j are carried out until the truncation error is insignificant.

2. Toroidal ring - required integrals

The actual integrals required can now be defined in terms of basic integrals.

$$\delta_1^j = (R_1 - R_p \sin \alpha_1) I_1^j + R_p \cos \alpha_1 I_2^j + R_p \sin \alpha_1 I_3^j, \quad (31)$$

$$\delta_2^j = \cos \alpha_1 I_2^j + \sin \alpha_1 I_3^j, \quad (32)$$

$$\delta_3^j = \cos^2 \alpha_1 I_4^j + \sin \alpha_1 \cos \alpha_1 I_5^j + \sin^2 \alpha_1 I_6^j, \quad (33)$$

$$\delta_4^j = \cos \alpha_1 I_3^j - \sin \alpha_1 I_2^j, \quad (34)$$

$$\delta_5^j = \sin \alpha_1 \cos \alpha_1 (I_6^j - I_4^j) + \frac{1}{2}(1 - 2\sin^2 \alpha_1) I_5^j, \quad (35)$$

$$\delta_6^j = \cos^2 \alpha_1 I_6^j - \sin \alpha_1 \cos \alpha_1 I_5^j + \sin^2 \alpha_1 I_4^j, \quad (36)$$

3. Conic ring - basic integrals

$$I_1^j = \int_0^S \xi^j d\xi = \frac{S^{j+1}}{j+1}, \quad j = 0, 1, \dots, 10 \quad (37)$$

$$I_2^j = \int_0^S \left(\frac{\xi^j}{R_1 + \xi \cos \alpha_1} \right) d\xi, \quad j = 0, 1, \dots, 10 \quad (38)$$

$$I_2^0 = \frac{1}{\cos \alpha_1} \ln \left(\frac{R_1 + S \cos \alpha_1}{R_1} \right), \quad (39)$$

$$I_2^1 = \frac{1}{\cos \alpha_1} \left[S - \frac{R_1}{\cos \alpha_1} \ln \left(\frac{R_1 + S \cos \alpha_1}{R_1} \right) \right], \quad (40)$$

$$I_2^j = \frac{S^{j+1}}{R_1} \sum_{i=0}^{\infty} \frac{(-1)^i \left[\frac{S \cos \alpha_1}{R_1} \right]^i}{(j+1+i)}, \quad j = 2, 3, \dots, 10. \quad (41)$$

4. Conic ring - required integrals

$$\delta_1^j = R_1 I_1^j + \psi_1 I_1^{j+1}, \quad (42)$$

$$\delta_2^j = \sin \alpha_1 I_1^j, \quad (43)$$

$$\delta_3^j = \sin^2 \alpha_1 I_2^j, \quad (44)$$

$$\delta_4^j = \psi_1 I_1^j, \quad j = 0, 1, \dots, 10 \quad (45)$$

$$\delta_5^j = \sin \alpha_1 \psi_1 I_2^j, \quad (46)$$

$$\delta_6^j = \psi_1^2 I_2^j, \quad (47)$$

5. Cylindrical ring - basic integrals

$$I_1^j = \int_0^S \xi^j d\xi = \frac{S^{j+1}}{j+1}, \quad j = 0, 1, \dots, 10 \quad (48)$$

$$I_2^j = \int_0^S \frac{\xi^j}{R_1} d\xi = \frac{1}{R_1} \left(\frac{S^{j+1}}{j+1} \right), \quad j = 0, 1, \dots, 10 \quad (49)$$

6. Cylindrical ring - required integrals

$$\delta_1^j = R_1 I_1^j + \psi_1 I_1^{j+1}, \quad (50)$$

$$\delta_2^j = \sin \alpha_1 I_1^j, \quad (51)$$

$$\delta_3^j = \sin^2 \alpha_1 I_2^j, \quad (52)$$

$$\delta_4^j = \delta_5^j = \delta_6^j = 0, \quad (53)$$

8.12.4 Elastic Constants Matrix Calculations

Form elastic constants matrix

$$[E] = \frac{1}{\left(1 - \frac{E_T}{E_P} \nu_{PT}^2\right)} \begin{bmatrix} E_P & E_T \nu_{PT} \\ E_T \nu_{PT} & E_T \end{bmatrix} \cdot 2 \times 2 \quad (54)$$

8.12.5 Stiffness Matrix Calculations (Subroutine KTØRDR of Module SMA1)

1. Define the constants

$$A = R_P, \quad (55)$$

$$V = \nu_{PT}, \quad (56)$$

$$C = E_P/E_T. \quad (57)$$

2. Form the stiffness matrix terms in field coordinates as shown in Tables 3, 4 and 5.
3. Transform the stiffness matrix from field coordinates to grid point degrees of freedom:

$$[\bar{K}] = [r_{\beta q}]^T [\tilde{K}] [r_{\beta q}]. \quad (58)$$

4. Transform the stiffness matrix from local to system coordinates:

$$[K] = [r_{rs}]^T [\bar{K}] [r_{rs}]. \quad (59)$$

5. The global-to-basic coordinate system transformation matrices
- $[T_i]$
- are expanded to 6 by 6 matrices:

$$[T_i^6] = \begin{bmatrix} T_i & 0 \\ 0 & I \end{bmatrix}, \quad i = 1, 2. \quad (60)$$

STRUCTURAL ELEMENT DESCRIPTIONS

Table 3. Columns 1, 2 and 3 of the Symmetric 10 by 10 Stiffness Matrix for the TØRDRG Element.

	Column 1	Column 2	Column 3
Row 1	$D_M(\frac{C}{A^2} \delta_1^0 + \frac{2V}{A} \delta_2^0 + \delta_3^0)$		
Row 2	$D_M(\frac{C}{A^2} \delta_1^1 + \frac{2V}{A} \delta_2^1 + \delta_3^1)$	$D_B \delta_6^0 + D_M(\frac{C}{A^2} \delta_1^2 + \frac{2V}{A} \delta_2^2 + \delta_3^2)$	
Row 3	$D_M(\frac{C}{A^2} \delta_1^2 + \frac{2V}{A} \delta_2^2 + \delta_3^2)$	$D_B(2V\delta_4^0 + 2\delta_6^1) + D_M(\frac{C}{A^2} \delta_1^3 + \frac{2V}{A} \delta_2^3 + \delta_3^3)$	$D_B 4(C\delta_1^0 + 2V\delta_4^1 + \delta_6^2) + D_M(\frac{C}{A^2} \delta_1^4 + \frac{2V}{A} \delta_2^4 + \delta_3^4)$
Row 4	$D_M(\frac{C}{A^2} \delta_1^3 + \frac{2V}{A} \delta_2^3 + \delta_3^3)$	$D_B(6V\delta_4^1 + 3\delta_6^2) + D_M(\frac{C}{A^2} \delta_1^4 + \frac{2V}{A} \delta_2^4 + \delta_3^4)$	$D_B 6(2C\delta_1^1 + 3V\delta_4^2 + \delta_6^3) + D_M(\frac{C}{A^2} \delta_1^5 + \frac{2V}{A} \delta_2^5 + \delta_3^5)$
Row 5	$D_M(\frac{C}{A^2} \delta_1^4 + \frac{2V}{A} \delta_2^4 + \delta_3^4)$	$D_B(12V\delta_4^2 + 4\delta_6^3) + D_M(\frac{C}{A^2} \delta_1^5 + \frac{2V}{A} \delta_2^5 + \delta_3^5)$	$D_B 2(12C\delta_1^2 + 16V\delta_4^3 + 4\delta_6^4) + D_M(\frac{C}{A^2} \delta_1^6 + \frac{2V}{A} \delta_2^6 + \delta_3^6)$
Row 6	$D_M(\frac{C}{A^2} \delta_1^5 + \frac{2V}{A} \delta_2^5 + \delta_3^5)$	$D_B(20V\delta_4^3 + 5\delta_6^4) + D_M(\frac{C}{A^2} \delta_1^6 + \frac{2V}{A} \delta_2^6 + \delta_3^6)$	$D_B 10(4C\delta_1^3 + 5V\delta_4^4 + \delta_6^5) + D_M(\frac{C}{A^2} \delta_1^7 + \frac{2V}{A} \delta_2^7 + \delta_3^7)$
Row 7	$D_M(\frac{V}{A} \delta_4^0 + \delta_5^0)$	$D_M(\frac{V}{A} \delta_4^1 + \delta_5^1)$	$D_M(\frac{V}{A} \delta_4^2 + \delta_5^2)$
Row 8	$D_M(\frac{C}{A} \delta_1^0 + \frac{V}{A} \delta_4^1 + V\delta_2^0 + \delta_5^1)$	$D_M(\frac{C}{A} \delta_1^1 + \frac{V}{A} \delta_4^2 + V\delta_2^1 + \delta_5^2)$	$D_M(\frac{C}{A} \delta_1^2 + \frac{V}{A} \delta_4^3 + V\delta_2^2 + \delta_5^3)$
Row 9	$D_M(\frac{2C}{A} \delta_1^1 + \frac{V}{A} \delta_4^2 + 2V\delta_2^1 + \delta_5^2)$	$D_M(2\frac{C}{A} \delta_1^2 + \frac{V}{A} \delta_4^3 + 2V\delta_2^2 + \delta_5^3)$	$D_M(2\frac{C}{A} \delta_1^3 + \frac{V}{A} \delta_4^4 + 2V\delta_2^3 + \delta_5^4)$
Row 10	$D_M(\frac{3C}{A} \delta_1^2 + \frac{V}{A} \delta_4^3 + 3V\delta_2^2 + \delta_5^3)$	$D_M(\frac{3C}{A} \delta_1^3 + \frac{V}{A} \delta_4^4 + 3V\delta_2^3 + \delta_5^4)$	$D_M(\frac{3C}{A} \delta_1^4 + \frac{V}{A} \delta_4^5 + 3V\delta_2^4 + \delta_5^5)$

THE TØRDRG ELEMENT

Table 4. Columns 4, 5, and 6 of the Symmetric 10 by 10 Stiffness Matrix for the TØRDRG Element.

	Column 4	Column 5	Column 6
Row 4	$D_B 9(4C\delta_1^2 + 4V\delta_4^3 + \delta_6^4)$ $+ D_M(\frac{C}{A^2} \delta_1^6 + \frac{2V}{A} \delta_2^6 + \delta_3^6)$		
Row 5	$D_B 12(6C\delta_1^3 + 5V\delta_4^4 + \delta_6^5)$ $+ D_M(\frac{C}{A^2} \delta_1^7 + \frac{2V}{A} \delta_2^7 + \delta_3^7)$	$D_B 16(9C\delta_1^4 + 6V\delta_4^5 + \delta_6^6)$ $+ D_M(\frac{C}{A^2} \delta_1^8 + \frac{2V}{A} \delta_2^8 + \delta_3^8)$	
Row 6	$D_B 15(8C\delta_1^4 + 6V\delta_4^5 + \delta_6^6)$ $+ D_M(\frac{C}{A^2} \delta_1^8 + \frac{2V}{A} \delta_2^8 + \delta_3^8)$	$D_B 20(12C\delta_1^5 + 7V\delta_4^6 + \delta_6^7)$ $+ D_M(\frac{C}{A^2} \delta_1^9 + \frac{2V}{A} \delta_2^9 + \delta_3^9)$	$D_B 25(16C\delta_1^6 + 8V\delta_4^7 + \delta_6^8)$ $+ D_M(\frac{C}{A^2} \delta_1^{10} + \frac{2V}{A} \delta_2^{10} + \delta_3^{10})$
Row 7	$D_M(\frac{V}{A} \delta_4^3 + \delta_5^3)$	$D_M(\frac{V}{A} \delta_4^4 + \delta_5^4)$	$D_M(\frac{V}{A} \delta_4^5 + \delta_5^5)$
Row 8	$D_M(\frac{C}{A} \delta_1^3 + \frac{V}{A} \delta_4^4)$ $+ V\delta_2^3 + \delta_5^4)$	$D_M(\frac{C}{A} \delta_1^4 + \frac{V}{A} \delta_4^5)$ $+ V\delta_2^4 + \delta_5^5)$	$D_M(\frac{C}{A} \delta_1^5 + \frac{V}{A} \delta_4^6)$ $+ V\delta_2^5 + \delta_5^6)$
Row 9	$D_M(\frac{2C}{A} \delta_1^4 + \frac{V}{A} \delta_4^5)$ $+ 2V\delta_2^4 + \delta_5^5)$	$D_M(\frac{2C}{A} \delta_1^5 + \frac{V}{A} \delta_4^6)$ $+ 2V\delta_2^5 + \delta_5^6)$	$D_M(\frac{2C}{A} \delta_1^6 + \frac{V}{A} \delta_4^7)$ $+ 2V\delta_2^6 + \delta_5^7)$
Row 10	$D_M(\frac{3C}{A} \delta_1^5 + \frac{V}{A} \delta_4^5)$ $+ 3V\delta_2^5 + \delta_5^5)$	$D_M(\frac{3C}{A} \delta_1^6 + \frac{V}{A} \delta_4^7)$ $+ 3V\delta_2^6 + \delta_5^7)$	$D_M(\frac{3C}{A} \delta_1^7 + \frac{V}{A} \delta_4^8)$ $+ 3V\delta_2^7 + \delta_5^8)$

STRUCTURAL ELEMENT DESCRIPTIONS

Table 5. Columns 7, 8, 9, and 10 of the Symmetric 10 by 10 Stiffness Matrix for the TØRDRG Element.

	Col. 7	Col. 8	Col. 9	Col. 10
Row 7	$D_M \delta_6^0$			
Row 8	$D_M(V\delta_4^0 + \delta_6^1)$	$D_M(C\delta_1^0 + 2V\delta_4^1 + \delta_6^2)$		
Row 9	$D_M(2V\delta_4^1 + \delta_6^2)$	$D_M(2C\delta_1^1 + 3V\delta_4^2 + \delta_6^3)$	$D_M(4C\delta_1^2 + 4V\delta_4^3 + \delta_6^4)$	
Row 10	$D_M(3V\delta_4^2 + \delta_6^3)$	$D_M(3C\delta_1^2 + 4V\delta_4^3 + \delta_6^4)$	$D_M(6C\delta_1^3 + 5V\delta_4^4 + \delta_6^5)$	$D_M(9C\delta_1^4 + 6V\delta_4^5 + \delta_6^6)$

6. The stiffness matrix, $[K]$, is partitioned into 6 by 6 matrices:

$$[K] \Rightarrow \begin{bmatrix} K_{11}^6 & K_{12}^6 \\ K_{21}^6 & K_{22}^6 \end{bmatrix} . \quad (61)$$

7. These 6x6 partitions are transformed to local coordinates:

$$[K_{pj}^6]_L = [T_p^6]^T [K_{pj}^6] [T_j^6], \quad (62)$$

where $j = 1, 2$, and p is the pivot point, $p = 1$ or 2 .

STRUCTURAL ELEMENT DESCRIPTIONS

8.12.6 Mass Matrix Calculations (Subroutine MTORDR of Module SMA2)

1. Form the mass matrix in field coordinates:

$$[\tilde{M}] = 2\pi\rho H_m \begin{bmatrix} \delta_1^0 & & & & & & & & & \\ \delta_1^1 & \delta_1^2 & & & & & & & & \\ \delta_1^2 & \delta_1^3 & \delta_1^4 & & & & & & & \\ \delta_1^3 & \delta_1^4 & \delta_1^5 & \delta_1^6 & \text{Symmetric} & & & & & \\ 0 & 0 & 0 & 0 & \delta_1^0 & & & & & \\ 0 & 0 & 0 & 0 & \delta_1^1 & \delta_1^2 & & & & \\ 0 & 0 & 0 & 0 & \delta_1^2 & \delta_1^3 & \delta_1^4 & & & \\ 0 & 0 & 0 & 0 & \delta_1^3 & \delta_1^4 & \delta_1^5 & \delta_1^6 & & \\ 0 & 0 & 0 & 0 & \delta_1^4 & \delta_1^5 & \delta_1^6 & \delta_1^7 & \delta_1^8 & \\ 0 & 0 & 0 & 0 & \delta_1^5 & \delta_1^6 & \delta_1^7 & \delta_1^8 & \delta_1^9 & \delta_1^{10} \end{bmatrix} \quad (63)$$

10x10

2. Transform the mass matrix to grid point degrees of freedom:

$$[\tilde{M}] = [\tilde{r}_{\beta q}]^T [\tilde{M}] [\tilde{r}_{\beta q}] \quad (64)$$

3. Transform the mass matrix from local to system coordinates:

$$[M] = [r_{rs}]^T [\tilde{M}] [r_{rs}] \quad (65)$$

4. The mass matrix, $[M]$, is partitioned into 6x6 matrices, and these 6x6 partitions are transformed to local coordinates (see Equations 61 and 62).

8.12.7 Thermal Load Calculations (Subroutine TTØRDR of Module SSG1)

1. Compute the temperature gradient constants:

$$\Delta T_1^{(m)} = T_1^{(m)} - T_0 , \quad (66)$$

$$\Delta T_2^{(m)} = T_2^{(m)} - T_1^{(m)} , \quad (67)$$

$$\Delta T_1^{(f)} = 0 , \quad (68)$$

$$\Delta T_2^{(f)} = 0 , \quad (69)$$

where $T_i^{(m)}$ are the membrane temperatures at point i .

2. Compute the thermal strain vectors:

$$\{\epsilon_T^{(0)}\} = \Delta T_1^{(m)} \{ALF\} , \quad (70)$$

$$\{\epsilon_T^{(1)}\} = \Delta T_2^{(m)} \{ALF\} , \quad (71)$$

$$\{H_T^{(0)}\} = \Delta T_1^{(f)} \{ALF\} , \quad (72)$$

$$\{H_T^{(1)}\} = \Delta T_2^{(f)} \{ALF\} , \quad (73)$$

where $\{ALF\}$ is a vector of length two; the first component is α_r and the second α_θ .

3. Form the matrices of integrals

$$[\tilde{F}_{ME}^{(0)}] = 2\pi H_m \begin{bmatrix} 0 & \delta_1^0 & 2\delta_1^1 & 3\delta_1^2 & \lambda_1 \delta_1^0 & \lambda_1 \delta_1^1 & \lambda_1 \delta_1^2 & \lambda_1 \delta_1^3 & \lambda_1 \delta_1^4 & \lambda_1 \delta_1^5 \\ \delta_4^0 & \delta_4^1 & \delta_4^2 & \delta_4^3 & \delta_2^0 & \delta_2^1 & \delta_2^2 & \delta_2^3 & \delta_2^4 & \delta_2^5 \end{bmatrix}_{2 \times 10} \quad (74)$$

STRUCTURAL ELEMENT DESCRIPTIONS

$$[\tilde{F}_{ME}^{(1)}] = \frac{2\pi H_m}{S} \begin{bmatrix} 0 & \delta_1^1 & 2\delta_1^2 & 3\delta_1^3 & \lambda_1 \delta_1^1 & \lambda_1 \delta_1^2 & \lambda_1 \delta_1^3 & \lambda_1 \delta_1^4 & \lambda_1 \delta_1^5 & \lambda_1 \delta_1^6 \\ \delta_4^1 & \delta_4^2 & \delta_4^3 & \delta_4^4 & \delta_2^1 & \delta_2^2 & \delta_2^3 & \delta_2^4 & \delta_2^5 & \delta_2^6 \end{bmatrix} \quad 2 \times 10 \quad (75)$$

$$[\tilde{F}_{FE}^{(0)}] = \frac{2\pi H_f^3}{12} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & -2\delta_1^0 & -6\delta_1^1 & -12\delta_1^2 & -20\delta_1^3 \\ 0 & 0 & 0 & 0 & 0 & -\delta_4^0 & -2\delta_4^1 & -3\delta_4^2 & -4\delta_4^3 & -5\delta_4^4 \end{bmatrix} \quad 2 \times 10 \quad (76)$$

$$[\tilde{F}_{FE}^{(1)}] = \frac{2\pi H_f^3}{12S} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & -2\delta_1^1 & -6\delta_1^2 & -12\delta_1^3 & -20\delta_1^4 \\ 0 & 0 & 0 & 0 & 0 & -\delta_4^1 & -2\delta_4^2 & -3\delta_4^3 & -4\delta_4^4 & -5\delta_4^5 \end{bmatrix} \quad 2 \times 10 \quad (77)$$

4. Compute the thermal load vector in field coordinates

$$\begin{aligned} \{\tilde{F}_T\} &= [\tilde{F}_{ME}^{(0)}]^T [E] \{\epsilon_T^{(0)}\} + [\tilde{F}_{ME}^{(1)}]^T [E] \{\epsilon_T^{(1)}\} \\ &+ [\tilde{F}_{FE}^{(0)}]^T [E] \{H_T^{(0)}\} + [\tilde{F}_{FE}^{(1)}]^T [E] \{H_T^{(0)}\} . \end{aligned} \quad (78)$$

5. Transform the thermal load vector to grid point degrees of freedom:

$$\{\bar{F}_T\} = [r_{Bq}]^T \{\tilde{F}_T\} . \quad (79)$$

6. Transform the thermal load vector from local to system coordinates:

$$\{F_T\} = [r_{rs}]^T \{\bar{F}_T\} . \quad (80)$$

7. Transform the thermal load vector to basic coordinates:

$$\{F_T\}_b = [T_{12}]^T \{F_T\} , \quad (81)$$

where

$$[T_{12}] = \begin{bmatrix} T_1 & 0 & 0 & 0 \\ 0 & I & 0 & 0 \\ 0 & 0 & T_2 & 0 \\ 0 & 0 & 0 & I \end{bmatrix} \quad (82)$$

and $[T_1]$ and $[I]$ are 3 by 3 matrices.

8. These vectors are added to the overall load vector, $\{P_g\}$.

8.12.8 Element Force and Stress Calculations (Subroutines STØRD1 and STØRD2 of Module SDR1)

Element stress and force data are calculated in two phases. The first phase (subroutine STØRD1) calculates the element stiffness and stress matrices. The second phase (subroutine STØRD2) calculates the element forces and stresses from the various subcase displacement vectors. Stresses are evaluated at both ends and at the mid-span of the element.

Phase 1 calculations are as follows:

1. Form the element stiffness matrix, $[K]$, as in Section 8.12.5.
2. Set up the coordinates of the three stress points:

$$\{X\} = \begin{Bmatrix} 0 \\ \frac{S}{2} \\ S \end{Bmatrix} \quad (83)$$

3. Compute the constants $\lambda_2, \lambda_3, \lambda_4$ as a function of the stress points coordinates.
 - a. If the element is a toroidal ring, then:

$$\lambda_2^{(i)} = \frac{\cos \left(\alpha_1 + \frac{X(i)}{R_p} \right)}{R_1 - R_p \left[\sin \alpha_1 - \sin \left(\alpha_1 + \frac{X(i)}{R_p} \right) \right]} \quad (84)$$

STRUCTURAL ELEMENT DESCRIPTIONS

$$\lambda_3^{(i)} = \frac{\sin \left(\alpha_1 + \frac{X(i)}{R_p} \right)}{R_1 - R_p \left[\sin \alpha_1 - \sin \left(\alpha_1 + \frac{X(i)}{R_p} \right) \right]} \quad i = 1, 2, 3, \quad (85)$$

$$\lambda_4^{(i)} = -\lambda_3^{(i)} / R_p . \quad (86)$$

b. If the element is a cylindrical or conical ring, then:

$$\lambda_2^{(i)} = \frac{\cos \alpha_1}{R_1 + X(i) \cos \alpha_1} , \quad (87)$$

$$\lambda_3^{(i)} = \frac{\cos \alpha_1}{R_1 + X(i) \cos \alpha_1} \quad i = 1, 2, 3, \quad (88)$$

$$\lambda_4^{(i)} = 0 . \quad (89)$$

c. If the element is a shell cap, then:

$$\lambda_2^{(i)} = \frac{\cos \left(\alpha_1 + \frac{X(i)}{R_p} \right)}{R_1 - R_p \left[\sin \alpha_1 - \sin \left(\alpha_1 + \frac{X(i)}{R_p} \right) \right]} \quad i = 1, 2, 3 , \quad (90)$$

$$\lambda_3^{(i)} = 1/R_p , \quad (91)$$

$$\lambda_4^{(i)} = -1/(R_p^2) . \quad (92)$$

4. Compute the stress matrix in field coordinates for the three stress points as shown in Tables 6 and 7. Note that the factors H and $H^3/12$ have been omitted for $[\tilde{S}_1^{(i)}]$ and $[\tilde{S}_2^{(i)}]$ respectively.

If the element is a shell cap, then modify $[\tilde{S}^{(i)}]$ by:

$$\tilde{S}_{12} = E_{12} + E_{11} , \quad (93)$$

THE TORDRG ELEMENT

$$\tilde{S}_{22} = E_{22} + E_{12} , \quad (94)$$

$$\tilde{S}_{37} = -2 (E_{12} + E_{11}) , \quad (95)$$

$$\tilde{S}_{47} = 2 (E_{22} + E_{12}) , \quad (96)$$

$$\tilde{S}_{58} = 3(E_{22} - 4 E_{11}) . \quad (97)$$

5. Form the master stress matrix in field coordinates:

$$[\tilde{S}] = \begin{bmatrix} \tilde{S}^{(1)} \\ \tilde{S}^{(2)} \\ \tilde{S}^{(3)} \end{bmatrix} \quad 15 \times 10 . \quad (98)$$

STRUCTURAL ELEMENT DESCRIPTIONS

Table 6. Terms in Columns 1 Through 6 of the 2 by 10 $[\tilde{S}_1^{(i)}]$ Matrix and the 3 by 10 $[\tilde{S}_2^{(i)}]$ Matrix.

The $[\tilde{S}_1^{(i)}]$ Matrix

Col. 1	Col. 2	Col. 3	Col. 4	Col. 5	Col. 6
$\lambda_2^{(i)} E_{12}$	E_{11}	$2E_{11}X(i)$	$3E_{11}X^2(i)$	$\lambda_1^{(i)} E_{11}$	$(\lambda_1^{(i)} E_{11} + \lambda_3^{(i)} E_{12})X(i)$
	$+ \lambda_2^{(i)} E_{12}X(i)$	$+ \lambda_2^{(i)} E_{12}X^2(i)$	$+ \lambda_2^{(i)} E_{12}X^3(i)$	$+ \lambda_3^{(i)} E_{12}$	
$\lambda_2^{(i)} E_{22}$	E_{12}	$2E_{12}X^2(i)$	$3E_{12}X^2(i)$	$\lambda_1^{(i)} E_{12}$	$(\lambda_1^{(i)} E_{12} + \lambda_3^{(i)} E_{22})X(i)$
	$+ \lambda_2^{(i)} E_{22}X(i)$	$+ \lambda_2^{(i)} E_{22}X^2(i)$	$+ \lambda_2^{(i)} E_{22}X^3(i)$	$+ \lambda_3^{(i)} E_{22}$	

The $[\tilde{S}_2^{(i)}]$ Matrix

0	0	0	0	0	$-\lambda_2^{(i)} E_{12}$
0	0	0	0	0	$\lambda_2^{(i)} E_{22}$
0	0	0	0	0	$g(i)$

where $g(i) = (\lambda_2^{(i)})^2 E_{22} - \lambda_4^{(i)} E_{12}$ and

$$[\tilde{S}^{(i)}] = \begin{bmatrix} \tilde{S}_1^{(i)} \\ \tilde{S}_2^{(i)} \end{bmatrix}_{5 \times 10}$$

THE TORDRG ELEMENT

Table 7. Terms in Columns 7 Through 10 of the 2 by 10 $[\tilde{S}_1^{(i)}]$ Matrix and the 3 by 10 $[\tilde{S}_2^{(i)}]$ Matrix.

The $[\tilde{S}_1^{(i)}]$ Matrix

	Col. 1	Col. 2	Col. 3	Col. 4
Row 1	$(\lambda_1^{(i)} E_{11} + \lambda_3^{(i)} E_{12} X^2(i))$	$(\lambda_1^{(i)} E_{11} + \lambda_3^{(i)} E_{12} X^3(i))$	$(\lambda_1^{(i)} E_{11} + \lambda_3^{(i)} E_{12} X^4(i))$	$(\lambda_1^{(i)} E_{11} + \lambda_3^{(i)} E_{12} X^5(i))$
Row 2	$(\lambda_1^{(i)} E_{12} + \lambda_3^{(i)} E_{22} X^2(i))$	$(\lambda_1^{(i)} E_{12} + \lambda_3^{(i)} E_{22} X^3(i))$	$(\lambda_1^{(i)} E_{12} + \lambda_3^{(i)} E_{22} X^4(i))$	$(\lambda_1^{(i)} E_{12} + \lambda_3^{(i)} E_{22} X^5(i))$

The $[\tilde{S}_2^{(i)}]$ Matrix

Row 1	$-2\lambda_2^{(i)} E_{12} X(i) - 2E_{11}$	$-3\lambda_2^{(i)} E_{12} X^2(i) - 6E_{11} X(i)$	$-4\lambda_2^{(i)} E_{12} X^3(i) - 12E_{11} X^2(i)$	$-5\lambda_2^{(i)} E_{12} X^4(i) - 20E_{11} X^3(i)$
Row 2	$2\lambda_2^{(i)} E_{22} X(i) + 2E_{12}$	$3\lambda_2^{(i)} E_{12} X^2(i) + 6E_{12} X(i)$	$4\lambda_2^{(i)} E_{22} X^3(i) + 12E_{12} X^2(i)$	$5\lambda_2^{(i)} E_{22} X^4(i) + 20E_{12} X^3(i)$
Row 3	$2g^{(i)} X(i) - 2\lambda_2^{(i)} E_{11}$	$3g^{(i)} X^2(i) - 6\lambda_2^{(i)} E_{11} X(i) - 6E_{11}$	$4g^{(i)} X^3(i) - 12\lambda_2^{(i)} E_{11} X^2(i) - 24E_{11} X(i)$	$5g^{(i)} X^4(i) - 20\lambda_2^{(i)} E_{11} X^3(i) - 60E_{11} X^2(i)$

where $g^{(i)} = (\lambda_2^{(i)})^2 E_{22} - \lambda_4^{(i)} E_{12}$ and

$$[\tilde{S}^{(i)}] = \begin{bmatrix} \tilde{S}_1^{(i)} \\ \tilde{S}_2^{(i)} \end{bmatrix}$$

STRUCTURAL ELEMENT DESCRIPTIONS

6. Transform the stress matrix to grid point degrees of freedom

$$[\bar{S}] = [\tilde{S}][\bar{r}_{\beta q}] \quad (99)$$

7. Transform the stress matrix from local to system coordinates

$$[S] = [\bar{S}][r_{rs}] \quad (100)$$

8. Transform the stress matrix to global coordinates

$$[S]_g = [S][T_{12}] \quad (101)$$

9. Compute the thermal stress vector for the three stress points:

$$\{T_s^{(i)}\} = \left\{ \begin{array}{l} \Delta T_1^{(m)} H_m (E_{11}\alpha_1 + E_{12}\alpha_2) + \Delta T_2^{(m)} H_m \frac{x_1}{S} (E_{11}\alpha_1 + E_{12}\alpha_2) \\ \Delta T_1^{(m)} H_m (E_{21}\alpha_1 + E_{22}\alpha_2) + \Delta T_2^{(m)} H_m \frac{x_1}{S} (E_{21}\alpha_1 + E_{22}\alpha_2) \\ \Delta T_1^{(f)} \frac{H_f^3}{12} (E_{11}\alpha_1 + E_{12}\alpha_2) + \Delta T_2^{(f)} \frac{H_f^3 x_1}{12S} (E_{11}\alpha_1 + E_{12}\alpha_2) \\ -\Delta T_1^{(f)} \frac{H_f^3}{12} (E_{21}\alpha_1 + E_{22}\alpha_2) - \Delta T_2^{(f)} \frac{H_f^3 x_1}{12S} (E_{21}\alpha_1 + E_{22}\alpha_2) \\ \Delta T_1^{(f)} \frac{H_f^3}{12} \lambda_2^{(i)} [(E_{11} - E_{12})\alpha_1 + (E_{12} - E_{22})\alpha_2] \\ + \Delta T_2^{(f)} \frac{H_f^3}{12S} \left\{ \lambda_2^{(i)} x_1 [(E_{11} - E_{12})\alpha_1 \right. \\ \left. + (E_{12} - E_{22})\alpha_2] + [E_{11}\alpha_1 + E_{12}\alpha_2] \right\} \end{array} \right\} \quad (5 \times 1) \quad (102)$$

where $\Delta T_1^{(m)}$, $\Delta T_2^{(m)}$, $\Delta T_1^{(f)}$ and $\Delta T_2^{(f)}$ are as given in Equations 66 through 69.

10. Form the master thermal stress vector

$$\{T_s\} = \begin{pmatrix} T_s^{(1)} \\ T_s^{(2)} \\ T_s^{(3)} \end{pmatrix} \quad (103)$$

Phase 2 calculations are as follows:

1. Extract the displacement vector, $\{\Delta\}$, at the two grid points from the global displacement vector.
2. Calculate the element forces:

$$\{P\} = [K]\{\Delta\} \quad (104)$$

3. Calculate the element stresses without regard to thermal loading:

$$\{\sigma'\} = [S]_g \{\Delta\} \quad (12 \times 1) \quad (105)$$

4. If there is no thermal loading, then

$$\{\sigma\} = \{\sigma'\} \quad (106)$$

5. If there is thermal loading, then

$$\sigma_j = \sigma_j' - (T_1 - T_0) T_{Sj} - (T_2 - T_1) T_{Sj} + 15 \quad (107)$$

for $j = 1$ and 2 and

$$\sigma_j = \sigma_j' \quad (108)$$

for $j = 3, 4, 5$.

THE VISC ELEMENT

8.13 THE VISC ELEMENT

The viscous element, VISC, has the same properties as the RØD element except that instead of generating a contribution to the stiffness matrix, the element generates a contribution to the damping matrix, $[B_{gg}]$.

8.13.1 Input Data for the VISC Element

1. The ECPT/EST entries for the VISC are:

<u>Symbol</u>	<u>Description</u>
SIL_a, SIL_b	Scalar indices for grid points a and b
$\left. \begin{array}{l} N_a, X_a, Y_a, Z_a \\ N_b, X_b, Y_b, Z_b \end{array} \right\}$	Local coordinate system number and basic coordinates of grid points.
C_1	Extensional damping coefficient
C_2	Torsional damping coefficient

Given $N_a, X_a, Y_a, Z_a, N_b, X_b, Y_b$ and Z_b and the CSTM (Coordinate System Transformation Matrices) data block, the 3 by 3 transformation matrices, $[T_a]$ and $[T_b]$, are calculated using utility routine TRANSD (see section 3.4.37).

8.13.2 Damping Matrix Calculations (Subroutine BVISC of Module SMA2)

1. Generate $\{n\}$ as in Equation 2, Section 8.1.2, and generate the transformation matrices $[T_a]$ and $[T_b]$.
2. Calculate:

$$[b_t] = C_1 \begin{bmatrix} n_1^2 & n_1 n_2 & n_1 n_3 \\ n_1 n_2 & n_2^2 & n_2 n_3 \\ n_1 n_3 & n_2 n_3 & n_3^2 \end{bmatrix}, \quad (1)$$

STRUCTURAL ELEMENT DESCRIPTIONS

$$[b_r] = c_2 \begin{bmatrix} n_1^2 & n_1 n_2 & n_1 n_3 \\ n_1 n_2 & n_2^2 & n_2 n_3 \\ n_1 n_3 & n_2 n_3 & n_3^2 \end{bmatrix} . \quad (2)$$

3. The 6x6 damping matrices are:

$$[B_{aa}] = \begin{bmatrix} T_a^T b_t T_a & 0 \\ 0 & T_a^T b_r T_a \end{bmatrix} . \quad (3)$$

$$[B_{ab}] = - \begin{bmatrix} T_a^T b_t T_b & 0 \\ 0 & T_a^T b_r T_b \end{bmatrix} . \quad (4)$$

$$[B_{bb}] = \begin{bmatrix} T_b^T b_t T_b & 0 \\ 0 & T_b^T b_r T_b \end{bmatrix} . \quad (5)$$

$$[B_{ba}] = - \begin{bmatrix} T_b^T b_t T_a & 0 \\ 0 & T_b^T b_r T_a \end{bmatrix} . \quad (6)$$

4. These matrices are added to $[B_{gg}]$.

8.14 INTEGRAL CALCULATIONS FOR THE TRIARG, TRAPRG ELEMENTS

Integrals of the form

$$I_{pq} = \int_{R_j}^{R_i} \int_{z_{mn}}^{z_{kl}} r^p z^q dz dr \quad (1)$$

must be calculated for the TRIARG and TRAPRG elements (see Sections 8.10.3 and 8.11.3).

The integration may be performed for any integer values of p and q. The area of integration is defined by the two lines $r = R_i$ and $r = R_j$, and by the two lines $z = b_{kl}r + a_{kl}$ and $z = b_{mn}r + a_{mn}$.

This integration is performed by the integration "driver" subroutine, FØRTRAN function DK1 in module SMA1, FØRTRAN function DMI in module SMA2, and FØRTRAN function AI in module SDR2.

The following input data are necessary for these routines:

- p - an integer that defines the power of the r variable.
- q - an integer that defines the power of the z variable.
- {R} - a vector of the r coordinates of all points used to describe the area of integration.
- {Z} - a vector of the z coordinates of all points used to describe the area of integration.
- k,l - the subscripts of R, Z defining one of the lines of the limit of integration (i.e., the line between points (r_k, z_k) and (r_l, z_l)).
- m,n - the subscripts of R, Z defining the second line on the limit of integration.
- i,j - the subscripts of R defining the other two lines on the limit of integration.

In the following paragraphs FØRTRAN names of functions auxiliary to DK1 are given. The corresponding FØRTRAN function names auxiliary to functions DMI and AI can be found in sections 4.28.8 and 4.46.8 respectively.

STRUCTURAL ELEMENT DESCRIPTIONS

The following slopes and y-intercepts are calculated in functions DKK and DKM

$$a_{k\ell} = \frac{R_\ell Z_k - R_k Z_\ell}{R_\ell - R_k}, \quad (2)$$

$$b_{k\ell} = \frac{Z_\ell - Z_k}{R_\ell - R_k}, \quad (3)$$

$$a_{mn} = \frac{R_n Z_m - R_m Z_n}{R_n - R_m}, \quad (4)$$

$$b_{mn} = \frac{Z_n - Z_m}{R_n - R_m}. \quad (5)$$

A test for a vanishing area of integration is made: if $R_i = R_j$, then $I_{pq} = 0$;
if $a_{k\ell} = a_{mn}$ and $b_{k\ell} = b_{mn}$, then $I_{pq} = 0$.

The formulas for evaluation of the integrals are dependent upon the values of p and q as given in the following sections.

8.14.1 Integral Calculation for $q > 0$ and any p . (Function DKINT)

Define the function

$$f_1(x, y) = \frac{1}{q+1} \sum_{t=0}^{q+1} C x^t y^{q+1-t} D, \quad (6)$$

where

$$C = \begin{cases} \prod_{s=1}^t \frac{q+1-s+1}{s} & \text{for } t \neq 0 \\ 1 & \text{for } t = 0 \end{cases}, \quad (7)$$

$$D = \begin{cases} \left[\frac{R_j^{(q+1+p+1-t)} - R_i^{(q+1+p+1-t)}}{q+1+p+1-t} \right] & \text{for } (q+1+p+1-t) \neq 0 \\ \ln(R_j/R_i) & \text{for } (q+1+p+1-t) = 0 \end{cases} \quad (8)$$

C and D are calculated in functions DKEF and DKJ respectively.

The integral is

$$I_{pq} = f_1(a_{mn}, b_{mn}) - f_1(a_{kl}, b_{kl}). \quad (9)$$

 8.14.2 Integral Calculation for $p \geq 0$ and $q < -1$ (Function DK89)

$$f_2(\alpha, x, y) = \frac{1}{y^{p+1}} \sum_{s=0}^p p! (-x)^s D, \quad (10)$$

where

$$D = \begin{cases} \frac{(x+yR_\alpha)^{p+1+q+1-s}}{(p-s)!s!(p+1+q+1-s)} & \text{for } (p+1+q+1-s) \neq 0 \\ \frac{\ln|x+yR_\alpha|}{(p+1+q+1)!(-q-2)!} & \text{for } (p+1+q+1-s) = 0 \end{cases}, \quad (11)$$

and $\alpha = i$ or j .

The integral is

$$I_{pq} = \frac{1}{2+q} [f_2(i, a_{kl}, b_{kl}) - f_2(i, a_{mn}, b_{mn}) - f_2(j, a_{kl}, b_{kl}) + f_2(j, a_{mn}, b_{mn})] \quad (12)$$

8.14.3 Integral Calculation for $p < 0$ and $q < -1$ (Function DK100)

Define the function

$$f_3(\alpha, x, y) = \frac{-1}{x^{(-p-q-2)}} \sum_{s=0}^{-p-q-3} (-p-q-3)! D, \quad (13)$$

where

$$D = \begin{cases} \frac{(x+yR_\alpha)^{(-p-1-s)} (-y)^s}{(-p-q-3-s)! s! (-p-1-s) R_\alpha^{(-p-1-s)}} & \text{for } (-p-1-s) \neq 0 \\ \frac{(-y)^{-p-1} \ln \left(\left| \frac{x+yR_\alpha}{R_\alpha} \right| \right)}{(-p-1)! (-q-2)!} & \text{for } (-p-1-s) = 0 \end{cases} \quad (14)$$

and $\alpha = i$ or j .

The integral is

$$I_{pq} = \frac{1}{2+q} [f_3(i, a_{kl}, b_{kl}) - f_3(i, a_{mn}, b_{mn}) - f_3(j, a_{kl}, b_{kl}) + f_3(j, a_{mn}, b_{mn})] \quad (15)$$

INTEGRAL CALCULATIONS FOR THE TRIARG, TRAPRG ELEMENTS

8.14.4 Integral Calculations for $p > -1$ and $q = -1$ (Function DKJAB)

Define the function

$$f_4(\alpha, x, y) = \frac{R_\alpha^{p+1} \ln(|x+yR_\alpha|)}{p+1} - \frac{y}{p+1} f_2(\alpha, x, y), \quad (16)$$

where f_2 is given in Equation 10.

The integral is

$$\begin{aligned} I_{pq} &= f_4(i, a_{k\ell}, b_{k\ell}) - f_4(i, a_{mn}, b_{mn}) \\ &\quad - f_4(j, a_{k\ell}, b_{k\ell}) - f_4(j, a_{mn}, b_{mn}) \end{aligned} \quad (17)$$

8.14.5 Integral Calculations for $p < -1$ and $q = -1$ (Function DK219)

Define the function

$$f_5(\alpha, x, y) = -\frac{\ln(|x+yR_\alpha|)}{(-p-1)R_\alpha^{(-p-1)}} + \frac{y}{(-p-1)} f_3(\alpha, x, y), \quad (18)$$

where f_3 is given in Equation 13.

The integral is

$$\begin{aligned} I_{pq} &= f_5(i, a_{k\ell}, b_{k\ell}) - f_5(i, a_{mn}, b_{mn}) \\ &\quad - f_5(j, a_{k\ell}, b_{k\ell}) - f_5(j, a_{mn}, b_{mn}) \end{aligned} \quad (19)$$

STRUCTURAL ELEMENT DESCRIPTIONS

8.14.6 Integral Calculations for $p = -1$ and $q = -1$ (Function DK211)

Define the function

$$f_6(\alpha, x, y) = \begin{cases} 0, & \text{for } yR_\alpha = x \\ \frac{1}{2} [\ln(|2 y R_\alpha|)]^2, & \text{for } (yR_\alpha)^2 = x^2 \\ & \text{and } yR_\alpha \neq x \\ \ln|x| \ln|R_\alpha| - \sum_{t=1}^{\infty} \frac{1}{t^2} \left[\frac{-yR_\alpha}{x} \right]^t, & \text{for } (yR_\alpha)^2 < x^2 \\ \frac{1}{2} [\ln(|yR_\alpha|)]^2 + \sum_{t=1}^{\infty} \frac{1}{t^2} \left[\frac{-x}{yR_\alpha} \right]^t, & \text{for } (yR_\alpha)^2 < x^2 \end{cases} \quad (20)$$

The summation term is calculated until its value becomes less than 1.0×10^{-6} .

The integral is

$$\begin{aligned} I_{pq} &= f_6(i, a_{kl}, b_{kl}) - f_6(i, a_{mn}, b_{mn}) \\ &- f_6(i, a_{kl}, b_{kl}) - f_6(i, a_{mn}, b_{mn}) \end{aligned} \quad (21)$$

THE FLUID2, FLUID3, FLUID4, AXIF2, AXIF3, AXIF4, AND MFREE ELEMENTS

8.15 THE FLUID2, FLUID3, FLUID4, AXIF2, AXIF3, AXIF4, AND MFREE ELEMENTS

8.15.1 Input Data for the Fluid Elements

1. The ECPT/EST entries for the FLUID2 element are:

<u>Symbol</u>	<u>Description</u>
SIL_1, SIL_2	Scalar indices for the connected scalar points
$N_i = 0, r_i, z_i, 0$ $i = 1, 2$	Reference number for the basic coordinate system and locations in the fluid coordinate systems.
ρ	Fluid density
B	Fluid bulk modulus
n	Harmonic number

2. The ECPT/EST entries for the FLUID3 and FLUID4 elements are identical except that three and four points are used respectively.
3. The ECPT/EST entries for the MFREE element are identical to the FLUID2 element except that a weight factor, γ , is used instead of ρ and B .
4. No other material or coordinate system data is necessary.
5. The AXIF elements are identical to the FLUID elements at this stage.

8.15.2 Matrix Calculations for the FLUID2 Element (Subroutine KFLUD2 of Module SMA1 and Subroutine MFLUD2 of Module SMA2)

The FLUID2 element is intended to model a fluid in the region adjacent to and including the axis of symmetry. The volume is defined by two circular ring points in the fluid. The shape is that of a disc having a conical or cylindrical outer boundary.

1. The integral parameters, for the stiffness matrix, $I_{2n,0}$, $I_{2n,1}$, $I_{2n,2}$, and $I_{2n+2,0}$ are calculated according to the following equations:

$$\left. \begin{aligned} I_{2n,0} &= I_{2n,1} = I_{2n,2} = 0 \\ I_{2n+2,0} &= \frac{1}{6} (z_2 - z_1)(r_2^2 + r_1 r_2 + r_1^2) \end{aligned} \right\} n = 0 \quad (1)$$

STRUCTURAL ELEMENT DESCRIPTIONS

$$\text{If } \left| \frac{r_2 - r_1}{z_2 - z_1} \right| < 10^{-6} :$$

$$\left. \begin{aligned} I_{2n,0} &= \left[\frac{r_1 + r_2}{2} \right]^{2n} \left(\frac{z_2 - z_1}{2n} \right) \\ I_{2n,1} &= \frac{I_{2n,0}}{2} (z_2 + z_1) \\ I_{2n,2} &= \frac{I_{2n,0}}{3} (z_2^2 + z_1 z_2 + z_1^2) \\ I_{2n+2,0} &= I_{2n,0} \left(\frac{2n}{2n+2} \right) r_1^2 \end{aligned} \right\} n > 0 . \quad (2)$$

$$\text{If } \left| \frac{r_2 - r_1}{z_2 - z_1} \right| > 10^{-6} :$$

$$D = \frac{z_2 - z_1}{r_2 - r_1} . \quad (3)$$

$$\left. \begin{aligned} I_{2n,0} &= \frac{D}{2n(2n+1)} (r_2^{2n+1} - r_1^{2n+1}) \\ I_{2n,1} &= \frac{D}{2n(2n+1)} \left[r_2^{2n+1} z_2 - r_1^{2n+1} z_1 - \left(\frac{D}{2n+2} \right) (r_2^{2n+2} - r_1^{2n+2}) \right] \\ I_{2n,2} &= \frac{D}{2n(2n+1)} \left\{ r_2^{2n+1} z_2^2 - r_1^{2n+1} z_1^2 - \left(\frac{2D}{2n+2} \right) [r_2^{2n+2} z_2 - r_1^{2n+2} z_1] \right. \\ &\quad \left. - \frac{D}{2n+3} (r_2^{2n+3} - r_1^{2n+3}) \right\} \\ I_{2n+2,0} &= \frac{D}{(2n+2)(2n+3)} [r_2^{2n+3} - r_1^{2n+3}] \end{aligned} \right\} n > 0 \quad (4)$$

2. The integral parameters for the mass matrix, $I_{2n+2,0}$, $I_{2n+2,1}$, and $I_{2n+2,2}$ are calculated with the same equations as above except the value $k = 2n+2$ is substituted for $k = 2n$.
3. The transformation matrix $[H_{qp}^n]$ is defined as:

$$[H_{pq}^n] = \frac{1}{z_2 - z_1} \begin{bmatrix} \frac{z_2}{r_1^n} & -\frac{z_1}{r_2^n} \\ -\frac{1}{r_1^n} & \frac{1}{r_2^n} \end{bmatrix} \quad (5)$$

4. The stiffness matrix is:

$$[K_p^n] = \frac{\pi}{\rho} [H_{pq}^n]^T \begin{bmatrix} (2n^2 I_{2n,0}) & (2n^2 I_{2n,1}) \\ (2n^2 I_{2n,1}) & (2n^2 I_{2n,2} + I_{2n+2,0}) \end{bmatrix} [H_{pq}^n] \quad (6)$$

$n > 0$

Note: if $n = 0$, a factor of 2 is used.

5. The mass matrix is:

$$[M_p^n] = \frac{\pi}{B} [H_{pq}^n]^T \begin{bmatrix} I_{2n+2,0} & I_{2n+2,1} \\ I_{2n+2,1} & I_{2n+2,2} \end{bmatrix} [H_{pq}^n], \quad n > 0 \quad (7)$$

Note: if $n = 0$ a factor of 2 is used.

6. Various tests are performed for the element.

If $|z_2 - z_1| = 0$, the calculations are skipped,

if $r_1 = 0$ or $r_2 = 0$, a fatal error exists,

if $\rho = 0$, a fatal error exists,

if $B = 0$, the mass calculations are skipped.

STRUCTURAL ELEMENT DESCRIPTIONS

8.15.3 Matrix Calculations for the FLUID3 Element (Subroutines KFLUD3 of Module SMA1 and Subroutine MFLUD3 of Module SMA2)

The FLUID3 element is used to model a volume of fluid defined by three connected fluid ring points.

1. The three connected points are arranged in the order such that the area factor, R , is positive. The area factor is defined by the equation:

$$R = (r_2 - r_1)(z_3 - z_1) - (r_3 - r_1)(z_2 - z_1) . \quad (8)$$

2. The transformation matrix, $[H_{pq}]$, is calculated as:

$$[H_{pq}] = \frac{1}{R} \begin{bmatrix} (r_2 z_3 - r_3 z_2) & (r_3 z_1 - r_1 z_3) & (r_1 z_2 - r_2 z_1) \\ (z_2 - z_3) & (z_3 - z_1) & (z_1 - z_2) \\ (r_3 - r_2) & (r_1 - r_3) & (r_2 - r_1) \end{bmatrix} . \quad (9)$$

3. The integral parameters, $I_{k\ell}$, for the stiffness matrix are the sum of the integrals, $G_{k\ell}$, for each of the three sides. The points defining each side are:

<u>SIDE</u>	<u>POINTS - a,b</u>
1	1, 2
2	2, 3
3	3, 1

The following parameters are used to generate the integrals, $G_{k\ell}$:

$$\left. \begin{aligned} \Delta r &= r_b - r_a \\ \Delta z &= z_b - z_a \\ \beta &= z_a - r_a \frac{\Delta z}{\Delta r} \end{aligned} \right\} . \quad (10)$$

The integrals for each side are:

$$\begin{aligned}
 G_{00} &= \beta \log \frac{r_a}{r_b} - \Delta z \\
 G_{10} &= -\beta \Delta r + \frac{\Delta z}{2\Delta r} (r_a^2 - r_b^2) \\
 G_{20} &= \frac{1}{2} \beta (r_a^2 - r_b^2) + \frac{\Delta z}{3\Delta r} (r_a^3 - r_b^3) \\
 G_{01} &= \frac{1}{2} \beta^2 \log \frac{r_a}{r_b} - \beta \Delta z + \frac{1}{4} \frac{\Delta z^2}{\Delta r^2} (r_a^2 - r_b^2) \\
 G_{11} &= -\frac{1}{2} \beta^2 \Delta r + \frac{1}{2} \beta \frac{\Delta z}{\Delta r} (r_a^2 - r_b^2) + \frac{1}{6} \left(\frac{\Delta z}{\Delta r} \right)^2 (r_a^3 - r_b^3) \\
 G_{02} &= \frac{1}{3} \beta^3 \log \frac{r_a}{r_b} - \beta^2 \Delta z + \frac{1}{2} \beta \left(\frac{\Delta z}{\Delta r} \right)^2 (r_a^2 - r_b^2) + \frac{1}{9} \left(\frac{\Delta z}{\Delta r} \right)^3 (r_a^3 - r_b^3)
 \end{aligned} \tag{11}$$

4. The stiffness matrix is:

$$[K_p^n] = \frac{\pi}{\rho} [H_{pq}]^T \begin{bmatrix} n^2 I_{00} & n^2 I_{10} & n^2 I_{01} \\ n^2 I_{10} & (n^2+1) I_{20} & n^2 I_{11} \\ n^2 I_{01} & n^2 I_{11} & (n^2 I_{02} + I_{20}) \end{bmatrix} [H_{pq}] . \tag{12}$$

The matrix terms are multiplied by two if $n = 0$.

5. The mass matrix terms are simply:

$$M_{ij}^n = \frac{\pi A}{60B} (r_1 + r_2 + r_3 + r_i + r_j) c_{ij} , \tag{13}$$

where

$$c_{ij} = 2 , \quad i = j,$$

$$c_{ij} = 1 , \quad i \neq j,$$

and

$$A = \frac{R}{2} \text{ is the area.}$$

STRUCTURAL ELEMENT DESCRIPTIONS

6. The tests performed are:

- if $r_i = 0$ a fatal error exists,
- if $R = 0$ the routine exits,
- if $\rho = 0$ a fatal error exists,
- if $B = 0$ the mass routine exits.

8.15.4 Matrix Generation for the FLUID4 Element (Subroutine KFLUD4 in Module SMA1 and Subroutine MFLUD4 in Module SMA2)

This element describes an axisymmetric volume of fluid defined by four fluid ring points. It is actually solved by subdividing the quadrilateral cross section into four triangles and calling the appropriate FLUID3 subroutine for each of the triangles. The parameters ρ and B are multiplied by two in order to account for the overlapping volumes and reduce the matrix terms.

1. A test is made in the stiffness routine to check the interior angles which must be less than 180° . For each of the four triangles, the area factor K is calculated which will be positive if the order of the points is counterclockwise. If K is negative for one or three out of the four triangles, a fatal error exists.
2. The triangles and their three connected points are:

<u>Triangle</u>	<u>Connected Points</u>		
	a	b	c
I	1	2	3
II	1	2	4
III	1	3	4
IV	2	3	4

(14)

The ECPT data is moved to a temporary storage space and the original ECPT is used for the data for each triangle.

3. Since matrix terms are only created if one of the connected points is the "pivot point", a test is made and the FLUID3 subroutine is not called if the "pivot point" is not one of the three points.

8.15.5 Matrix Calculations for the MFREE Element (Subroutine MFREE in Module SMA2)

The data for this element is generated by subroutine IFP4 from the free surface information and is not available as a user-input element. The element describes the effect of gravity on a surface in between two fluid ring points. In a special case, the surface is interior to a circle defined by one fluid ring point.

1. If the two connected points are identical ($SIL_1 = SIL_2$), the special case exists and the equations are:

$$\left. \begin{aligned} M_{ii} &= \frac{\pi r^2}{2\gamma(2n+2)} , & n > 0 \\ M_{ii} &= \frac{\pi r^2}{2\gamma} , & n = 0 \end{aligned} \right\} \quad (15)$$

A factor of two is included in the denominator because the terms will be calculated twice.

2. If the connected points are unique, the equation for the mass matrix is:

$$[M_p^n] = \frac{\pi(r_2 - r_1)}{12\gamma} \begin{bmatrix} 3r_1 + r_2 & r_1 + r_2 \\ r_1 + r_2 & 3r_2 + r_1 \end{bmatrix} , \quad n > 0. \quad (16)$$

The values are multiplied by two for $n = 0$.

8.15.6 Stress Calculations for the AXIF Elements, Phase 1.

The SDR2 calculations for these elements are actually the calculations of the velocity of the fluid passing through a fluid element.

The data placed on the ESTB file are:

1. Id_e - Element Id
2. $SIL_1, SIL_2, (SIL_3, SIL_4)$ - Scalar indices of connected points
3. $[S_v]$ - the velocity-pressure matrix

The $[S_v]$ matrix for the CAXIF2 element is a four by two matrix given as follows:

STRUCTURAL ELEMENT DESCRIPTIONS

$$[S_v] = \begin{bmatrix} S_c^2 \\ S_e^2 \end{bmatrix} \quad (17)$$

where

$$\begin{pmatrix} v_{rc} \\ v_{zc} \\ v_{se} \\ v_{\phi e} \end{pmatrix} = [S_v] \begin{pmatrix} p_1 \\ p_2 \end{pmatrix} \quad (18)$$

v_{rc} and v_{zc} are velocities at $r = 0$, v_{se} and $v_{\phi e}$ are velocities at the midpoint of the outer edge, along the edge and circumferential.

The two by two matrix $[S_c]$, for the center, is:

$$[S_c^2] = \frac{1}{\rho} \begin{bmatrix} 0 & 0 \\ \frac{1}{z_2 - z_1} & \frac{-1}{z_2 - z_1} \end{bmatrix} \quad n = 0 \quad (19)$$

$$[S_c^2] = \frac{1}{\rho} \begin{bmatrix} \frac{-1}{r_1 + r_2} & \frac{-1}{r_1 + r_2} \\ 0 & 0 \end{bmatrix} \quad n = 1 \quad (20)$$

$$[S_c^2] = [0] \quad n > 1 \quad (21)$$

The two by two $[S_e^2]$ matrix, for the outer edge is:

$$[S_e^2] = -\frac{(\bar{r})^{n-1}}{\rho \ell} \begin{bmatrix} n\Delta r & (n\bar{z}\Delta r + \bar{r}\Delta z) \\ n\ell & n\bar{z}\ell \end{bmatrix} [H_{qp}^n] \quad (22)$$

where

$$\left. \begin{aligned}
 \Delta r &= r_2 - r_1, \\
 \Delta z &= z_2 - z_1, \\
 \bar{r} &= \frac{1}{2} (r_2 + r_1), \\
 \bar{z} &= \frac{1}{2} (z_2 + z_1), \\
 \ell &= \sqrt{\Delta r^2 + \Delta z^2}, \\
 [H_{qp}^n] &= \frac{1}{\Delta z} \begin{bmatrix} \frac{z_2}{r_1^n} & -\frac{z_1}{r_2^n} \\ \frac{-1}{r_1^n} & \frac{1}{r_2^n} \end{bmatrix}
 \end{aligned} \right\} \quad (23)$$

The nine by three $[S_v^t]$ matrix for the CAXIF3 element is calculated with the following equations

$$[S_v] = \begin{bmatrix} [S_c^t] \\ \hline [S_e^t] \end{bmatrix} \quad (24)$$

The three by three $[S_c^t]$ matrix relates three pressures to the three velocities in the basic coordinate system V_r, V_ϕ, V_z .

$$[S_e^t] = -\frac{1}{\rho} \begin{bmatrix} 0 & 1 & 0 \\ \frac{n}{r_c} & n & \frac{nz_c}{r_c} \\ 0 & 0 & 1 \end{bmatrix} [H_{qp}^n] \quad (25)$$

where $[H_{qp}^n]$ is a three by three transformation matrix between pressures and generalized coordinates defined in Section 8.15.3.

The six by three matrix, $[S_e^t]$, which defines the velocities at each edge, tangential and circumferential, is:

STRUCTURAL ELEMENT DESCRIPTIONS

$$[S_e^t] = \frac{1}{\rho} \begin{bmatrix} \frac{1}{l_{12}} & -\frac{1}{l_{12}} & 0 \\ \frac{n}{r_1+r_2} & \frac{n}{r_1+r_2} & 0 \\ 0 & \frac{1}{l_{23}} & -\frac{1}{l_{23}} \\ 0 & \frac{n}{r_2+r_3} & \frac{n}{r_2+r_3} \\ -\frac{1}{l_{13}} & 0 & \frac{1}{l_{13}} \\ \frac{n}{r_1+r_3} & 0 & \frac{n}{r_1+r_3} \end{bmatrix} \quad (26)$$

where $l_{ij} = \sqrt{(r_j - r_i)^2 + (z_j - z_i)^2}$

The CAXIF4 element is composed of four overlapping triangles. For each triangle I, II, III or IV the connected points 1, 2, 3, 4 are allocated as follows:

<u>Triangles</u>	<u>Connected points a, b, c</u>
I	1 2 3
II	1 2 4
III	1 3 4
IV	2 3 4

For each triangle calculate the 3x3 $[S_c^t]$ matrix from Equation 9 and add each column to one of four columns corresponding to the connected point. The results are divided by 4 to provide an average $[S_c^q]$ matrix for the quadrilateral.

The $[S_c^q]$ matrix for the quadrilateral is:

$$[S_e^q] = -\frac{1}{p} \begin{bmatrix} \frac{1}{l_{12}} & -\frac{1}{l_{12}} & & & \\ \frac{n}{r_1+r_2} & \frac{n}{r_1+r_2} & & & \\ & \frac{1}{l_{23}} & -\frac{1}{l_{23}} & & \\ & \frac{n}{r_2+r_3} & \frac{n}{r_2+r_3} & & \\ & & \frac{1}{l_{34}} & -\frac{1}{l_{34}} & \\ & & \frac{n}{r_3+r_4} & \frac{n}{r_3+r_4} & \\ -\frac{1}{l_{41}} & & & \frac{1}{l_{41}} & \\ \frac{n}{r_4+r_1} & & & \frac{n}{r_4+r_1} & \end{bmatrix} \quad (27)$$

where $l_{ij} = \sqrt{(r_j - r_i)^2 + (z_j - z_i)^2}$.

The resulting $[S_v]$ matrix for the quadrilateral CAXIF4 element is:

$$[S_v] = \begin{bmatrix} S_c^q \\ \hline S_e^q \end{bmatrix} \quad (28)$$

8.15.7 Stress Calculations for the AXIF Elements, Phase 2.

The element identification number, the indices of the connected points, and the $[S_v]$ matrices are given in the ESTB table. The pressures at the connected points, $\{P_i\}$, are given in the UGV matrix data block. Depending on the rigid format, the pressure values are either real or complex numbers and associated with each vector of pressures is a real eigenvalue, λ ; a frequency, f , or a complex eigenvalue, P . The equation for velocity is

$$\{V\} = \frac{1}{\omega} [S_v] P_i \quad (29)$$

where $\{V\}$ is the vector of velocities in the element.

$\omega = \sqrt{|\lambda|}$ (real) in Rigid Format 3 ($\{P_i\}$ is real)

$\omega = 2\pi f$ (real) in Rigid Formats 8 and 11 ($\{P_i\}$ is complex)

$\omega = p$ (complex) in Rigid Formats 7 and 10 ($\{P_i\}$ is complex)

$\omega = 1.0$ in all other Rigid Formats ($\{P_i\}$ is real)

and $[S_v]$ is dimensioned 4x2, 9x3, or 11x4

for the CAXIF2, CAXIF3, and CAXIF4 elements respectively.

8.16 THE SLØT3 AND SLØT4 FLUID ELEMENTS

8.16.1 Input Data for the SLØT3 And SLØT4 Elements

1. The ECPT/EST entries for the SLØT3 are:

<u>Symbol</u>	<u>Description</u>
SIL_1, SIL_2, SIL_3	Scalar indices for the connected grid points
$r_i, z_i, w_i, i = 1,2,3$	Radius and axis location and slot width of connected grid points, i.
ρ	Density
B	Bulk Modulus
M	Number of Slots
N	Harmonic Number

2. ECPT/EST entries for the SLØT4 are the same as for the SLØT3 except four points are used.

8.16.2 General Calculations for the SLØT Elements

1. The overall factor for the number of slots is:

$$F = M, \quad 2N = 0, M, 2M, 3M, \dots \quad (1)$$

$$F = \frac{M}{2}, \quad 2N \neq 0, M, 2M, 3M, \dots \quad (2)$$

2. The SLØT4 element is composed of four overlapping triangles. If the SLØT4 element is used, its data is rearranged to the SLØT3 format and the following operations are carried out for all four subtriangles. A test is made on the direction of the vector normal to the surface of all four triangles if the number of negative normal vectors (NNEG) is one or three, a valid quadrilateral is impossible and a fatal error is set.

8.16.3 Stiffness Matrix Generation for the SLØT3 Element

1. For each triangle the following terms are calculated:

$$2 \cdot A = A_2 = [r_1(z_2 - z_3) + r_2(z_3 - z_1) + r_3(z_1 - z_2)] \quad (3)$$

$$C_o = \frac{F}{6\rho|A_2|} [w_1 + w_2 + w_3] \quad (4)$$

$$\left. \begin{aligned} F_{ir} &= (r_k - r_j) \\ F_{iz} &= (z_j - z_k) \end{aligned} \right\} (i,j,k) = \begin{cases} (1,2,3) \\ (2,3,1) \\ (3,1,2) \end{cases} \quad (5)$$

2. The stiffness matrix terms are:

$$K_{ij} = C_o [F_{ij} F_{jr} + F_{iz} F_{jz}] \quad (6)$$

where

i = the "pivot point"

j = 1, 2, 3

8.16.4 Mass Matrix Generation for the SLØT3 Elements

1. The following coefficients are generated:

$$2 \cdot A = A_2 = (r_2 - r_1)(z_3 - z_1) - (r_3 - r_1)(z_2 - z_1) \quad (7)$$

$$\bar{w} = w_1 + w_2 + w_3 + w_i, \quad (8)$$

where i is the "pivot point"

$$C_o = \frac{F|A_2|}{120 B}$$

2. The mass matrix terms are:

$$M_{ij} = C_o(\bar{w} + w_j) \quad j = 1, 2, 3 \neq i$$

$$M_{ij} = 2C_o(\bar{w} + w_j) \quad j = i$$

where i is the "pivot point".

8.16.5 Stress Matrix Calculations in the SLØT Elements (Phase 1)

The velocities in the SLØT elements are calculated in the same manner as stresses in a structural element. Phase 1 involves calculating pressure field - velocity matrices of the fluid passing through the element.

1. The data placed on the ESTB file are:

Id_e - element identification number

$SIL_1, SIL_2, \dots, SIL_i$ - scalar indices

$[S_v]$ - matrix relation between pressure and velocity.

2. The $[S_v]$ matrix for the CSLØT3 element is a five by three matrix given as follows:

$$[S_v] = -\frac{1}{\rho} \begin{bmatrix} \frac{z_2 - z_3}{A} & \frac{z_3 - z_1}{A} & \frac{z_1 - z_2}{A} \\ \frac{r_3 - r_2}{A} & \frac{r_1 - r_3}{A} & \frac{r_2 - r_1}{A} \\ \hline -\frac{1}{l_{12}} & \frac{1}{l_{12}} & 0 \\ 0 & -\frac{1}{l_{23}} & \frac{1}{l_{23}} \\ \frac{1}{l_{23}} & 0 & -\frac{1}{l_{13}} \end{bmatrix} = \begin{bmatrix} s_v^t \\ \hline s_v^e \end{bmatrix} \quad (9)$$

where

$$l_{ij} = \sqrt{(r_j - r_i)^2 + (z_j - z_i)^2} \quad (10)$$

$$A = \frac{1}{2} [r_1(z_2 - z_3) + r_2(z_3 - z_1) + r_3(z_1 - z_2)]$$

The five rows of the matrix correspond to the velocities V_{rc} and V_{zc} at the centroid in the r and z direction and V_1, V_2, V_3 corresponding to velocities along the three edges.

3. The CSLØT4 element is composed of four overlapping triangles. The velocity at the intersection of the triangles is calculated to be the average of the velocities in each sub-triangle. The subtriangles I, II, III and IV are each given three of the four points 1, 2, 3, 4 as in the following chart:

Triangle Number	Connected Points		
	a	b	c
I	1	2	3
II	1	2	4
III	1	3	4
IV	2	3	4

The $[S_V^t]$ matrix for each triangle is calculated and each of the three columns is inserted in one of the four corresponding columns in the $[S_V^q]$ matrix for the quadrilateral. For instance the first column of $[S_V^t]$ for triangle IV is inserted in column 2 of the $[S_V^q]$ matrix. Rows four through seven of the $[S_V^q]$ matrix are recalculated to correspond to the sides of the quadrilateral. The resulting matrix for the CSLØT4 element is:

$$[S_V^q] = \begin{bmatrix} \text{First 2 rows} = \frac{1}{4} \sum [S_V^t] \\ \hline \frac{1}{\rho l_{12}} & -\frac{1}{\rho l_{12}} & & & \\ & \frac{1}{\rho l_{23}} & -\frac{1}{\rho l_{23}} & & \\ & & \frac{1}{\rho l_{34}} & -\frac{1}{\rho l_{34}} & \\ -\frac{1}{\rho l_{41}} & & & \frac{1}{\rho l_{41}} & \end{bmatrix} \quad (11)$$

where

$$l_{ij} = \sqrt{(r_j - r_i)^2 + (z_j - z_i)^2}$$

8.16.6 CSLØT1 Element, Phase 2

The data calculated above are extracted from the ESTB data file and the corresponding pressures, p_i , are extracted from the UGV data block. Associated with each vector is a real or complex number. The general equation for velocity in the element is:

$$\{V\} = \frac{1}{\omega} [S_v] \{p_i\} \quad (12)$$

where $\{V\}$ is the vector of velocities in the element

$\omega = \sqrt{|\lambda|}$ (real) in Rigid Format 3 ($\{p_i\}$ is real)

$\omega = 2\pi f$ (real) in Rigid Formats 8 and 11 ($\{p_i\}$ is complex)

$\omega = p$ (complex) in Rigid Formats 7 and 10 ($\{p_i\}$ is complex)

$\omega = 1.0$ (real) in all other Rigid Formats ($\{p_i\}$ is real)

and $[S_v]$ has dimensions 7x3 or 8x4 for the CSLØT3 and CSLØT4 elements respectively

8.17 SOLID POLYHEDRA ELEMENTS, TETRA, WEDGE, HEXA1, HEXA2

These elements define three-dimensional shapes with four points defining a tetrahedron (TETRA), six points defining a wedge (WEDGE), and eight points defining a hexahedron (HEXA1 or HEXA2). Constant strain and stress is assumed in each tetrahedron. The wedge and hexahedron elements are automatically fabricated from tetrahedron elements.

8.17.1 Input Data for the Solid Polyhedra Elements

1. The ECPT entries for the solid elements are:

<u>Symbol</u>	<u>Description</u>
I_d	Element identification number
M	Material identification number
$SIL_i, i = 1, N$	Scalar indices of connected grid points. N = 4, 6, or 8
$CS_i, X_i, Y_i, Z_i \left. \vphantom{\begin{matrix} CS_i, X_i, Y_i, Z_i \\ i = 1, N \end{matrix}} \right\} i = 1, N$	Coordinate system identification number and location in basic coordinates of connected grid points
\bar{T}	Average element temperature

2. Coordinate System Data

The numbers CS_i, X_i, Y_i , and Z_i are used to calculate 3×3 global-to-basic transformation matrices $[T_i]$ for the connected points. Subroutine TRANSD or TRANSS is used.

3. Material Data

Subroutine MAT is used to generate the following material coefficients:

E	Modulus of elasticity
G	Shear modulus
ν	Poisson's ratio
ρ	Mass density
α	Thermal expansion coefficient
T_0	Reference temperature

STRUCTURAL ELEMENT DESCRIPTIONS

8.17.2 Basic Equations for the TETRA Element

1. The matrix which transforms generalized displacements to grid point displacements is $[H_{qu}]$ where

$$[H_{uq}] = \begin{bmatrix} 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \\ 1 & x_4 & y_4 & z_4 \end{bmatrix} \quad (1)$$

This matrix is inverted to produce the matrix $[H_{qu}] = [H_{uq}]^{-1}$ and the determinant, D , of $[H_{uq}]$.

The value of the determinant is checked to see if it is consistent with the determinants of the other tetrahedra being used in a single element.

2. The material coefficients E , G , and ν are used to generate the 6×6 matrix $[G]$ where the nonzero terms are:

$$G_{11} = G_{22} = G_{33} = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)} \quad (2)$$

$$G_{12} = G_{21} = G_{13} =$$

$$G_{31} = G_{23} = G_{32} = \frac{E\nu}{(1+\nu)(1-2\nu)} \quad (3)$$

$$G_{44} = G_{55} = G_{66} = G. \quad (4)$$

3. The four 6×3 matrices $[C^i]$ which transform displacements at points to strains are generated using elements of the H_{qu} matrix: H_{11} , H_{12} , etc. The equation is:

$$[C^i] = \begin{bmatrix} H_{2i} & 0 & 0 \\ 0 & H_{3i} & 0 \\ 0 & 0 & H_{4i} \\ 0 & H_{4i} & H_{3i} \\ H_{3i} & 0 & H_{1i} \\ H_{2i} & H_{1i} & 0 \end{bmatrix} \quad (5)$$

8.17.3 Stiffness Matrix Generations for the TETRA Element (Subroutine KTETRA of Module SMA1)

The 3 x 3 partition of the element stiffness matrix (in global coordinates) connecting points i and j is:

$$[K_{ij}] = [T_i]^T [C^i]^T [G] [C^j] [T_j],$$

where $[T_i]$ and $[T_j]$ are the 3 x 3 global-to-basic transformation matrices. The matrices are produced for point j corresponding to the pivot point in the matrix assembly process.

8.17.4 Mass Matrix Generation for the TETRA Element (Subroutine MSOLID of Module SMA2)

The mass matrix for each point of the tetrahedron is formed as a 6 x 6 matrix and inserted on the diagonal of the overall mass matrix. Its equation is:

$$M_{ii} = \begin{bmatrix} m/4 & & & & & \\ & m/4 & & & & \\ & & m/4 & & & \\ & & & 0 & & \\ -0- & & & & 0 & \\ & & & & & 0 \end{bmatrix}, \quad (6)$$

where $m = 1/6 \rho |D|$. (D is the determinant of the $[H_{uq}]$ matrix.)

8.17.5 Thermal Load Generation for the TETRA Element (Subroutine TETRA of Module SSG1)

The 3 x 1 thermal load vector $\{P_i\}$ for point i of the tetrahedron is:

STRUCTURAL ELEMENT DESCRIPTIONS

$$\{P_i\} = [T_i]^T [C_i]^T [G] \{\epsilon_t\} , \quad (7)$$

$$\text{where } \{\epsilon_t\} = \begin{Bmatrix} \alpha \\ \alpha \\ \alpha \\ 0 \\ 0 \\ 0 \end{Bmatrix} (\bar{T} - T_0) , \quad (8)$$

and $\bar{T} = 1/4 (T_1 + T_2 + T_3 + T_4)$ is the average temperature of the four connected points, given in data block GPTT.

8.17.6 Stress Calculations for the TETRA Elements (Subroutines SSØLID1 and SSØLID2 of Module SPR2)

The stress is calculated in two phases. Phase I is used to calculate the transformation matrices between displacements and temperatures to stresses. Phase II uses the actual displacements and temperatures to calculate stresses.

1. In Phase I, the following calculations are performed:

$$[S_i] = [G] [C_i] [T_i] \quad i = 1, 2, 3, 4, \quad (9)$$

$$\{S_t\} = [G] \{\alpha\} , \quad (10)$$

where

$$\{\alpha\} = \begin{Bmatrix} \alpha \\ \alpha \\ \alpha \\ 0 \\ 0 \\ 0 \end{Bmatrix} . \quad (11)$$

2. In Phase II, the 3×1 displacement vector, $\{u_i\}$, for each point, i , is extracted from the $\{u_q\}$ displacement vector and the average temperature, \bar{T} , is extracted from the GPTT data block. The stresses are calculated as follows:

$$\begin{pmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{yz} \\ \tau_{xy} \\ \tau_{xy} \end{pmatrix} = \sum_{i=1}^4 [S_i] \{u_i\} - \{S_t\} (\bar{T} - T_0). \quad (12)$$

The hydrostatic pressure, P , and the octahedral shear stress, τ_0 , are calculated by the equations:

$$P = -1/3 (\sigma_x + \sigma_y + \sigma_z), \quad (13)$$

$$\tau_0 = 1/3 [2\sigma_x (\sigma_x - \sigma_y - \sigma_z) + 2\sigma_y (\sigma_y - \sigma_z) + 2\sigma_z^2 + 6 (\tau_{yz}^2 + \tau_{xz}^2 + \tau_{xy}^2)]^{1/2} \quad (14)$$

8.17.7 Basic Equations for the WEDGE, HEXA1, and HEXA2 Elements

The wedge element is connected to six grid points and is divided into four tetrahedron sub-elements. The connected points assigned to each tetrahedron are:

<u>TETRA Number</u>	<u>Connected Points</u>			
I	1	2	3	6
II	1	2	6	5
III	1	4	5	6

The HEXA1 and HEXA2 elements are connected to eight grid points and are subdivided into five tetrahedra for the HEXA1 element and ten overlapping tetrahedra for the HEXA2 element. The connected points original to each tetrahedron are:

STRUCTURAL ELEMENT DESCRIPTIONS

<u>Subelement Number</u>		<u>Connected Points</u>			
HEXA1	HEXA2				
I	I	1	2	3	6
II	II	1	3	4	8
III	III	1	3	8	6
IV	IV	1	5	6	8
V	V	3	6	7	8
	VI	2	3	4	7
	VII	1	2	4	5
	VIII	2	4	5	7
	IX	2	5	6	7
	X	4	5	7	8

The basic procedure used with these elements is to extract the data associated with each tetrahedron subelement and go to the tetrahedron calculations. In subroutine KSØLID of Module SMA1, the tetrahedron calculations and matrix insertion is done by calling subroutine KTETRA. In subroutine MSØLID of Module SMA2, the tetrahedron calculations and insertion are done in an internal subroutine. In subroutine SØLID of Module SSG1, the tetrahedron subroutine STETRA is used to calculate and invert the thermal loads. In subroutines SSØLID1 and SSØLID2 of Module SDR2, the tetrahedron calculations are done with an internal subroutine and the results are summed together to produce average stresses.

8.17.8 Stiffness Matrix Calculations and Geometry Checks for the WEDGE, HEXA1, and HEXA2 Elements (Subroutine KSØLID of Module SMA1)

With these elements, the order of the connections and the resulting geometry is critical for reasonable results. Three basic criteria must be met:

1. If the connections are correct, the calculated volumes for all tetrahedron subelements will be consistent. When the subroutine is called for the first time for each element, all of the tetrahedra are processed to produce the signs of the determinants of the $[H_{uq}]$ matrices. The signs must be either all positive or all negative, or an error is indicated.

SOLID POLYHEDRA ELEMENTS, TETRA, WEDGE, HEXA1, HEXA2

2. The order of the connected points is checked by calculating the normal vectors to the top and bottom faces assuming a right-hand rule. The normal vectors must not have a negative scalar product.
3. The wedge has three quadrilateral faces and the hexagonal elements have six quadrilateral faces. Subroutine KPLTST is used to check these faces. The points must not deviate from being a plane by more than 10 percent.

Wedge

Face Number	Points on Face
1	1, 2, 5, 4
2	1, 4, 6, 3
3	2, 3, 6, 5

Hexahedron

Face Number	Points on Face
1	1, 2, 3, 4
2	1, 2, 6, 5
3	2, 3, 7, 6
4	3, 4, 8, 7
5	4, 1, 5, 8
6	5, 6, 7, 8

In the KSOLID subroutine, each element is tested for geometric consistency when the pivot point equals the first connected point. In any event, the ECPT data is converted to the TETRA format for as many times as necessary, and subroutine KTETRA is called each time. If a HEXA2 element is being processed, a flag is set, so the KTETRA subroutine will divide the stiffness of each tetrahedron by two.

8.17.9 Mass Matrix Generation for the WEDGE, HEXA1 and HEXA2 Elements (Subroutine MSOLID of Module SMA2)

The mass calculations involve the calculation of the total mass of each tetrahedron in the element and assigning one-fourth of the mass to each of the four points. If a HEXA2 element is used, the mass of each tetrahedron is divided by two.

STRUCTURAL ELEMENT DESCRIPTIONS

8.17.10 Thermal Load Generation for the WEDGE, HEXA1 and HEXA2 Elements (Subroutine SØLID of Module SSG2)

This subroutine arranges the ECPT data into the TETRA format for each tetrahedron in the element. Subroutine TETRA is called each time to calculate the thermal loads and insert them in the load vector. If a HEXA2 element is used, a flag is set, so that the TETRA routine will divide the results by two.

8.17.11 Stress Data Recovery for the WEDGE, HEXA1 and HEXA2 Elements (Subroutines SSØLID1 and SSØLID2 of Module SDR2)

The first phase of stress recovery involves the calculation of displacement-stress matrices $[S_{ie}]$ and the temperature stress vector $\{S_{te}\}$. A 6×3 $[S_{ie}]$ matrix is generated for each connected point. Its equation is:

$$[S_{ie}] = \frac{1}{N} \sum_{\alpha=1}^N [S_i]_{\alpha} \quad (15)$$

where $[S_i]_{\alpha}$ is the matrix corresponding to tetrahedron number α associated with point i , and N is the total number of tetrahedra in the element. The $[S_i]_{\alpha}$ matrices are described in Section 8.17.6. As each tetrahedron is processed, the four $[S_i]$ matrices and the $\{S_t\}$ vector are added to the appropriate $[S_{ie}]$ matrices for the whole element. The TETRA element is also processed by this code with $N = 1$. The thermal stress vectors are added by the equation:

$$\{S_{te}\} = \frac{1}{N} \sum_{\alpha=1}^N \{S_t\}_{\alpha} \quad (16)$$

In Phase II of stress recovery, the logic is given in Section 8.17.6. The code is identical for all elements, with the only difference being the number of connected grid points.

8.17.12 Thermal Analysis Calculations for the Solid Elements

The "stiffness" matrix terms for the solid elements are generated by subroutines KSØLID and KTETRA of module SMA1. All of the solid elements (TETRA, WEDGE, HEXA1, and HEXA2) use the KTETRA subroutine to calculate and insert the final matrix terms. For thermal analysis, the following operations are performed:

1. The geometry is processed and the matrix $[H_{uq}]$ and the determinant, D , are produced for either structure or thermal analysis. See Section 8.17.3.
2. For thermal analysis, subroutine HMAT is used with INFLAG = 3 to produce the following data:

$$K_{xx} \ K_{xy} \ K_{xz} \ K_{yy} \ K_{yz} \ K_{zz}$$

3. The material matrix $[G_e]$ is calculated where:

$$[G_e] = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & K_{xx} & K_{xy} & K_{xz} \\ 0 & K_{xy} & K_{yy} & K_{yz} \\ 0 & K_{xz} & K_{yz} & K_{zz} \end{bmatrix}$$

4. The matrix terms associated with the pivot point (j) are:

$$\begin{aligned} & \{K_{j1} \ K_{j2} \ K_{j3} \ K_{j4}\}^T \\ &= \frac{D}{6} \{H_{uq}^j\}^T [G_e] [H_{uq}]. \end{aligned}$$

The vector $\{H_{uq}^j\}$ is the column of the $[H_{uq}]$ matrix associated with point j. The values K_{ji} are inserted in the KGG matrix in column number = SIL_j , and row number = SIL_i .

The "mass" matrix for the heat transfer analysis is generated by subroutine MSØLID of module SMA2. The thermal capacity coefficient, C_p , is returned from subroutine HMAT. Each subelement tetrahedron is used to calculate terms, which are placed in the BGG matrix. For each point, i, on each tetrahedron, the value is:

$$B_{ii} = \frac{C_p \cdot |D|}{6},$$

where D is the determinant of the $[H_{uq}]$ matrix. Results for the HEXA2 element are divided by 2.

STRUCTURAL ELEMENT DESCRIPTIONS

The "stress" data recovery for heat transfer analysis is performed by subroutines SDHTF1, SDHTFF, and SDHTF2 of module SDR2. In Phase 1, the matrix K and the matrix C are calculated where K is the 3 x 3 material matrix and C_e is a 3 by number of points matrix. For the tetrahedron:

$$[C_e] = \begin{bmatrix} H_{21} & H_{22} & H_{23} & H_{24} \\ H_{31} & H_{32} & H_{33} & H_{34} \\ H_{41} & H_{42} & H_{43} & H_{44} \end{bmatrix}$$

For the WEDGE, HEXA1, and HEXA2 elements, the [C] matrix is calculated for each subelement. The column corresponding to each point of the tetrahedron is added to the column of the C_e matrix corresponding to that point in the whole element. The results are divided by the number of subelements.

In Phase 2, the temperature gradient vector and flux vector are calculated with the equations:

$$\{\Delta T\} = [C] \{u\}$$

$$\{q\} = -[K]\{\Delta T\},$$

where $\{u\}$ is the vector of temperatures of the connected points.

THE HBDY ELEMENTS

8.18 THE HBDY ELEMENTS

8.18.1 Input Data

The boundary condition element HBDY produces matrix terms only in a heat transfer analysis. The following data will be needed from the ECPT table.

<u>Symbol</u>	<u>Description</u>
ID	Element ID
IFLAG	Element Type Flag
$SIL_i, i = 1, 2 \dots 8$	Scalar indices
V_1, V_2, V_3	Orientation vector
Mat Id	Material identification number for MAT4 or MAT5 data
AF	Area factor
ϵ	Emissivity coefficient
α	Absorbtivity coefficient
R_1, R_2	Radii of elliptic cylinder
C_{Si}, X_i, Y_i, Z_i $i = 1, 2 \dots 8$	Coordinate systems and location coordinates of connected grid points; 1 - 4 are element points, 5 - 8 are points in fluid
To	Average element temperature for initial estimate

The meaning of the data for various values of IFLAG are:

IFLAG	Element Type	N= Number of Gridpoints	AF
1	point	1	area
2	line	2	width
3	revolution	2	---
4	triangle	3	---
5	quadrilateral	4	---
6	elliptic cylinder	2	circumference

STRUCTURAL ELEMENT DESCRIPTIONS

8.18.2 Stiffness Matrix Calculations (Subroutine KHB DY of Module SMA1)

For the revolution elements $x_i > 0$ and $y_i = 0$, otherwise there is illegal geometry. The matrix produced will be $N \times N$. The material coefficient, H , is extracted from the material data block with $INFLAG = 1$, and T_0 as the average temperature. The $[C]$ matrix for each element type is given in the following table:

IFLAG	C
1	$H(AF)$
2 or 6	$\frac{H(AF)\ell}{6} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$
3	$\frac{H(2\pi)\ell}{12} \begin{bmatrix} (3x_1 + x_2) & (x_1 + x_2) \\ (x_1 + x_2) & (x_1 + 3x_2) \end{bmatrix}$
4	$\frac{Ha}{24} \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}$
5	<p>or</p> $\frac{H}{48} \begin{bmatrix} 2(a_2 + a_3 + a_4) & (a_3 + a_4) & (a_2 + a_4) & (a_2 + a_3) \\ & 2(a_1 + a_3 + a_4) & (a_1 + a_4) & (a_1 + a_3) \\ & & 2(a_1 + a_2 + a_4) & (a_1 + a_2) \\ & & & 2(a_1 + a_2 + a_3) \end{bmatrix}$ <p>where</p> $C_{ij} = \frac{H}{48} [(1 + \delta_{ij}) (a_1 + a_2 + a_3 + a_4) - (a_i + a_j)],$ $\delta_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$

The length ℓ appears only when $N = 2$ or 6 , and:

$$\ell = [(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2]^{1/2}.$$

The factor a is twice the area of a triangle ($N = 3$).

THE HBDY ELEMENTS

Let

$$\vec{r}_i = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix}$$

$$a = |(\vec{r}_2 - \vec{r}_1) \times (\vec{r}_3 - \vec{r}_2)|$$

The factors a_i are two times the area of the triangle which does not touch vertex i of a quadrilateral.

$$a_1 = |(\vec{r}_3 - \vec{r}_2) \times (\vec{r}_4 - \vec{r}_3)|$$

$$a_2 = |(\vec{r}_4 - \vec{r}_3) \times (\vec{r}_1 - \vec{r}_4)|$$

$$a_3 = |(\vec{r}_1 - \vec{r}_4) \times (\vec{r}_2 - \vec{r}_1)|$$

$$a_4 = |(\vec{r}_2 - \vec{r}_1) \times (\vec{r}_3 - \vec{r}_2)|$$

The C matrices are placed in the "stiffness" matrix using the following matrix equation:

$$\{P\} = [K] \{u\},$$

or:

$$\begin{pmatrix} P_1 \\ \vdots \\ P_N \\ \hline P_5 \\ \vdots \\ P_{4+N} \end{pmatrix} = \begin{bmatrix} & & & \\ & C & & -C \\ & & \vdots & \\ & & & \\ \hline -C & & & C \end{bmatrix} \begin{pmatrix} u_1 \\ \vdots \\ u_N \\ \hline u_5 \\ \vdots \\ u_{4+N} \end{pmatrix}$$

In the octimal code, only the columns corresponding to a "pivot point" u_i are generated. The terms K_{ji} are inserted for every point u_i in the element SIL list. The terms are placed in the SIL_i column of the matrix.

8.18.3 Capacity Matrix Calculations for the HBDY Element (Subroutine MHBDY of Module SMA2)

The heat capacity matrix is calculated much like a mass matrix and is placed in matrix BGG in the SMA2 module. The material coefficient C_p is extracted from the material tables using subroutine HMAT with INFLAG = 5. The scalar terms for each point are:

STRUCTURAL ELEMENT DESCRIPTIONS

IFLAG	B
1	$B_{11} = AF \cdot Cp$
2 or 6	$B_{11} = B_{22} = \frac{AF \cdot Cp \cdot \ell}{2}$
	$B_{11} = \frac{\pi Cp \ell}{3} (2x_1 + x_2)$
3	$B_{22} = \frac{\pi Cp \ell}{3} (2x_2 + x_1)$
4	$B_{11} = B_{22} = B_{33} = \frac{a_1 \cdot Cp}{6}$
5	$B_{ii} = \frac{(a_T - a_i) \cdot Cp}{12} \quad i = 1, 4$

where $a_T = a_1 + a_2 + a_3 + a_4$ and the a values are defined in the previous section.

One-half the value of the terms are placed in the BGG matrix in positions row = column = SIL_i . If the corresponding boundary layer point SIL_{i+4} is nonzero, the value, $\frac{1}{2}B_{ii}$, is also placed in the row and column corresponding to SIL_{i+4} .

8.18.4 Convective Heat Flux Recovery for the HBDY Element (Subroutines SDHTF1, SDHTFF, and SDHTF2 in Module SDR2)

Phase 1

Subroutine SDHTF1 rearranges the EST data and calls the HMAT routine for material property, H. The output of Phase 1 consists of the following data:

<u>Word</u>	<u>Description</u>
1	Element Id
2-9	Scalar indices of connected points
10,11	BCD words for "HBDY"
12	Number of generalized coordinates (=1)
13	NP - Number of connected points (=8)
14	The value of H
22-29	The eight C values

THE HBDY ELEMENTS

Subroutine SDHTFF calculates the nonzero values of C and the value K with the following equations

$$\text{IFLAG} = 1$$

$$C_1 = -C_5 = 1.0$$

$$K = A_f H$$

$$\text{IFLAG} = 2 \text{ or } 6$$

$$C_1 = C_2 = -C_5 = -C_6 = \frac{1}{2}$$

$$K = \frac{A_f \cdot \ell \cdot H}{2}$$

$$\text{IFLAG} = 3 \quad C_1 = \frac{1}{3\bar{r}} (2x_1 + x_2) = -C_5$$

$$C_2 = \frac{1}{3\bar{r}} (2x_2 + x_1) = -C_6$$

$$K = \pi \bar{r} \ell H,$$

$$\text{where } \bar{r} = \frac{1}{2} (x_1 + x_2)$$

$$\text{IFLAG} = 4 \quad C_1 = C_2 = C_3 = -C_5 = -C_6 = -C_7 = \frac{1}{3}$$

$$K = AH,$$

$$\text{where } A = \frac{1}{2} |(\bar{r}_2 - \bar{r}_1) \times (\bar{r}_3 - \bar{r}_1)|$$

$$\text{IFLAG} = 5 \quad C_1 = C_2 = C_3 = C_4 = -C_5 = -C_6 = C_7 = -C_8 = \frac{1}{4}$$

$$K = A_q H, \text{ where } A_q = \frac{1}{2} \sum (\text{area of each triangle})$$

Phase 2

Subroutine SDHTF2 calculates the temperature difference and total heat flow across the boundary layer with the equation

$$\Delta T = \{C\}^T \{u\}$$

$$q = -K \Delta T,$$

STRUCTURAL ELEMENT DESCRIPTIONS

where K and $\{C\}$ are the Phase 1 outputs and $\{u\}$ is the vector of temperatures given in the UGV vector in the SIL^{th} positions.

8.18.5 Area Factor Calculations for the HBDY Element (Utility Subroutine HBDY)

For purposes of radiation calculations, thermal load generation, and data recovery, the HBDY elements are processed by subroutine HBDY. The input data consists of the EST entry for each element. The output is dependent on the option as follows:

a) Option = 1

<u>Word</u>	<u>Symbol</u>	<u>Description</u>
1	ID	Element ID
2	A_T	Total area
3	ϵ	Emissivity
4	Not used	
5-8	SIL_i	Scalar indices
9-12	G_i	Fraction of total area associated with each point

b) Option = 2

<u>Word</u>	<u>Symbol</u>	<u>Description</u>
1	ID	Element ID
2	A_T	Total area
3-6	SIL_i	Scalar indices
7-10	A_i	Area factors
11-13	\bar{n}_1	Normal unit vector
14-16	\bar{n}_2	Second normal vector for IFLAG = 6 only

The output quantities are calculated as follows:

$$IFLAG = 1$$

$$\bar{n}_1 = \frac{\bar{v}}{|\bar{v}|}$$

$$A_T = AF$$

$$A_i = AF$$

THE HBDY ELEMENTS

$$\text{IFLAG} = 2 \quad \vec{R} = \begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \\ z_2 - z_1 \end{pmatrix}$$

$$\ell = |\vec{R}|$$

$$A_T = AF \cdot \ell$$

$$A_1 = A_2 = \frac{1}{2} A_T$$

$$\vec{Z} = \vec{V} - \frac{(\vec{R} \cdot \vec{V})}{\ell^2} \vec{R}$$

$$\vec{n}_1 = \frac{\vec{Z}}{|\vec{Z}|}$$

$$\text{IFLAG} = 3$$

$$\Delta r = r_2 - r_1$$

$$\Delta z = z_2 - z_1$$

$$\ell = \sqrt{\Delta r^2 + \Delta z^2}$$

$$A_1 = \frac{\pi \ell}{3} (2r_1 + r_2)$$

$$A_2 = \frac{\pi \ell}{3} (2r_2 + r_1)$$

$$A_T = A_1 + A_2$$

$$\vec{n}_1 = \frac{1}{\ell} \begin{pmatrix} \Delta z \\ 0 \\ -\Delta r \end{pmatrix}$$

$$\text{IFLAG} = 4$$

$$\vec{D}_1 = \vec{x}_2 - \vec{x}_1$$

$$\vec{D}_2 = \vec{x}_3 - \vec{x}_1$$

$$\vec{Z} = \vec{D}_1 \times \vec{D}_2$$

$$A_T = \frac{1}{2} |\vec{Z}|$$

$$\vec{n} = \frac{\vec{Z}}{|\vec{Z}|}$$

STRUCTURAL ELEMENT DESCRIPTIONS

$$A_1 = A_2 = A_3 = \frac{1}{3} A_T$$

IFLAG = 5

$$\bar{D}_{12} = \bar{r}_2 - \bar{r}_1$$

$$\bar{D}_{13} = \bar{r}_3 - \bar{r}_1$$

$$\bar{D}_{24} = \bar{r}_4 - \bar{r}_2$$

$$\bar{Z} = \bar{D}_{13} \times \bar{D}_{24}$$

$$A_T = \frac{1}{2} |\bar{Z}|$$

$$\bar{n}_1 = \frac{\bar{Z}}{|\bar{Z}|}$$

$$A_{123} = \frac{1}{2} |\bar{D}_{12} \times \bar{D}_{13}|$$

$$A_{412} = \frac{1}{2} |\bar{D}_{24} \times \bar{D}_{12}|$$

$$A_1 = \frac{1}{6} (A_T + A_{412})$$

$$A_2 = \frac{1}{6} (A_T + A_{213})$$

$$A_3 = \frac{1}{6} (2A_T - A_{412})$$

$$A_4 = \frac{1}{6} (2A_T - A_{213})$$

IFLAG = 6

$$\bar{x} = \bar{r}_2 - \bar{r}_1$$

$$\ell = |\bar{x}|$$

$$A_T = A_f \cdot \ell - \text{Option 1}$$

or $A_T = \ell - \text{Option 2}$

$$A_1 = A_2 = \frac{1}{2} A_T$$

$$\bar{y} = \bar{v} \times \bar{x}$$

THE HBDY ELEMENTS

$$\bar{z} = \bar{x} \times \bar{y}$$

$$n_1 = \frac{\bar{z}}{|\bar{z}|} \cdot R_2$$

$$n_2 = \frac{\bar{y}}{|\bar{y}|} \cdot R_1$$

Note for Option 1, the factors G_i are

$$G_i = \frac{A_i}{A_T}$$

8.19 QDMEM1 ISOPARAMETRIC QUADRILATERAL ELEMENT

8.19.1 Input Data for QDMEM1 Element

1. ECPT entries for QDMEM1 are:

<u>Symbol</u>	<u>Description</u>
EID	Element identification number
$\left. \begin{array}{l} \text{SIL}_1 \\ \text{SIL}_2 \\ \text{SIL}_3 \\ \text{SIL}_4 \end{array} \right\}$	Scalar indices of the connected grid points
θ	
Mat ID	
h	
μ	Nonstructural mass per unit area
$\left. \begin{array}{l} N_i \\ X_i \\ Y_i \\ Z_i \end{array} \right\} i = 1, 4$	Local coordinate system numbers and location coordinates in the basic system for the connected grid points
\bar{t}	
	Temperature of element

2. Coordinate system data

The numbers N_i , X_i , Y_i , and Z_i are used to calculate 3 by 3 basic-to-global coordinate transformation matrices $[T_i]$ for points $i = 1, 2, 3$, and 4.

3. Material Data

<u>Symbol</u>	<u>Description</u>
$[G]$	3x3 Stress-strain matrix
ρ	Mass density
$\alpha_x, \alpha_y, \alpha_{xy}$	Thermal expansion coefficients
t_0	Reference temperature
g_e	Structural damping coefficient
$\sigma_t, \sigma_c, \sigma_s$	Stress limits for tension, compression, and shear

8.19.2 Basic Equations for QDMEM1

1. Calculation of unit vectors in mean plane.

The following equations are used to calculate the three unit vectors in the mean plane, $\{i\}$, $\{j\}$, and $\{k\}$ (see figure 1), which define the element coordinate system.

$$\{V_i\} = \begin{Bmatrix} X_i \\ Y_i \\ Z_i \end{Bmatrix}, \quad i = 1, 2, 3, 4 \quad (1)$$

The diagonals are:

$$\{d_1\} = \{V_3\} - \{V_1\} \quad (2)$$

$$\{d_2\} = \{V_4\} - \{V_2\} \quad (3)$$

The normal to the plane is calculated from:

$$\{k\} = \frac{\{d_1\} \times \{d_2\}}{|\{d_1\} \times \{d_2\}|} \quad (4)$$

$$\{a_1\} = \{V_2\} - \{V_1\} \quad (5)$$

$$\hat{h} = \frac{1}{2} \{a_1\}^T \{k\} \quad (6)$$

The vectors lying in the mean plane are computed from:

$$\{i\} = \frac{\{a_1\} - 2\hat{h}\{k\}}{|\{a_1\} - 2\hat{h}\{k\}|} \quad (7)$$

$$\{j\} = \{k\} \times \{i\} \quad (8)$$

2. The displacement transformation matrix from basic coordinates to in-plane coordinates is:

$$[E]^T = \begin{bmatrix} \bar{E}^T & 0 & 0 & 0 \\ 0 & \bar{E}^T & 0 & 0 \\ 0 & 0 & \bar{E}^T & 0 \\ 0 & 0 & 0 & \bar{E}^T \end{bmatrix} \quad (9)$$

where

$$[\bar{E}]^T = \begin{bmatrix} i_1 & i_2 & i_3 \\ j_1 & j_2 & j_3 \\ k_1 & k_2 & k_3 \end{bmatrix} \quad (10)$$

3. The coordinates or difference of coordinates of the points in the element coordinate system are:

$$x_{12} = x_1 - x_2 = -\{a_1\}^T \{i\} \quad (11)$$

$$x_{13} = x_1 - x_3 = -\{d_1\}^T \{i\} \quad (12)$$

$$x_{24} = x_2 - x_4 = -\{d_2\}^T \{i\} \quad (13)$$

$$x_{14} = x_{12} + x_{24} \quad (14)$$

$$x_{23} = x_{13} - x_{12} \quad (15)$$

$$x_{34} = x_{14} - x_{13} \quad (16)$$

$$y_3 = \{d_1\}^T \{j\} \quad (17)$$

$$y_4 = \{d_2\}^T \{j\} \quad (18)$$

$$y_{34} = y_3 - y_4 \quad (19)$$

STRUCTURAL ELEMENT DESCRIPTIONS

4. The transformation of displacements at the user-specified grid points from the displacements at the projected points in the mean plane is given by the 12×8 matrix $[B]$. The nonzero elements of this matrix are given by:

$$B(1,1) = B(2,2) = B(4,3) = B(5,4) = B(7,5) = B(8,6) = B(10,7) = B(11,8) = 1 \quad (20)$$

$$B(3,1) = \frac{-\bar{h}}{\ell_a} \quad (21)$$

$$B(3,2) = \bar{h} \left(\frac{-1}{\ell_d \sin \theta_1} + \frac{\cot \theta_1}{\ell_a} \right) \quad (22)$$

$$B(3,3) = -B(3,1) \quad (23)$$

$$B(3,4) = \bar{h} \left(\frac{\cot \theta_2}{\ell_a} \right) \quad (24)$$

$$B(3,7) = \bar{h} \left(\frac{\cos \theta_{42}}{\ell_d \Delta_2} \right) \quad (25)$$

$$B(3,8) = \bar{h} \left(\frac{\sin \theta_{42}}{\ell_d \Delta_2} \right) \quad (26)$$

$$B(6,1) = -B(3,1) \quad (27)$$

$$B(6,2) = -\bar{h} \left(\frac{\cot \theta_1}{\ell_a} \right) \quad (28)$$

$$B(6,3) = B(3,1) \quad (29)$$

$$B(6,4) = \bar{h} \left(\frac{-\cot \theta_2}{\ell_a} + \frac{1}{\ell_b \sin \theta_2} \right) \quad (30)$$

$$B(6,5) = \bar{h} \left(\frac{-\sin \theta_{31}}{\ell_b \Delta_1} \right) \quad (31)$$

$$B(6,6) = \bar{h} \left(\frac{-\cos \theta_{31}}{\ell_b \Delta_1} \right) \quad (32)$$

$$B(9,4) = \bar{h} \left(\frac{-1}{\ell_b \sin \theta_2} \right) \quad (33)$$

$$B(9,5) = \bar{h} \left(\frac{\sin \theta_{31}}{\ell_b \Delta_1} + \frac{\cos \theta_{32}}{\ell_c \Delta_1} \right) \quad (34)$$

$$B(9,6) = \bar{h} \left(\frac{\cos \theta_{31}}{\ell_b \Delta_1} + \frac{\sin \theta_{32}}{\ell_c \Delta_1} \right) \quad (35)$$

$$B(9,7) = \bar{h} \left(\frac{-\sin \theta_{41}}{\ell_c \Delta_2} \right) \quad (36)$$

$$B(9,8) = \bar{h} \left(\frac{\cos \theta_{41}}{\ell_c \Delta_2} \right) \quad (37)$$

$$B(12,2) = \bar{h} \left(\frac{1}{\ell_d \sin \theta_1} \right) \quad (38)$$

$$B(12,5) = \bar{h} \left(\frac{-\cos \theta_{32}}{\ell_c \Delta_1} \right) \quad (39)$$

$$B(12,6) = \bar{h} \left(\frac{-\sin \theta_{32}}{\ell_c \Delta_1} \right) \quad (40)$$

$$B(12,7) = \bar{h} \left(\frac{-\cos \theta_{42}}{\ell_d \Delta_2} + \frac{\sin \theta_{41}}{\ell_c \Delta_2} \right) \quad (41)$$

$$B(12,8) = \bar{h} \left(\frac{-\sin \theta_{42}}{\ell_d \Delta_2} - \frac{\cos \theta_{41}}{\ell_c \Delta_2} \right) \quad (42)$$

where

$$\bar{h} = -\hat{h} \quad (43)$$

$$\ell_a = -x_{12} \quad (44)$$

STRUCTURAL ELEMENT DESCRIPTIONS

$$l_b = \sqrt{x_{23}^2 + y_3^2} \quad (45)$$

$$l_c = \sqrt{x_{34}^2 + y_{34}^2} \quad (46)$$

$$l_d = \sqrt{x_{14}^2 + y_4^2} \quad (47)$$

$$\sin \theta_1 = \sin \theta_{41} = \frac{y_4}{l_d} \quad (48)$$

$$\cos \theta_1 = \cos \theta_{41} = \frac{-x_{14}}{l_d} \quad (49)$$

$$\sin \theta_2 = \cos \theta_{32} = \frac{y_3}{l_b} \quad (50)$$

$$\cos \theta_2 = \sin \theta_{32} = \frac{x_{23}}{l_b} \quad (51)$$

$$\sin \theta_{31} = \cos \theta_{42} = -\frac{y_{34}}{l_c} \quad (52)$$

$$\cos \theta_{31} = \sin \theta_{42} = \frac{x_{34}}{l_c} \quad (53)$$

$$\Delta_1 = \cos \theta_{31} \cos \theta_{32} - \sin \theta_{31} \sin \theta_{32} \quad (54)$$

$$\Delta_2 = \cos \theta_{41} \cos \theta_{42} + \sin \theta_{41} \sin \theta_{42} \quad (55)$$

See Figure 2 for characteristic lengths and angles of quadrilateral membrane in mean plane.

8.19.3 Stiffness Matrix Calculation for QDMEM1 (Subroutine KQDMM1 of Module SMA1)

1. Element interior angle tests.

The interior angles of the quadrilateral in the mean plane are checked to verify that they are less than 180° . The following tests are used:

<u>Test</u>	<u>Point with angle greater than 180°</u>
If $y_4 < 0$	1
If $y_3 < 0$	2
If $x_{14} < x_{12} + x_{23} \frac{y_4}{y_3}$	3
If $x_{13} > \frac{y_3}{y_4} x_{14}$	4

2. The stiffness matrix in the mean plane in terms of the displacements in the mean plane is calculated first as

$$[U]_{8 \times 8} = \int_0^1 \int_0^1 [A]^T [G] [A] J d\xi d\eta \quad (56)$$

where $[G]$ is the 3×3 stress-strain matrix,

$$J = -x_{12}y_4 - x_{12}y_{34} \xi + (y_3x_{14} - y_4x_{23})\eta \quad (57)$$

and $[A]$, a 3×8 strain-displacement matrix is represented in the following manner.

$$[A]_{3 \times 8} = \frac{1}{J} \begin{bmatrix} (a_1 + b_1\eta + c_1\xi) & (a_4 + b_4\eta + c_4\xi) & . & . & . & (a_{22} + b_{22}\eta + c_{22}\xi) \\ (a_2 + b_2\eta + c_2\xi) & . & . & . & . & (a_{23} + b_{23}\eta + c_{23}\xi) \\ (a_3 + b_3\eta + c_3\xi) & . & . & . & . & (a_{24} + b_{24}\eta + c_{24}\xi) \end{bmatrix} \quad (58)$$

In the expression for the matrix $[A]$,

$$a_1 = -y_4 \quad b_1 = y_3 \quad c_1 = -y_{34} \quad (59)$$

$$a_2 = 0 \quad b_2 = 0 \quad c_2 = 0 \quad (60)$$

STRUCTURAL ELEMENT DESCRIPTIONS

$$a_3 = -x_{24}$$

$$a_4 = 0$$

$$a_5 = -x_{24}$$

$$a_6 = -y_4$$

$$a_7 = y_4$$

$$a_8 = 0$$

$$a_9 = x_{14}$$

$$a_{10} = 0$$

$$a_{11} = x_{14}$$

$$a_{12} = y_4$$

$$a_{13} = 0$$

$$a_{14} = 0$$

$$a_{15} = 0$$

$$a_{16} = 0$$

$$a_{17} = 0$$

$$a_{18} = 0$$

$$a_{19} = 0$$

$$a_{20} = 0$$

$$a_{21} = -x_{12}$$

$$a_{22} = 0$$

$$a_{23} = -x_{12}$$

$$a_{24} = 0$$

$$b_3 = x_{23}$$

$$b_4 = 0$$

$$b_5 = x_{23}$$

$$b_6 = y_3$$

$$b_7 = -y_4$$

$$b_8 = 0$$

$$b_9 = -x_{14}$$

$$b_{10} = 0$$

$$b_{11} = -x_{14}$$

$$b_{12} = -y_4$$

$$b_{13} = y_4$$

$$b_{14} = 0$$

$$b_{15} = x_{14}$$

$$b_{16} = 0$$

$$b_{17} = x_{14}$$

$$b_{18} = y_4$$

$$b_{19} = -y_3$$

$$b_{20} = 0$$

$$b_{21} = -x_{23}$$

$$b_{22} = 0$$

$$b_{23} = -x_{23}$$

$$b_{24} = -y_3$$

$$c_3 = x_{34} \quad (61)$$

$$c_4 = 0 \quad (62)$$

$$c_5 = x_{34} \quad (63)$$

$$c_6 = -y_{34} \quad (64)$$

$$c_7 = y_{34} \quad (65)$$

$$c_8 = 0 \quad (66)$$

$$c_9 = -x_{34} \quad (67)$$

$$c_{10} = 0 \quad (68)$$

$$c_{11} = -x_{34} \quad (69)$$

$$c_{12} = y_{34} \quad (70)$$

$$c_{13} = 0 \quad (71)$$

$$c_{14} = 0 \quad (72)$$

$$c_{15} = -x_{12} \quad (73)$$

$$c_{16} = 0 \quad (74)$$

$$c_{17} = -x_{12} \quad (75)$$

$$c_{18} = 0 \quad (76)$$

$$c_{19} = 0 \quad (77)$$

$$c_{20} = 0 \quad (78)$$

$$c_{21} = x_{12} \quad (79)$$

$$c_{22} = 0 \quad (80)$$

$$c_{23} = x_{12} \quad (81)$$

$$c_{24} = 0 \quad (82)$$

QDMEM1 ISOPARAMETRIC QUADRILATERAL ELEMENT

The approach used to evaluate the stiffness matrix [U] is to identify each term in the triple product (equation 56) prior to the integration. Toward this end, the 8 x 8 matrix [U] is thought of as being constructed of sixteen 2 x 2 submatrices, each identified by subscripts (i,j) as in the following table

	j = 1	j = 2	j = 3	j = 4
i = 1				
i = 2				
i = 3				
i = 4				

2 x 2 Partition
(Typical)

The subscript i indicates a horizontal slice of 2 rows of the matrix and the subscript j indicates a vertical slice of 2 columns of the matrix. Further if the row identifier, k, and column identifier, l, are introduced to locate an element within any particular 2 x 2 submatrix (k and l assume values of 1 and 2), then a general term in the U matrix is located by specification of i, j, k, and l. Thus, when advantage is taken of some zero terms in the matrix A,

$$U_{ijkl} = h \int_0^1 \int_0^1 \frac{d + en + f\xi + h^*n\xi + pn^2 + q\xi^2}{J} d\xi dn \quad (83)$$

$$d = G_{k1} a_{k1} a_{l1} + G_{k3} a_{k1} a_{l2} + G_{31} a_{k2} a_{l1} + G_{33} a_{k2} a_{l2} \quad (84)$$

$$e = G_{k1} (a_{k1} b_{l1} + b_{k1} a_{l1}) + G_{k3} (a_{k1} b_{l2} + b_{k1} a_{l2}) + G_{31} (a_{k2} b_{l1} + b_{k2} a_{l1}) + G_{33} (a_{k2} b_{l2} + b_{k2} a_{l2}) \quad (85)$$

$$f = G_{k1} (a_{k1} c_{l1} + c_{k1} a_{l1}) + G_{k3} (a_{k1} c_{l2} + c_{k1} a_{l2}) + G_{31} (a_{k2} c_{l1} + c_{k2} a_{l1}) + G_{33} (a_{k2} c_{l2} + c_{k2} a_{l2}) \quad (86)$$

$$\begin{aligned}
 h^* = & G_{k\ell}(b_{k_1}c_{\ell_1} + c_{k_1}b_{\ell_1}) + G_{k3}(b_{k_1}c_{\ell_2} + c_{k_1}b_{\ell_2}) \\
 & + G_{3\ell}(b_{k_2}c_{\ell_1} + c_{k_2}b_{\ell_1}) + G_{33}(b_{k_2}c_{\ell_2} + c_{k_2}b_{\ell_2})
 \end{aligned} \quad (87)$$

$$p = G_{k\ell}b_{k_1}b_{\ell_1} + G_{k3}b_{k_1}b_{\ell_2} + G_{3\ell}b_{k_2}b_{\ell_1} + G_{33}b_{k_2}b_{\ell_2} \quad (88)$$

$$q = G_{k\ell}c_{k_1}c_{\ell_1} + G_{k3}c_{k_1}c_{\ell_2} + G_{3\ell}c_{k_2}c_{\ell_1} + G_{33}c_{k_2}c_{\ell_2} \quad (89)$$

where additional subscripts are identified as

$$k_1 = 6(i-1) + 4(k-1) + 1 \quad (90)$$

$$k_2 = 6(i-1) + 3(k-1) + 3 \quad (91)$$

$$\ell_1 = 6(j-1) + 4(\ell-1) + 1 \quad (92)$$

$$\ell_2 = 6(j-1) + 3(\ell-1) + 3 \quad (93)$$

3. Gaussian Quadrature

The evaluation of equation (83) is carried out by Gaussian quadrature utilizing a 4 x 4 grid. Thus

$$U_{ijkl} = \frac{h}{4} \sum_{m=0}^3 \sum_{n=0}^3 \bar{a}_m \bar{a}_n \left[\frac{d + e\eta_n + f\xi_m + h^*\eta_n\xi_m + p\eta_n\eta_n + q\xi_m\xi_m}{J(\eta_n, \xi_m)} \right] \quad (94)$$

where

$$\xi_0 = \eta_0 = -\frac{1}{2} \sqrt{\left(\frac{30 + \sqrt{480}}{70}\right)} + \frac{1}{2} = .069431844 \quad (95)$$

$$\xi_1 = \eta_1 = -\frac{1}{2} \sqrt{\left(\frac{30 - \sqrt{480}}{70}\right)} + \frac{1}{2} = .330009478 \quad (96)$$

$$\xi_2 = \eta_2 = \frac{1}{2} \sqrt{\left(\frac{30 - \sqrt{480}}{70}\right)} + \frac{1}{2} = .669990522 \quad (97)$$

$$\xi_3 = \eta_3 = \frac{1}{2} \sqrt{\left(\frac{30 + \sqrt{480}}{70}\right)} + \frac{1}{2} = .930568156 \quad (98)$$

$$\bar{a}_0 = \bar{a}_3 = .347854845 \quad (99)$$

$$\bar{a}_1 = \bar{a}_2 = .652145155 \quad (100)$$

4. The 3 x 3 stiffness matrix partition $[k_{ij}]$ in global coordinates is then given by

$$[k_{ij}] = [T_i]^T ([E][B][U][B]^T[E]^T)_{ij} [T_j] \quad (101)$$

where i corresponds to the current pivot number and assumes values 1, 2, 3, 4;
 j corresponds to one of the connected grid points and assumes values 1, 2, 3, 4;
 $([E][B][U][B]^T[E]^T)_{ij}$ is the appropriate 3 x 3 partition of the 12 x 12 matrix
 $[E][B][U][B]^T[E]^T$.

5. For use in the overall structural matrix, the matrices are expanded to 6 x 6 to form:

$$[K_{ij}] = \begin{bmatrix} k_{ij} & | & 0 \\ \hline 0 & | & 0 \end{bmatrix} \quad (102)$$

8.19.4 Mass Matrix Generation for the QDMEM1 (Subroutine MASSTQ of Module SMA2)

The mass matrix for the isoparametric element is calculated in the same manner as QDMEM element which subdivides the quadrilateral into two triangles. See Section 8.4.11 page 8 of the Programmer's Manual.

8.19.5 Element Load Calculations for the QDMEM1 Element (Subroutine QDMEM1 of Module SSG1)

The first step in the calculation of the thermal load vector is to compute the following matrix

STRUCTURAL ELEMENT DESCRIPTIONS

$$[C]_{8 \times 3} = h \int_0^1 \int_0^1 [A]^T J d\xi d\eta \quad (103)$$

The integrations in equation (103) can be performed in closed form and results in the following

$$C_{11} = -\frac{hy_4}{2} \quad (104)$$

$$C_{12} = 0 \quad (105)$$

$$C_{13} = -\frac{hx_{24}}{2} \quad (106)$$

$$C_{21} = 0 \quad (107)$$

$$C_{22} = -\frac{hx_{24}}{2} \quad (108)$$

$$C_{23} = -\frac{hy_4}{2} \quad (109)$$

$$C_{31} = \frac{hy_3}{2} \quad (110)$$

$$C_{32} = 0 \quad (111)$$

$$C_{33} = \frac{hx_{13}}{2} \quad (112)$$

$$C_{41} = 0 \quad (113)$$

$$C_{42} = \frac{hx_{13}}{2} \quad (114)$$

$$C_{43} = \frac{hy_3}{2} \quad (115)$$

$$C_{51} = \frac{hy_4}{2} \quad (116)$$

$$C_{52} = 0 \quad (117)$$

$$C_{53} = \frac{hx_{24}}{2} \quad (118)$$

$$C_{61} = 0 \quad (119)$$

$$C_{62} = \frac{hx_{24}}{2} \quad (120)$$

$$C_{63} = \frac{hy_4}{2} \quad (121)$$

$$C_{71} = -\frac{hy_3}{2} \quad (122)$$

$$C_{72} = 0 \quad (123)$$

$$C_{73} = -\frac{hx_{13}}{2} \quad (124)$$

$$C_{81} = 0 \quad (125)$$

$$C_{82} = -\frac{hx_{13}}{2} \quad (126)$$

$$C_{83} = -\frac{hy_3}{2} \quad (127)$$

Given the temperature of the element, \bar{t} , either directly or from the temperature of the four grid points t_1 , t_2 , t_3 , and t_4 in the GPTT data block, the routine generates force vectors by the equation:

$$\{P\}_i = [T_i]^T ([E][B][C][G]\{\alpha\})_i (\bar{t} - t_0) \quad (128)$$

where $([E][B][C][G]\{\alpha\})_i$ is the appropriate 3×1 subvector of the 12×1 vector $[E][B][C][G]\{\alpha\}$,

$$\{\alpha\} = \begin{Bmatrix} \alpha_x \\ \alpha_y \\ \alpha_{xy} \end{Bmatrix} \quad (129)$$

STRUCTURAL ELEMENT DESCRIPTIONS

and t_0 is the reference or stress-free temperature of the material. The forces are placed in the PG load vector data block.

8.19.6 Element Stress Calculations for the QDMEM1 Element (Subroutines SQDM11 and SQDM12 of Module SDR2)

1. Calculations performed in SQDM11 (Phase 1 calculations).

(a) The stresses in the QDMEM1 elements are calculated at the intersection of the diagonals identified by ξ^* , η^* . For parallelograms

$$\xi^* = 1/2 \quad (130)$$

$$\eta^* = 1/2 \quad (131)$$

For other geometries, first x^* and y^* must be found as

$$x^* = \frac{y_4 x_{13} x_{12}}{y_3 x_{24} - y_4 x_{13}} \quad (132)$$

$$y^* = - \frac{y_4 (x^* + x_{12})}{x_{24}} \quad (133)$$

provided $x_{24} \neq 0$. In the event that $x_{13} = 0$, a simplification results and

$$x^* = - x_{13} = 0 \quad (134)$$

$$y^* = - \frac{y_4}{x_{24}} x_{12} \quad (135)$$

For the situation where $x_{24} = 0$,

$$x^* = - x_{12} \quad (136)$$

$$y^* = \frac{y_3 x_{12}}{x_{13}} \quad (137)$$

After the values of x^* and y^* are obtained, in the situation where $x_{34} \neq -x_{12}$,

$$\eta^* = - \frac{b^* \pm \sqrt{(b^*)^2 - 4a^*x_{12}y^*}}{2a^*} \quad (138)$$

where the \pm sign is chosen such that $0 < \eta^* < 1$ and

$$b^* = -y_4x_{12} + c^* \quad (139)$$

$$a^* = -y_4x_{23} + y_3x_{14} \quad (140)$$

$$c^* = y_{34}x^* - y^*(x_{34} + x_{12}) \quad (141)$$

Should $a^* = 0$, then

$$\eta^* = \frac{-x_{12}y^*}{b^*} \quad (142)$$

Further, if $y_{34} \neq 0$,

$$\xi^* = \frac{-c^* + (y_4x_{23} - y_3x_{14})\eta^*}{y_{34}x_{12}} \quad (143)$$

If $y_{34} = 0$,

$$\xi^* = \frac{x^* + x_{14}\eta^*}{[\eta^*(x_{34} + x_{12}) - x_{12}]} \quad (144)$$

For the situation where $x_{34} = -x_{12}$ but $y_{34} \neq 0$

$$\eta^* = \frac{-\hat{b} \pm \sqrt{(\hat{b})^2 + 4\hat{a}x_{12}y^*}}{2\hat{a}} \quad (145)$$

where the \pm sign is chosen such that $0 < \eta^* < 1$ and

$$\hat{b} = x_{12}y_4 - y_{34}x^* \quad (146)$$

$$\hat{a} = -x_{14}y_{34} \quad (147)$$

Should $\hat{a} = 0$, then

$$\eta^* = \frac{x_{12}y^*}{\hat{b}} \quad (148)$$

After ξ^* is evaluated, then

$$\xi^* = \frac{y^* - y_4\eta^*}{y_{34}\eta^*} \quad (149)$$

(b) The values of ξ^* and η^* are substituted into matrix $[A]$ and the transformations from displacements to stress are then given by

$$[S_i] = ([G][A][B]^T[E]^T)_i [T_i] \quad i = 1, 2, 3, 4 \quad (150)$$

where $([G][A][B]^T[E]^T)_i$ is the appropriate 3×3 partition of the 3×12 matrix $[G][A][B]^T[E]^T$. The temperature-stress relation is

$$\{S_t\} = -[G]\{\alpha\} \quad (151)$$

2. Calculations performed by SQDM12 (Phase 2 calculations)

The equation for stress is

$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \sigma_{xy} \end{Bmatrix} = \sum_{i=1}^4 [S_i] \{u_{g_i}\} + \{S_t\}(\bar{t} - t_0) \quad (152)$$

The principal stresses are:

$$\sigma_1 = \frac{\sigma_x + \sigma_y}{2} + \sqrt{\left(\frac{\sigma_x - \sigma_y}{2}\right)^2 + \sigma_{xy}^2} \quad (153)$$

$$\sigma_2 = \frac{\sigma_x + \sigma_y}{2} - \sqrt{\left(\frac{\sigma_x - \sigma_y}{2}\right)^2 + \sigma_{xy}^2} \quad (154)$$

$$\phi_1 = \frac{1}{2} \tan^{-1} \left(\frac{2\sigma_{xy}}{\sigma_x - \sigma_y} \right) \text{ (converted to degrees)} \quad (155)$$

where ϕ_1 is limited to $-90^\circ \leq \phi_1 \leq 90^\circ$.

QDMEM1 ISOPARAMETRIC QUADRILATERAL ELEMENT

The maximum shear stress τ is given by

$$\tau = \sqrt{\left(\frac{\sigma_x - \sigma_y}{2}\right)^2 + \sigma_{xy}^2} \quad (156)$$

STRUCTURAL ELEMENT DESCRIPTIONS

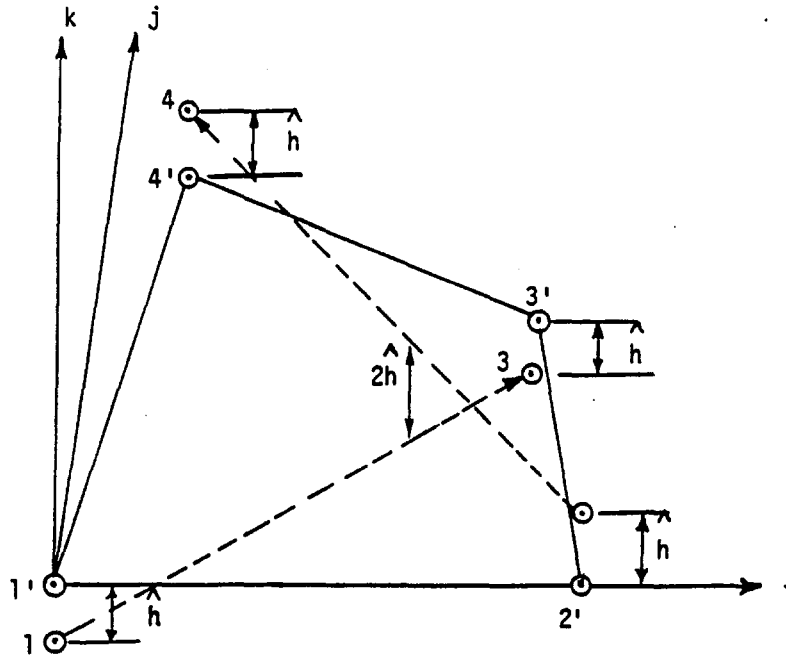


Figure 1. Mean plane for quadrilateral membrane element. (Actual grid points are indicated by unprimed numbers and projection of grid points onto mean plane are indicated by primed numbers.)

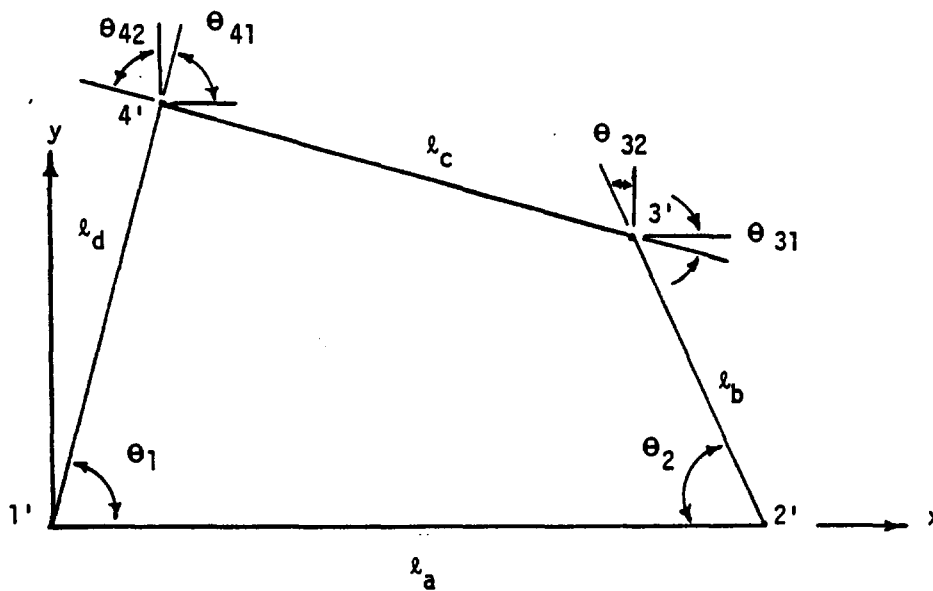


Figure 2. Characteristic lengths and angles for quadrilateral membrane element in mean plane.

8.20 THE QDMEM2 ELEMENT

8.20.1 Input Data for the QDMEM2 Element

1. The ECPT/EST data block contents are:

<u>Symbol</u>	<u>Description</u>
EID	Element identification number
SIL _i , i = 1,2,3,4	Scalar indices of the connected grid points
θ	Anisotropic material orientation angle
t	Thickness
μ	Nonstructural mass per unit area
N _i , X _i , Y _i , Z _i , i = 1,2,3,4	BGPDT data giving local coordinate system number and basic coordinate system locations for the connected grid points
T _m	Temperature for material properties

2. Material data:

The material subroutine, MAT, returns the following data

<u>Symbol</u>	<u>Description</u>
G ₁₁ , G ₁₂ , G ₁₃ G ₂₂ , G ₂₃ , G ₃₃	Elements of 3x3 stress-strain matrix in material coordinates
ρ	Mass density
{ α }	3x1 thermal expansion vector
T ₀	Reference temperature
G _e	Structural damping coefficient
$\sigma_t, \sigma_c, \sigma_s$	Stress limits

3. Coordinate system data

For each connected grid point, i, the BGPDT data is used to calculate a 3x3 global-to-basic transformation matrix, [T_i].

STRUCTURAL ELEMENT DESCRIPTIONS

8.20.2 Basic Calculations for the QDMEM2 Element (Subroutine Q2BCS)

1. The element coordinate system is defined by the following equations:

$$\{D_{13}\} = \begin{pmatrix} x_3 - x_1 \\ y_3 - y_1 \\ z_3 - z_1 \end{pmatrix}, \quad (1)$$

$$\{D_{24}\} = \begin{pmatrix} x_4 - x_2 \\ y_4 - y_2 \\ z_4 - z_2 \end{pmatrix}, \quad (2)$$

$$\{N\} = \{D_{13}\} \times \{D_{24}\}, \quad (3)$$

$$A = \frac{1}{2} |\{N\}|, \quad (4)$$

$$\{k\} = \frac{1}{|\{N\}|} \{N\}, \quad (5)$$

$$\{D_{12}\} = \begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \\ z_2 - z_1 \end{pmatrix}, \quad (6)$$

$$h = \frac{1}{2} \{D_{12}\} \cdot \{k\}, \quad (7)$$

$$\{x\} = \{D_{12}\} - 2h\{k\}, \quad (8)$$

$$\{i\} = \frac{1}{|\{x\}|} \{x\}, \quad (9)$$

$$\{j\} = \{k\} \times \{i\}, \quad (10)$$

THE QDMEM2 ELEMENT

The unit vectors are stored in the transformation matrix E as follows:

$$[E] = \begin{bmatrix} \{i\}^T \\ \{j\}^T \\ \{k\}^T \end{bmatrix} \quad (11)$$

If $h^2/A < \epsilon_p$, a flag is set indicating the element lies approximately in a plane.

2. The locations of the points in element coordinates are:

$$\{r_1\} = \begin{pmatrix} 0 \\ 0 \\ -h \end{pmatrix}, \quad (12)$$

$$\{r_i\} = [E] \begin{pmatrix} x_i - x_1 \\ y_i - y_1 \\ z_i - z_1 \end{pmatrix} + \{r_1\}, \quad (13)$$

$$i = 2, 3, 4, 5,$$

where

$$\begin{pmatrix} x_5 \\ y_5 \\ z_5 \end{pmatrix} = \frac{1}{4} [\{x_1\} + \{x_2\} + \{x_3\} + \{x_4\}]. \quad (14)$$

8.20.3 Subtriangle Calculations for the QDMEM2 Element (Subroutine Q2TRMS)

1. Input data:

<u>Stiffness</u>	<u>Thermal Load</u>	<u>Stress Recovery</u>
a,b	a,b	a,b
[r]	[r]	[r]
[G _e]	[G _e]	[G _e]
cosθ _m , sinθ _m	cosθ _m , sinθ _m	cosθ _m , sinθ _m
t	t	t
	{α _e }(T-T ₀)	{α _e }

STRUCTURAL ELEMENT DESCRIPTIONS

where

- $[r]$ is a 5x3 matrix, each row is the location of a point in element coordinates.
- a, b are pointers to the two connected points in the $[r]$ matrix. The fifth row is always the third point.
- $[G_e]$ is the 3x3 material matrix in material coordinates defined in /MATOUT/.
- $\cos\theta_m, \sin\theta_m$ are the orientation angle cosine and sine.
- t is the element thickness.
- α_e is the thermal expansion coefficient in material coordinates, given in /MATOUT/.
- $(T-T_0)$ is the temperature of the triangle.

2. Output data:

<u>Stiffness</u>	<u>Thermal Load</u>	<u>Stress Recovery</u>	
$[K_{ca}]$	$[K_{ca}]$	$[K_{ca}]$	$\{p_a^t\}$
$[K_{cb}]$	$[K_{cb}]$	$[K_{cb}]$	$\{p_b^t\}$
$[K_{cc}]$	$[K_{cc}]$	$[K_{cc}]$	$\{p_c^t\}$
$[K_{aa}]$	$\{P_a\}$	$[S_a]$	$\{Z_a\}$
$[K_{ab}]$	$\{P_b\}$	$[S_b]$	$\{Z_b\}$
$[K_{ba}]$	$\{P_c\}$	$[S_c]$	$\{Z_c\}$
$[K_{bb}]$		$[K_{aa}]$	
		$[K_{ab}]$	
		$[K_{ba}]$	
		$[K_{bb}]$	

3. The equations to produce this data are:

$$\{V_{12}\} = \{r_b\} - \{r_a\},$$

$$\{V_{13}\} = \{r_c\} - \{r_b\},$$

THE QDMEM2 ELEMENT

$$\{N\} = \{V_{12}\} \times \{V_{13}\},$$

$$\{k\} = \frac{1}{|\{N\}|} \{N\},$$

$$A = \frac{1}{2} |\{N\}|,$$

$$\{i\} = \frac{1}{|\{V_{12}\}|} \{V_{12}\},$$

$$\{j\} = \{k\} \times \{i\},$$

$$[E_2] = \begin{bmatrix} \{i\}^T \\ \{j\}^T \end{bmatrix},$$

$$x_b = |\{V_{12}\}|,$$

$$x_c = \{i\}^T \{V_{13}\},$$

$$y_c = \{j\}^T \{V_{13}\},$$

$$c_1 = \frac{1}{x_b} c_2 = \frac{x_c}{x_b} c_3 = \frac{1}{y_c} c_4 = c_3(c_2 - 1)$$

$$[C_a] = \begin{bmatrix} -c_1 & 0 \\ 0 & c_4 \\ c_4 & -c_1 \end{bmatrix}$$

$$[C_b] = \begin{bmatrix} c_1 & 0 \\ 0 & -c_2 c_3 \\ -c_2 c_3 & c_1 \end{bmatrix}$$

$$[C_c] = \begin{bmatrix} 0 & 0 \\ 0 & c_3 \\ c_3 & 0 \end{bmatrix}$$

STRUCTURAL ELEMENT DESCRIPTIONS

The material orientation cosine (C_m) and sine (S_m) are calculated by the equations

$$\begin{Bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{Bmatrix} = \{r_b - r_a\},$$

$$l = \sqrt{\Delta x^2 + \Delta y^2},$$

$$C_m = (\cos\theta_m \Delta x + \sin\theta_m \Delta y) / l,$$

$$S_m = (\sin\theta_m \Delta x - \cos\theta_m \Delta y) / l.$$

The material orientation is used to transform anisotropic materials,

$$[T^m] = \begin{bmatrix} C_m^2 & S_m^2 & C_m S_m \\ S_m^2 & C_m^2 & -C_m S_m \\ -2C_m S_m & 2C_m S_m & (C_m^2 - S_m^2) \end{bmatrix}.$$

The equation for each stiffness matrix partition is

$$[K_{ij}] = At [H_i]^T [G_e] [H_j],$$

where

$$[H_i] = [T^m] [C_i] [E_2],$$

$$i = a, b, c.$$

The equation for the thermal loads is

$$\{P_i\} = At [H_i]^T [G_e] \{\alpha_e\} (T - T_0),$$

$$i = a, b, c.$$

The equations for the stress matrices are

$$[S_i] = [G_e] [H_i],$$

$$\{p_i^t\} = At [H_i]^T [G_e] \{\alpha_e\},$$

$$\{Z_i\}^T = t [T_3^m]^T [S_i],$$

THE QDMEM2 ELEMENT

where $\{T_3^m\}$ is the third column of the $[T^m]$ matrix,

$$i = a, b, c.$$

8.20.4 Stiffness Matrix Calculations for the QDMEM2 Element (Subroutine KQDM2S of Module SMA1)

1. The material properties in the material coordinates are calculated by calling subroutine MAT with $\theta = 0$ degrees.
2. The area of core for storing element stiffness matrix partitions is set to zero, and the basic element calculations are done with subroutine Q2BCS, which returns the following data

$\{r_i\}$, $i = 1, \dots, 5$ The locations of the five points in element coordinates.

$[E]$ The 3x3 basic-to-element coordinate system transformation.

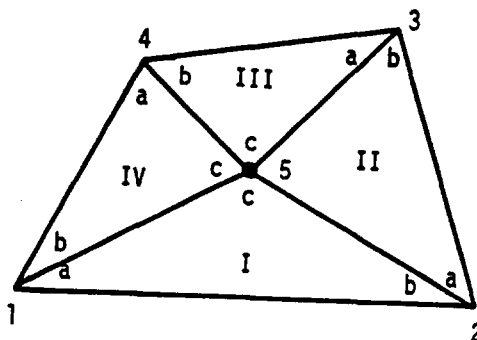
PFLAG Flag indicating whether or not the element approximates a plane.

3. The four subtriangles are processed with subroutine Q2TRMS. Each triangle, I , is given three points defined by the mapping matrix where

$$\text{Point number} = M_{I\alpha}, \quad \begin{matrix} \alpha = 1, 2, 3 \\ I = 1, 2, 3, 4, \end{matrix}$$

$$[M] = \begin{bmatrix} 1 & 2 & 5 \\ 2 & 3 & 5 \\ 3 & 4 & 5 \\ 4 & 1 & 5 \end{bmatrix}.$$

The orientation of the triangles is shown below



4. Each triangle produces seven matrices,

$$[K_{5a}], [K_{5b}], [K_{5c}], [K_{aa}], [K_{ab}], [K_{ba}], [K_{bb}].$$

These are added to the appropriate matrices for the entire element

$$[K_{5i}] \quad i = 1, \dots, 5,$$

$$[K_{ij}] \quad i = 1, \dots, 4, j = 1, \dots, 4.$$

5. The displacement of the center point will be constrained by the equation

$$\{u_5\} = \sum_i [G_i] \{u_i\}.$$

If the element is not planar,

$$G_i = -[K_{55}]^{-1} [K_{5i}]$$

If the element is planar,

$$\bar{K}_i = [K_{5i}],$$

with the third rows modified as follows:

$$[\bar{K}_i] = \begin{bmatrix} x & x & x \\ x & x & x \\ 0 & 0 & -.25 \end{bmatrix} \quad i \neq 5$$

$$[\bar{K}_5] = \begin{bmatrix} x & x & x \\ x & x & x \\ 0 & 0 & 1.0 \end{bmatrix} \quad i = 5$$

The equation for G_i is

$$[G_i] = -[\bar{K}_5]^{-1} [\bar{K}_i].$$

6. The "pivot" point, p , is found and the constrained matrices, $[K_{ij}^e]$, are produced.

If the element is not planar,

$$[K_{pj}^e] = [K_{pj}] + [K_{5p}]^T [G_j].$$

THE QDMEM2 ELEMENT

If the element is planar,

$$[K_{pj}^e] = [K_{pj}] + [K_{5p}]^T [G_j] + [G_p]^T [K_{5j}] + [G_p]^T [K_{55}] [G_j],$$

(p = pivot point, j = 1,2,3,4).

7. The matrices are transformed to global coordinates and output as double precision 6x6 partitions,

$$[K_{pj}^g] = ([E] [T_p])^T [K_{pj}] ([E] [T_j]).$$

Output:

$$[K_{pj}^g] = \begin{bmatrix} K_{pj}^g & 0 \\ 0 & 0 \end{bmatrix}.$$

8. If the structural damping matrix, $[K_{gg}^h]$, is being produced, the damping matrices, $[K_{pj}^h]$ are also output where

$$[K_{pj}^h] = G_e [K_{pj}^0].$$

8.20.5 Mass Matrix Generation

Subroutine MASSTQ of module SMA2 is used to generate a lumped mass matrix. The element is treated exactly like a QDMEM element.

8.20.6 Thermal Loads

The basic calculations indicated in Sections 8.20.2 and 8.20.4 are performed. The average temperature, T, is given in the GPTT data. The material routine, MAT, is called to produce the $\{\alpha_e\}$ vector and the reference temperature, T_0 . The following product is passed to the Q2TRMS subroutine,

$$\{d\} = \{\alpha_e\} (T - T_0).$$

The Q2TRMS subroutine will return the three 3x1 load vectors:

$$\{P_a\}, \{P_b\}, \text{ and } \{P_c\}.$$

It will also return the following 3x3 matrices,

$$[K_{5a}], [K_{5b}], [K_{5c}].$$

The loads are added directly to the five load vectors corresponding to the five connected grid points, $\{P_1\}$, $\{P_2\}$, $\{P_3\}$, $\{P_4\}$, $\{P_5\}$.

The $[K_{5i}]$ matrices are added to the five 3x3 matrices,

$$[K_{51}], [K_{52}], [K_{53}], [K_{54}], [K_{55}].$$

When all triangles have been processed, the center point load is eliminated by the equation

$$\{P_i^e\} = \{\bar{P}_i\} + [G_i^T] \{\bar{P}_5\} \quad i=1,2,3,4,$$

where the 3x3 matrix, $[G]$, is calculated as in step 5, Section 8.20.4.

The resulting loads are transformed to global coordinates by the equation

$$\{P_i^g\} = [T]^T [E]^T \{P_i^e\}.$$

8.20.7 QDMEM2 Element Stress and Force Calculations (Subroutines SQDM21 and SQDM22 of Module SDR2)

Phase 1

1. In the Phase 1 calculations, the basic calculations indicated in Sections 8.20.2 and 8.20.4 are performed, and the material data is extracted with subroutine MAT. The subroutine is called for in each triangle. The output will be

$$[K_{ij}] \quad i = a,b,c \text{ and } j = a,b,c$$

Nine stiffness matrices in element coordinates (3x3).

$$[S_a], [S_b], [S_c].$$

Three stress matrices (3x3).

$$\{P_a^t\}, \{P_b^t\}, \{P_c^t\}$$

Three stress temperature vectors (3x1).

$$\{Z_a\}, \{Z_b\}, \{Z_c\}$$

Three edge force-displacement vectors.

2. This data is added to the appropriate positions in the data corresponding to all the connected points of the element. This data is

$$[R_{ij}^e] \quad i = 1,2,3,4 \text{ and } j = 1,2,3,4$$

Sixteen matrices (3x3).

$$[R_{5i}^e] \quad i = 1,2,3,4,5$$

Five matrices (3x3).

$$[S_i^e] \quad i = 1,2,3,4,5$$

Five matrices (3x3).

$$\{P_i^t\} \quad i = 1,2,3,4,5$$

Five vectors (3x1).

3. The $\{Z_a\}$, $\{Z_b\}$, and $\{Z_c\}$ vectors are stored in five 4x3 $[R]$ matrices. For triangle No. I, the data vector, $\{Z_a\}$, is stored in row I of the matrix corresponding to point a.
4. The constraints on the center point are applied with the following logic. The matrix, $[G]$, is calculated as in step 5 of Section 8.20.4.
- The equations for the reduced data are as follows:

If the planar flag is "off,"

$$[K_{ij}^e] = [R_{ij}^e] + [R_{5i}^e]^T [G_j].$$

If the planar flag is "on,"

$$[K_{ij}^e] = [R_{ij}^e] + [R_{5i}^e]^T [G_j] + [G_i]^T [R_{5j}^e] + [G_i]^T [R_{55}^e] [G_j].$$

For either case,

$$[S_i^e] = \frac{1}{2} [S_i^e] + [S_5^e] [G_i],$$

$$[R_i] = [R_i] + [R_5] [G_i],$$

$$\{P_i^t\} = \{P_i^t\} + [G_i]^T \{P_5^t\}.$$

STRUCTURAL ELEMENT DESCRIPTIONS

5. The data is transformed to global coordinates by the equations

$$[K_{ij}^g] = [Q_i] [K_{ij}^e] [E] [T_j],$$

$$\{P_i^g\} = [Q_i] \{P_i^t\},$$

$$[S_i^g] = [S_i^e] [E] [T_i],$$

$$[R_i^g] = [R_i] [E] [T_i],$$

where the $[Q]$ matrices necessary to transform corner forces to directions parallel to the sides are calculated as follows:

$$\{d_1\} = \{r_2\} - \{r_1\}$$

$$\{d_2\} = \{r_3\} - \{r_2\}$$

$$\{d_3\} = \{r_4\} - \{r_3\}$$

$$\{d_4\} = \{r_1\} - \{r_4\}$$

$$\{k_1\} = - \{d_1\} \times \{d_4\}$$

$$\{k_2\} = - \{d_2\} \times \{d_1\}$$

$$\{k_3\} = - \{d_3\} \times \{d_2\}$$

$$\{k_4\} = - \{d_4\} \times \{d_3\}$$

$$\{\alpha_i\} = \frac{1}{|\{d_i\}|} \{d_i\},$$

$$\{\delta_i\} = \frac{1}{|\{k_i\}|} \{k_i\},$$

$$[Q_i] = [-\{\alpha_i\} \mid \{\alpha_{i-1}\} \mid \{\delta_i\}]^{-1},$$

$$i = 1, 2, 3, 4, \quad (\{\alpha_0\} = \{\alpha_4\}).$$

THE QDMEM2 ELEMENT

6. The following 250 words of data is saved on the ESTB data block

<u>Symbol</u>	<u>Description</u>
ID	Element identification
SIL_i $i=1,2,3,4$	Connected point SIL values
t	Thickness
T_0	Reference temperature
$[K_{ij}^g]$	Sixteen point force versus displacement matrices (3x3)
$[S_i^g]$	Four stress matrices (3x3)
$\{P_i^t\}$	Four point force versus temperature vectors
$\{S_t\}$	One stress temperature vector = $[G_e] \{\alpha_e\}$
$[R_i]$	Four side force matrices (4x3)

Phase 2

The displacement vectors for the connected grid points, $\{u_i\}$, are extracted from open core. The temperature of the element, \bar{T} , is given in the GPTT data block. If "STRESS" for the element is requested, the following data is calculated:

$$\begin{pmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{pmatrix} = \sum_{i=1}^4 [S_i] \{u_i\} - \{S_t\} (\bar{T} - T_0).$$

The stress output is identical to the QDMEM element output.

If "ELFORCE" output is requested, the following data is calculated:

$$\{F^i\} = \sum_{j=1}^4 [K_{ij}] \{u_j\} - \{P_i^t\} (T - T_0).$$

STRUCTURAL ELEMENT DESCRIPTIONS

The in-plane forces are:

$$\begin{array}{cccc} f_1 = F_2^1 & f_3 = F_2^2 & f_5 = F_2^3 & f_7 = F_2^4 \\ f_2 = F_1^1 & f_4 = F_1^2 & f_6 = F_1^3 & f_8 = F_1^4 \end{array}$$

The "kick" forces are:

$$K_i = F_3^i, \quad i=1,2,3,4.$$

The "shear" forces are:

$$\begin{pmatrix} q_1 \\ -q_2 \\ q_3 \\ -q_4 \end{pmatrix} = \sum [R_i] \{u_i\}.$$

THE IHEX1, IHEX2, AND IHEX3 ELEMENTS

8.21 THE IHEX1, IHEX2, AND IHEX3 ELEMENTS

8.21.1 Input Data for the IHEXi Elements

1. The EST entries for IHEXi are:

<u>Symbol</u>	<u>Description</u>
$SIL_i, i=1,2,\dots,NGP$	Scalar indices of the connected grid points. NGP = number of grid points per element
N_i X_i Y_i Z_i	$i = 1,2,\dots,NGP$ { Referenced local coordinate system and location in basic coordinates of connected grid points.
MID	Material identification number.
CID	Coordinate system identification number in which anisotropic material is defined.
NIP	Number of integration points in each coordinate direction.
AR	Maximum aspect ratio of element
ALFA	Maximum angle (in degrees) for face normals
BETA	Maximum angle (in degrees) for mid-edge points
$GPT_i, i=1,2,\dots,NGP$	Grid Point temperatures

2. Coordinate System Data

The numbers N_i , X_i , Y_i , and Z_i are used to calculate the three by three global-to-basic coordinate transformation matrix $[T_i]$ for the i^{th} grid point via calls to either TRANSS or TRANS D.

The CID is used to calculate the six by six global-to-basic coordinate transformation matrices $[T^M]$ for anisotropic material. If the CID references a cylindrical or spherical coordinate system, $[T^M]$ must be computed at each integration point. Otherwise, it need be computed once only for the element.

STRUCTURAL ELEMENT DESCRIPTIONS

3. Material Data

<u>Symbol</u>	<u>Description</u>
$[G_m]$	Six by six stress-strain matrix defined in coordinate system CID
ρ	Mass density
$\{\alpha_m\}$	Vector of six thermal expansion coefficients defined in coordinate system CID
T_0	Reference temperature
g_e	Structural damping coefficient

8.21.2 Basic Equations for IHEXi Elements

1. Numerical Integration

The method of Gaussian Quadrature is used to numerically integrate the element matrices. The weighting coefficients H_i and abscissas s_i are listed in Table 1 at the conclusion of this Section 8.23.7.

2. Element Coordinates

The coordinates of the grid points in element coordinates are:

For the IHEX1 element:

i	ξ_i	η_i	ζ_i
1	-1	-1	-1
2	1	1	-1
3	1	1	-1
4	-1	1	-1
5	-1	-1	1
6	1	-1	1
7	1	1	1
8	-1	1	1

THE IHEX1, IHEX2, AND IHEX3 ELEMENTS

For the IHEX2 element:

i	ξ_i	η_i	ζ_i
1	-1	-1	-1
2	0	-1	-1
3	1	-1	-1
4	1	0	-1
5	1	1	-1
6	0	1	-1
7	-1	1	-1
8	-1	0	-1
9	-1	-1	0
10	1	1	0
11	1	1	0
12	-1	1	0
13	0	-1	1
14	0	-1	1
15	1	0	1
16	1	0	1
17	1	1	1
18	0	1	1
19	-1	1	1
20	-1	0	1

STRUCTURAL ELEMENT DESCRIPTIONS

For the IHEX3 element:

i	ξ_i	η_i	ζ_i
1	-1	-1	-1
2	-1/3	-1	-1
3	1/3	-1	-1
4	1	-1	-1
5	1	-1/3	-1
6	1	1/3	-1
7	1	1	-1
8	1/3	1	-1
9	-1/3	1	-1
10	-1	1	-1
11	-1	1/3	-1
12	-1	-1/3	-1
13	-1	-1	-1/3
14	1	-1	-1/3
15	1	1	-1/3
16	-1	1	-1/3
17	-1	-1	1/3
18	1	-1	1/3
19	1	1	1/3
20	-1	1	1/3
21	-1	-1	1
22	-1/3	-1	1
23	1/3	-1	1
24	1	-1	1
25	1	-1/3	1
26	1	1/3	1
27	1	1	1
28	1/3	1	1

THE IHEX1, IHEX2, AND IHEX3 ELEMENTS

29	-1/3	1	1
30	-1	1	1
31	-1	1/3	1
32	-1	-1/3	1

3. Interpolating Functions and Their Derivatives

The interpolating function, or isoparametric shape function N_i and the derivative of this function with respect to the element's coordinates is given for the i^{th} grid point by the following tables: 2, 3 and 4.

4. Jacobian Matrix

The Jacobian matrix at any point (x,y,z) within the element is:

$$[J] = \begin{bmatrix} \frac{\partial N_1}{\partial \xi} & \frac{\partial N_2}{\partial \xi} & \dots & \frac{\partial N_{\text{NGP}}}{\partial \xi} \\ \frac{\partial N_1}{\partial \eta} & \frac{\partial N_2}{\partial \eta} & \dots & \frac{\partial N_{\text{NGP}}}{\partial \eta} \\ \frac{\partial N_1}{\partial \zeta} & \frac{\partial N_2}{\partial \zeta} & \dots & \frac{\partial N_{\text{NGP}}}{\partial \zeta} \end{bmatrix} \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ \vdots & \vdots & \vdots \\ x_{\text{NGP}} & y_{\text{NGP}} & z_{\text{NGP}} \end{bmatrix} \quad (1)$$

5. Strain Displacement Relations

The transformations from displacements to strain for the i^{th} grid point are:

$$[C_i] = \begin{bmatrix} C_{1i} & 0 & 0 \\ 0 & C_{2i} & 0 \\ 0 & 0 & C_{3i} \\ C_{2i} & C_{1i} & 0 \\ 0 & C_{3i} & C_{2i} \\ C_{3i} & 0 & C_{1i} \end{bmatrix} \quad (2)$$

STRUCTURAL ELEMENT DESCRIPTIONS

where

$$\begin{Bmatrix} c_{1i} \\ c_{2i} \\ c_{3i} \end{Bmatrix} = [J]^{-1} \begin{Bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \\ \frac{\partial N_i}{\partial z} \end{Bmatrix} \quad (3)$$

8.21.3 Stiffness and Mass Matrix Calculation for IHEXi Elements

The equation used in the stiffness matrix generation in global coordinates is:

$$[K_{ij}] = [T_i]^T \left\{ \sum_{\ell=1}^{NIP} \sum_{m=1}^{NIP} \sum_{n=1}^{NIP} H_{\ell} H_m H_n |J| [C_i]^T [T^M]^T [G_m] [T^M] [C_j] \right\} [T_j] \quad (4)$$

The equation used in the coupled mass matrix generation in global coordinates is:

$$[M_{ij}] = [T_i]^T \left\{ \sum_{\ell=1}^{NIP} \sum_{m=1}^{NIP} \sum_{n=1}^{NIP} \rho H_{\ell} H_m H_n |J| \begin{bmatrix} N_i N_j & 0 & 0 \\ 0 & N_i N_j & 0 \\ 0 & 0 & N_i N_j \end{bmatrix} \right\} [T_j] \quad (5)$$

$[K_{ij}]$ and $[M_{ij}]$ are three by three partitions of the element matrices coupling the i^{th} and j^{th} element grid points. No provision is made for the computation of a lumped mass matrix for the isoparametric hexahedron elements. Such a computation would necessitate an arbitrary distribution of mass reducing the accuracy advantages of these type elements.

8.21.4 Element Load Calculations for IHEXi Elements

1. Thermal Force Vector (Subroutine IHEX of Module SSG1)

The thermal force vector is generated with the following equation:

$$\{P_i\} = [T_i]^T \left\{ \sum_{\ell=1}^{NIP} H_{\ell} \sum_{m=1}^{NIP} H_m \sum_{n=1}^{NIP} H_n |J| [C_i]^T [T^M]^T [G_e] \{\alpha_e\} \left(\sum_{j=1}^{NGP} N_j t_j \right) - T_0 \right\} \quad (6)$$

where $\{P_i\}$ is the load vector of length three for the i^{th} element grid point and t_j is the temperature at the j^{th} element grid point.

2. Pressure Force Vector (Subroutine PLØAD3 of Module SSG1)

The generation of the pressure force vector is described in Section 4.41.8.9.

3. Heat Generation Vector for Heat Transfer Problems (Subroutine QIHEX of Module SSG1)

The terms of the heat generation vector are given by:

$$P_i = -Q \sum_{\ell=1}^2 \sum_{m=1}^2 \sum_{n=1}^2 |J| N_i \quad (7)$$

where Q is the external heat per unit volume.

8.21.5 Element Stress Calculations for IHEXi Elements

1. Calculations performed in SIHEX1 (Phase 1 calculations)

The phase 1 calculations include the computation of stress matrices at each point within the element at which stresses are to be evaluated. The transformation matrix from displacements at the i^{th} grid point to stresses at the j^{th} stress point is:

$$[S_{ji}] = [T^M]^T [G_m] [T^M] [C_i] [T_i] \quad (8)$$

where $[T_i]$ is the global to basic coordinate transformation matrix. All matrices on the right-hand side of this equation are computed at stress point j .

The temperature to stress relation at the j^{th} stress point is:

$$\{S_{tj}\} = -[T^M]^T [G_m] \{\alpha_m\} \quad (9)$$

2. Calculations performed in SIHEX2 (Phase 2 calculations)

The phase 2 calculations include the computation of stresses at each stress point within the element.

The equation for stress at the j^{th} stress point is:

$$\begin{bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \sigma_{xy} \\ \sigma_{yz} \\ \sigma_{zx} \end{bmatrix}_j = \sum_{i=1}^{\text{NGP}} [S_{ji}] \{U_{gi}\} + \{S_{tj}\} \left[\left(\sum_{k=1}^{\text{NGP}} N_k t_k \right) - T_0 \right] \quad (10)$$

where $\{U_{gi}\}$ is the three by one global displacement vector at the i^{th} grid point and t_k is the loading temperature at the k^{th} grid point. The three principal stresses are the roots of the following cubic equation in S :

$$\begin{aligned} S^3 - (\sigma_x + \sigma_y + \sigma_z) S^2 + (\sigma_x \sigma_y + \sigma_y \sigma_z + \sigma_x \sigma_z - \sigma_{xy}^2 - \sigma_{yz}^2 - \sigma_{zx}^2) S - \\ (\sigma_x \sigma_y \sigma_z + 2\sigma_{xy} \sigma_{yz} \sigma_{zx} - \sigma_x \sigma_{yz}^2 - \sigma_y \sigma_{zx}^2 - \sigma_z \sigma_{xy}^2) = 0 \end{aligned} \quad (11)$$

The direction cosines of the normals to the principal plane on which each of the principal stresses is acting are found by solving:

$$\begin{bmatrix} (S - \sigma_x) & -\sigma_{xy} & -\sigma_{zx} \\ -\sigma_{xy} & (S - \sigma_y) & -\sigma_{yz} \\ -\sigma_{zx} & -\sigma_{yz} & (S - \sigma_z) \end{bmatrix} \begin{Bmatrix} l \\ m \\ n \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix} \quad (12)$$

using the constraint equation:

$$l^2 + m^2 + n^2 = 1 \quad (13)$$

THE IHEX1, IHEX2, AND IHEX3 ELEMENTS

The mean stress or hydrostatic pressure is:

$$\sigma_n = -\frac{1}{3}(\sigma_x + \sigma_y + \sigma_z) \quad (14)$$

Note: The minus sign is used to be consistent with the hydrostatic pressure computations for elements CTETRA, CWEDGE, CHEXA1, and CHEXA2.

The octahedral shear stress is:

$$\sigma_o = \left\{ \frac{1}{3} [(S_x + \sigma_n)^2 + (S_y + \sigma_n)^2 + (S_z + \sigma_n)^2] \right\}^{1/2} \quad (15)$$

8.21.6 Differential Stiffness Calculations for IHEXi Elements

The differential stiffness matrix in generalized coordinates is (the stresses are in the basic system):

$$[K_w^d] = \begin{bmatrix} \sigma_y + \sigma_z & -\sigma_{xy} & -\sigma_{xz} \\ -\sigma_{xy} & \sigma_x + \sigma_z & -\sigma_{yz} \\ -\sigma_{xz} & -\sigma_{yz} & \sigma_x + \sigma_y \end{bmatrix} \quad (16)$$

The transformation from displacements at the nodal points to rigid body rotation are:

$$[C_i^d] = \frac{1}{2} \begin{bmatrix} 0 & -C_{3i} & C_{2i} \\ C_{3i} & 0 & -C_{1i} \\ -C_{2i} & C_{1i} & 0 \end{bmatrix} \quad (17)$$

where C_{1i} , C_{2i} , and C_{3i} are given in Section 4.87. .2.

The three by three partitions of the global element differential stiffness matrix coupling the i^{th} and j^{th} element grid points are given by:

$$[K_{ij}^d] = [T_i]^T \left\{ \sum_{\ell=1}^2 \sum_{m=1}^2 \sum_{n=1}^2 |J| [C_i^d]^T [K_w^d] [C_j^d] \right\} [T_j] \quad (18)$$

8.21.7 Heat Transfer Calculations for IHEXi Elements

The terms of the heat conductance matrix are given by:

$$K_{ij} = \sum_{\ell=1}^{NIP} \sum_{m=1}^{NIP} \sum_{n=1}^{NIP} H_{\ell} H_m H_n |J| \{C_i\}^T [T^M]^T [k] [T^M] \{C_j\} \quad (19)$$

where

$$\{C_i\} = \begin{Bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \\ \frac{\partial N_i}{\partial z} \end{Bmatrix} \quad (20)$$

and $[k]$ is the three by three matrix of thermal conductivity coefficients in global coordinates.

The terms of the heat capacitance matrix are given by:

$$B_{ij} = \sum_{\ell=1}^{NIP} \sum_{m=1}^{NIP} \sum_{n=1}^{NIP} H_{\ell} H_m H_n |J| N_i c_p N_j \quad (21)$$

and c_p is the thermal capacity per unit volume.

THE IHEX1, IHEX2, AND IHEX3 ELEMENTS

Table 1. Gaussian Quadrature Formula

$\int_{-1}^1 \int_{-1}^1 \int_{-1}^1 f(x,y,z) dx dy dz = \sum_{i=1}^n \sum_{k=1}^n \sum_{j=1}^n H_j f(s_j, s_k, s_i)$		
n	Abcissa (s)	Weight Coefficient (H)
2	± 0.57735026919	1.0
3	± 0.77459666924 0.0	0.55555555555 0.88888888888
4	± 0.86113631159 ± 0.33998104358	0.34785484514 0.65214515486

STRUCTURAL ELEMENT DESCRIPTIONS

Table 2. Isoparametric Shape Functions and Their Derivatives
for the IHEX1 Element - 8 Grid Points

Corner Grid Points	
$\xi_i = \pm 1, \eta_i = \pm 1, \zeta_i = \pm 1$	
$N_i = \frac{1}{8} (1 + \xi_0) (1 + \eta_0) (1 + \zeta_0)$	
$\frac{\partial N_i}{\partial \xi} = \frac{1}{8} \xi_i (1 + \eta_0) (1 + \zeta_0)$	
$\frac{\partial N_i}{\partial \eta} = \frac{1}{8} \eta_i (1 + \xi_0) (1 + \zeta_0)$	
$\frac{\partial N_i}{\partial \zeta} = \frac{1}{8} \zeta_i (1 + \xi_0) (1 + \eta_0)$	

where $\xi_0 = \xi \xi_i, \eta_0 = \eta \eta_i, \zeta_0 = \zeta \zeta_i$

THE IHEX1, IHEX2, AND IHEX3 ELEMENTS

Table 3. Isoparametric Shape Functions and Their Derivatives
for the IHEX2 Element - 20 Grid Points

Corner Grid Points

$$\xi_i = \pm 1, \quad \eta_i = \pm 1, \quad \zeta_i = \pm 1$$

$$N_i = \frac{1}{8} (1 + \xi_0)(1 + \eta_0)(1 + \zeta_0)(\xi_0 + \eta_0 + \zeta_0 - 2)$$

$$\frac{\partial N_i}{\partial \xi} = \frac{1}{8} \xi_i (1 + \eta_0)(1 + \zeta_0)(2\xi_0 + \eta_0 + \zeta_0 - 1)$$

$$\frac{\partial N_i}{\partial \eta} = \frac{1}{8} \eta_i (1 + \xi_0)(1 + \zeta_0)(2\eta_0 + \xi_0 + \zeta_0 - 1)$$

$$\frac{\partial N_i}{\partial \zeta} = \frac{1}{8} \zeta_i (1 + \xi_0)(1 + \eta_0)(2\zeta_0 + \xi_0 + \eta_0 - 1)$$

Mid-Side Grid Points

$$\xi_i = 0, \quad \eta_i = \pm 1, \quad \zeta_i = \pm 1$$

$$N_i = \frac{1}{4} (1 - \xi^2)(1 + \eta_0)(1 + \zeta_0)$$

$$\frac{\partial N_i}{\partial \xi} = -\frac{1}{2} \xi (1 + \eta_0)(1 + \zeta_0)$$

$$\frac{\partial N_i}{\partial \eta} = \frac{1}{4} (1 - \xi^2)(1 + \zeta_0) \eta_i$$

$$\frac{\partial N_i}{\partial \zeta} = \frac{1}{4} (1 - \xi^2)(1 + \eta_0) \zeta_i$$

Mid-Side Grid Points

$$\xi_i = \pm 1, \quad \eta_i = 0, \quad \zeta_i = \pm 1$$

$$N_i = \frac{1}{4} (1 - \eta^2)(1 + \xi_0)(1 + \zeta_0)$$

$$\frac{\partial N_i}{\partial \xi} = \frac{1}{4} (1 - \eta^2)(1 + \zeta_0) \xi_i$$

$$\frac{\partial N_i}{\partial \eta} = -\frac{1}{2} \eta (1 + \xi_0)(1 + \zeta_0)$$

$$\frac{\partial N_i}{\partial \zeta} = \frac{1}{4} (1 - \eta^2)(1 + \xi_0) \zeta_i$$

Mid-Side Grid Points

$$\xi_i = \pm 1, \quad \eta_i = \pm 1, \quad \zeta_i = 0$$

$$N_i = \frac{1}{4} (1 - \zeta^2)(1 + \xi_0)(1 + \eta_0)$$

$$\frac{\partial N_i}{\partial \xi} = \frac{1}{4} (1 - \zeta^2)(1 + \eta_0) \xi_i$$

$$\frac{\partial N_i}{\partial \eta} = \frac{1}{4} (1 - \zeta^2)(1 + \xi_0) \eta_i$$

$$\frac{\partial N_i}{\partial \zeta} = -\frac{1}{2} \zeta (1 + \xi_0)(1 + \eta_0)$$

where $\xi_0 = \xi \xi_i$, $\eta_0 = \eta \eta_i$, $\zeta_0 = \zeta \zeta_i$

Table 4. Isoparametric Shape Functions and Their Derivatives
for the IHEX3 Element - 32 Grid Points

Corner Grid Points

$$\xi_1 = \pm 1, \quad \eta_1 = \pm 1, \quad \zeta_1 = \pm 1$$

$$N_1 = \frac{9}{64} (1 + \xi_0)(1 + \eta_0)(1 + \zeta_0)(\xi^2 + \eta^2 + \zeta^2 - \frac{19}{9})$$

$$\frac{\partial N_1}{\partial \xi} = \frac{9}{64} (1 + \eta_0)(1 + \zeta_0) [\xi_1 (3\xi^2 + \eta^2 + \zeta^2 - \frac{19}{9}) + 2\xi]$$

$$\frac{\partial N_1}{\partial \eta} = \frac{9}{64} (1 + \xi_0)(1 + \zeta_0) [\eta_1 (3\eta^2 + \xi^2 + \zeta^2 - \frac{19}{9}) + 2\eta]$$

$$\frac{\partial N_1}{\partial \zeta} = \frac{9}{64} (1 + \xi_0)(1 + \eta_0) [\zeta_1 (3\zeta^2 + \xi^2 + \eta^2 - \frac{19}{9}) + 2\zeta]$$

Mid-Side Grid Points

$$\xi_1 = \pm \frac{1}{3}, \quad \eta_1 = \pm 1, \quad \zeta_1 = \pm 1$$

$$N_1 = \frac{9}{64} (1 - \xi^2)(1 + 9\xi_0)(1 + \eta_0)(1 + \zeta_0)$$

$$\frac{\partial N_1}{\partial \xi} = \frac{9}{64} (1 + \eta_0)(1 + \zeta_0) (-2\xi + 9\xi_1 - 27\xi\xi_0)$$

$$\frac{\partial N_1}{\partial \eta} = \frac{9}{64} (1 - \xi^2)(1 + 9\xi_0)(1 + \zeta_0) \eta_1$$

$$\frac{\partial N_1}{\partial \zeta} = \frac{9}{64} (1 - \xi^2)(1 + 9\xi_0)(1 + \eta_0) \zeta_1$$

Mid-Side Grid Points

$$\xi_1 = \pm 1, \quad \eta_1 = \pm \frac{1}{3}, \quad \zeta_1 = \pm 1$$

$$N_1 = \frac{9}{64} (1 - \eta^2)(1 + 9\eta_0)(1 + \xi_0)(1 + \zeta_0)$$

$$\frac{\partial N_1}{\partial \xi} = \frac{9}{64} (1 - \eta^2)(1 + 9\eta_0) \xi_1 (1 + \zeta_0)$$

$$\frac{\partial N_1}{\partial \eta} = \frac{9}{64} (1 + \xi_0)(1 + \zeta_0) (-2\eta + 9\eta_1 - 27\eta\eta_0)$$

$$\frac{\partial N_1}{\partial \zeta} = \frac{9}{64} (1 - \eta^2)(1 + 9\eta_0) \zeta_1 (1 + \xi_0)$$

Mid-Side Grid Points

$$\xi_1 = \pm 1, \quad \eta_1 = \pm 1, \quad \zeta_1 = \pm \frac{1}{3}$$

$$N_1 = \frac{9}{64} (1 - \zeta^2)(1 + 9\zeta_0)(1 + \xi_0)(1 + \eta_0)$$

$$\frac{\partial N_1}{\partial \xi} = \frac{9}{64} (1 - \zeta^2)(1 + 9\zeta_0) \xi_1 (1 + \eta_0)$$

$$\frac{\partial N_1}{\partial \eta} = \frac{9}{64} (1 - \zeta^2)(1 + 9\zeta_0) \eta_1 (1 + \xi_0)$$

$$\frac{\partial N_1}{\partial \zeta} = \frac{9}{64} (1 + \xi_0)(1 + \eta_0) (-2\zeta + 9\zeta_1 - 27\zeta\zeta_0)$$

where $\xi_0 = \xi\xi_1$, $\eta_0 = \eta\eta_1$, $\zeta_0 = \zeta\zeta_1$

THE TRAPAX ELEMENT

8.22 THE TRAPAX ELEMENT

8.22.1 Input Data for TRAPAX Element

1. The ECPT/EST entries for the assym-trapezoidal ring (TRAPAX) element are

<u>Symbol</u>	<u>Description</u>
EID	Element ID*1000 + harmonic number + 1
SIL ₁ ,SIL ₂ ,SIL ₃ ,SIL ₄	Scalar index numbers for the four grid points
Y	Material property orientation angle (degrees)
MAT I.D.	Material property identification number
Θ _i , i= 14	Angles defining points around element
N ₁ ,R ₁ ,Z ₁ ,0.0 N ₂ ,R ₂ ,Z ₂ ,0.0 N ₃ ,R ₃ ,Z ₃ ,0.0 N ₄ ,R ₄ ,Z ₄ ,0.0	Local coordinates systems number and location in basic coordinate of the four grid points
T _μ	Element temperature for material properties
N	Harmonic index

For this element N₁ must equal zero for i = 1, 2, 3 and 4, and we define:

$$\{R_s\} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \quad (1)$$

$$Z_s = \begin{pmatrix} z_{s1} \\ z_{s2} \\ z_{s3} \\ z_{s4} \end{pmatrix} = \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{pmatrix} \quad (2)$$

STRUCTURAL ELEMENT DESCRIPTIONS

2. Material Property Input

The material property input for the TRAPAX element is the same as that for the TRIAAX element (see Section 4.87.17.1).

8.22.2 General Calculations

1. Local coordinate calculations are:

$$Z_{\min} = \text{minimum of } (Z_{s1}, Z_{s2}, Z_{s3}, Z_{s4}) \quad , \quad (3)$$

$$\{R_L\} = \{R_S\} \quad , \quad (4)$$

$$\{Z_L\} = \{Z_S\} - \begin{pmatrix} Z_{\min} \\ Z_{\min} \\ Z_{\min} \\ Z_{\min} \end{pmatrix} \quad . \quad (5)$$

2. Test for a rectangle. Let R_{Li} be the i^{th} component of $\{R_L\}$, and define:

$$R_{M14} = (R_{L1} + R_{L4})/2 \quad , \quad (6)$$

$$R_{M23} = (R_{L2} + R_{L3})/2 \quad . \quad (7)$$

$$\text{If } \left| \frac{R_{L1} - R_{L4}}{R_{M14}} \right| < 0.005 \text{ then } R_{L1} = R_{L4} = R_{M14} .$$

$$\text{If } \left| \frac{R_{L2} - R_{L3}}{R_{M23}} \right| < 0.005 \text{ then } R_{L2} = R_{L3} = R_{M23} .$$

If $R_{L1} = R_{L4}$ and $R_{L2} = R_{L3}$, then the element is rectangular.

(NOTE: .005 is geometry dependent and may need modification)

THE TRAPAX ELEMENT

3. Generate the transformation matrix from field coordinates to grid point degrees of freedom:

$$[\bar{H}] = \begin{bmatrix} 1 & R_{L1} & Z_{L1} & R_{L1}Z_{L1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & R_{L1} & Z_{L1} & R_{L1}Z_{L1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & R_{L1} & Z_{L1} & R_{L1}Z_{L1} \\ 1 & R_{L2} & Z_{L2} & R_{L2}Z_{L2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & R_{L2} & Z_{L2} & R_{L2}Z_{L2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & R_{L2} & Z_{L2} & R_{L2}Z_{L2} \\ 1 & R_{L3} & Z_{L3} & R_{L3}Z_{L3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & R_{L3} & Z_{L3} & R_{L3}Z_{L3} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & R_{L3} & Z_{L3} & R_{L3}Z_{L3} \\ 1 & R_{L4} & Z_{L4} & R_{L4}Z_{L4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & R_{L4} & Z_{L4} & R_{L4}Z_{L4} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & R_{L4} & Z_{L4} & R_{L4}Z_{L4} \end{bmatrix} \quad (8)$$

$$[H] = [\bar{H}]^{-1}$$

8.22.3 Integral Calculations

a. Compute the integrals over the cross-section of the trapezoid that are of the form:

$$\delta_{pq} = \iint_{r,z} r^p z^q dr dz \quad (9)$$

Subroutine DKL calculates the values:

$$\delta_{-1,0}, \delta_{-1,1}, \delta_{-1,2}, \delta_{0,0}, \delta_{0,1}, \delta_{0,2}, \delta_{1,0}, \delta_{1,1}, \delta_{1,2}, \delta_{2,0}, \delta_{2,1}, \delta_{2,2}, \delta_{3,0}, \delta_{3,1}, \delta_{3,2} \quad .$$

See section 16.4-2 of the Theoretical Manual for equations on calculating the integrals.

8.22.4 Elastic Constants Matrix Calculation

The elastic constants matrix in element coordinates, $[Eg]$, for the TRAPAX element is calculated identically to the elastic constant matrix for the TRIAAX element (see Section 8.17.4).

8.22.5 Stiffness Matrix Generation (Subroutine KTPZ of Module EMG)

1. Generate the terms of the symmetric element stiffness matrix in field coordinates as shown below. Each term must be multiplied by $\pi(2\pi \text{ if } n=0)$ to form $[\tilde{k}]$.

$$[\tilde{k}_n] = \begin{bmatrix} \tilde{k}_{11} & \tilde{k}_{12} & \tilde{k}_{13} & \tilde{k}_{14} & \tilde{k}_{15} & \tilde{k}_{16} & \tilde{k}_{17} & \tilde{k}_{18} & \tilde{k}_{19} & \tilde{k}_{1,10} & \tilde{k}_{1,11} & \tilde{k}_{1,12} \\ & \tilde{k}_{22} & \tilde{k}_{23} & \tilde{k}_{24} & \tilde{k}_{25} & \tilde{k}_{26} & \tilde{k}_{27} & \tilde{k}_{28} & \tilde{k}_{29} & \tilde{k}_{2,10} & \tilde{k}_{2,11} & \tilde{k}_{2,12} \\ & & \tilde{k}_{33} & \tilde{k}_{34} & \tilde{k}_{35} & \tilde{k}_{36} & \tilde{k}_{37} & \tilde{k}_{38} & \tilde{k}_{39} & \tilde{k}_{3,10} & \tilde{k}_{3,11} & \tilde{k}_{3,12} \\ & & & \tilde{k}_{44} & \tilde{k}_{45} & \tilde{k}_{46} & \tilde{k}_{47} & \tilde{k}_{48} & \tilde{k}_{49} & \tilde{k}_{4,10} & \tilde{k}_{4,11} & \tilde{k}_{4,12} \\ & & & & \tilde{k}_{55} & \tilde{k}_{56} & \tilde{k}_{57} & \tilde{k}_{58} & \tilde{k}_{59} & \tilde{k}_{5,10} & \tilde{k}_{5,11} & \tilde{k}_{5,12} \\ & & & & & \tilde{k}_{66} & \tilde{k}_{67} & \tilde{k}_{68} & 0 & \tilde{k}_{6,10} & \tilde{k}_{6,11} & \tilde{k}_{6,12} \\ & & & & & & \tilde{k}_{77} & \tilde{k}_{78} & \tilde{k}_{79} & \tilde{k}_{7,10} & \tilde{k}_{7,11} & \tilde{k}_{7,12} \\ & & & & & & & \tilde{k}_{88} & \tilde{k}_{89} & \tilde{k}_{8,10} & \tilde{k}_{8,11} & \tilde{k}_{8,12} \\ & & & & & & & & \tilde{k}_{99} & \tilde{k}_{9,10} & \tilde{k}_{9,11} & \tilde{k}_{9,12} \\ & & & & & & & & & \tilde{k}_{10,10} & \tilde{k}_{10,11} & \tilde{k}_{10,12} \\ & & & & & & & & & & \tilde{k}_{11,11} & \tilde{k}_{11,12} \\ & & & & & & & & & & & \tilde{k}_{12,12} \end{bmatrix} \quad (10)$$

Symmetric

THE TRAPAX ELEMENT

where for the n^{th} harmonic

$$\begin{aligned}
 \tilde{K}_{11} &= (E_{33} + n^2 E_{55}) \delta_{-1,0} \\
 \tilde{K}_{12} &= (E_{13} + E_{33} + n^2 E_{55}) \delta_{0,0} \\
 \tilde{K}_{13} &= (E_{33} + n^2 E_{55}) \delta_{-1,1} + E_{34} \delta_{0,0} \\
 \tilde{K}_{14} &= (E_{13} + E_{33} + n^2 E_{55}) \delta_{0,1} + E_{34} \delta_{1,0} \\
 \tilde{K}_{15} &= n(E_{33} + E_{55}) \delta_{-1,0} \\
 \tilde{K}_{16} &= nE_{33} \delta_{0,0} \\
 \tilde{K}_{17} &= n(E_{33} + E_{55}) \delta_{-1,1} - nE_{56} \delta_{0,0} \\
 \tilde{K}_{18} &= nE_{33} \delta_{0,1} - nE_{56} \delta_{1,0} \\
 \tilde{K}_{19} &= n^2 E_{56} \delta_{-1,0} \\
 \tilde{K}_{1,10} &= \delta_{0,0} (E_{34} + n^2 E_{56}) \\
 \tilde{K}_{1,11} &= \delta_{0,0} (E_{23} + n^2 \delta_{-1,1} E_{56}) \\
 \tilde{K}_{1,12} &= E_{23} \delta_{1,0} + \delta_{0,1} (E_{34} + n^2 E_{56}) \\
 \tilde{K}_{2,2} &= E_{23} \delta_{1,0} + \delta_{0,1} (E_{34} + n^2 E_{56}) \\
 \tilde{K}_{2,3} &= (E_{11} + 2E_{13} + E_{33} + n^2 E_{55}) \delta_{1,0} \\
 \tilde{K}_{2,4} &= (E_{11} + 2E_{13} + E_{33} + n^2 E_{55}) \delta_{1,1} + (E_{14} + E_{34}) \delta_{2,0} \\
 \tilde{K}_{2,5} &= n(E_{13} + E_{33} + E_{55}) \delta_{0,0} \\
 \tilde{K}_{2,6} &= n(E_{13} + E_{33}) \delta_{1,0} \\
 \tilde{K}_{2,7} &= n(E_{13} + E_{33} + E_{55}) \delta_{0,1} - nE_{56} \delta_{1,0} \\
 \tilde{K}_{2,8} &= n(E_{13} + E_{33}) \delta_{1,1} - nE_{56} \delta_{2,0} \\
 \tilde{K}_{2,9} &= n^2 E_{56} \delta_{0,0} \\
 \tilde{K}_{2,10} &= \delta_{1,0} (E_{14} + n^2 E_{56}) \\
 \tilde{K}_{2,11} &= \delta_{1,0} (E_{12} + E_{23}) + n^2 \delta_{0,1} E_{56} \\
 \tilde{K}_{2,12} &= \delta_{2,0} (E_{12} + E_{23}) + \delta_{1,1} (E_{14} + E_{34}) + n^2 \delta_{1,1} E_{56} \\
 \tilde{K}_{3,3} &= (E_{33} + n^2 E_{55}) \delta_{-1,2} + 2E_{34} \delta_{0,1} + E_{44} \delta_{1,0} \\
 \tilde{K}_{3,4} &= (E_{13} + E_{33} + n^2 E_{55}) \delta_{0,2} + E_{44} \delta_{2,0} + (E_{14} + 2E_{34}) \delta_{1,1} \\
 \tilde{K}_{3,5} &= n(E_{33} + E_{55}) \delta_{-1,1} + nE_{34} \delta_{0,0} \\
 \tilde{K}_{3,6} &= nE_{33} \delta_{0,1} + nE_{34} \delta_{1,0} \\
 \tilde{K}_{3,7} &= n(E_{33} + E_{55}) \delta_{-1,2} + n(E_{34} - E_{56}) \delta_{0,1} \\
 \tilde{K}_{3,8} &= nE_{33} \delta_{0,2} + n(E_{34} - E_{56}) \delta_{1,1} \\
 \tilde{K}_{3,9} &= n^2 E_{56} \delta_{-1,1}
 \end{aligned}$$

STRUCTURAL ELEMENT DESCRIPTIONS

$$\begin{aligned}
 \tilde{K}_{3,10} &= E_{44}\delta_{1,0} + \delta_{0,1}(E_{34} + n^2E_{56}) \\
 \tilde{K}_{3,11} &= E_{24}\delta_{1,0} + \delta_{0,1}E_{23} + n^2\delta_{-1,2}E_{56} \\
 \tilde{K}_{3,12} &= E_{24}\delta_{2,0} + (E_{23} + E_{24})\delta_{1,1} + (E_{34} + n^2E_{56})\delta_{0,2} \\
 \tilde{K}_{4,4} &= (E_{11} + 2E_{13} + E_{33} + n^2E_{55})\delta_{1,2} + (2E_{14} + 2E_{34})\delta_{2,1} + E_{44}\delta_{3,0} \\
 \tilde{K}_{4,5} &= n(E_{13} + E_{33} + E_{55})\delta_{0,1} + nE_{34}\delta_{1,0} \\
 \tilde{K}_{4,6} &= n(E_{13} + E_{33})\delta_{1,1} + nE_{34}\delta_{2,0} \\
 \tilde{K}_{4,7} &= n(E_{13} + E_{23} + E_{55})\delta_{0,2} + n(E_{34} - E_{56})\delta_{1,1} \\
 \tilde{K}_{4,8} &= n(E_{13} + E_{33})\delta_{1,2} + n(E_{34} - E_{56})\delta_{2,1} \\
 \tilde{K}_{4,9} &= n^2E_{56}\delta_{0,1} \\
 \tilde{K}_{4,10} &= \delta_{1,1}(E_{12} + E_{34} + n^2E_{56}) + \delta_{2,0}E_{44} \\
 \tilde{K}_{4,11} &= \delta_{1,1}(E_{12} + E_{23}) + \delta_{2,0}E_{24} + n^2\delta_{0,2}E_{56} \\
 \tilde{K}_{4,12} &= \delta_{2,1}(E_{12} + E_{23} + E_{44}) + \delta_{1,2}(E_{14} + E_{34} + n^2E_{56}) + \delta_{3,0}E_{24} \\
 \tilde{K}_{5,5} &= (E_{55} + n^2E_{33})\delta_{-1,0} \\
 \tilde{K}_{5,6} &= n^2E_{33}\delta_{0,0} \\
 \tilde{K}_{5,7} &= (E_{55} + n^2E_{33})\delta_{-1,1} - E_{56}\delta_{0,0} \\
 \tilde{K}_{5,8} &= n^2E_{33}\delta_{0,1} - E_{56}\delta_{1,0} \\
 \tilde{K}_{5,9} &= nE_{56}\delta_{-1,0} \\
 \tilde{K}_{5,10} &= n\delta_{0,0}(E_{34} + E_{56}) \\
 \tilde{K}_{5,11} &= n\delta_{0,0}E_{23} + \delta_{-1,1}E_{56} \\
 \tilde{K}_{5,12} &= n\delta_{1,0}E_{23} + n\delta_{0,1}(E_{34} + E_{56}) \\
 \tilde{K}_{6,6} &= n^2E_{33}\delta_{1,0} \\
 \tilde{K}_{6,7} &= n^2E_{33}\delta_{0,1} \\
 \tilde{K}_{6,8} &= n^2E_{33}\delta_{1,1} \\
 \tilde{K}_{6,10} &= n\delta_{1,0}E_{34} \\
 \tilde{K}_{6,11} &= n\delta_{1,0}E_{23} \\
 \tilde{K}_{6,12} &= n\delta_{2,0}E_{23} + n\delta_{1,1}E_{34} \\
 \tilde{K}_{7,7} &= (E_{55} + n^2E_{33})\delta_{-1,2} - 2E_{56}\delta_{0,1} + E_{66}\delta_{1,0} \\
 \tilde{K}_{7,8} &= n^2E_{33}\delta_{0,2} - E_{56}\delta_{1,1} + E_{66}\delta_{2,0} \\
 \tilde{K}_{7,9} &= nE_{56}\delta_{-1,1} - nE_{66}\delta_{0,0} \\
 \tilde{K}_{7,10} &= n\delta_{0,1}(E_{34} + E_{56}) - n\delta_{1,0}E_{66} \\
 \tilde{K}_{7,11} &= n\delta_{0,1}(E_{23} - E_{66}) + n\delta_{-1,2}E_{56} \\
 \tilde{K}_{7,12} &= n\delta_{1,1}(E_{23} - E_{66}) + n\delta_{0,2}(E_{34} + E_{56})
 \end{aligned}$$

THE TRAPAX ELEMENT

$$\begin{aligned}
 \tilde{K}_{8,8} &= E_{66}\delta_{3,0} + n^2 E_{33}\delta_{1,2} \\
 \tilde{K}_{8,9} &= n E_{66}\delta_{1,0} \\
 \tilde{K}_{8,10} &= n\delta_{1,1}E_{34} = n\delta_{2,0}E_{66} \\
 \tilde{K}_{8,11} &= n\delta_{1,1}(E_{23} - E_{66}) \\
 \tilde{K}_{8,12} &= n\delta_{2,1}(E_{23} + E_{66}) + n\delta_{1,2}E_{34} \\
 \tilde{K}_{9,9} &= n^2 E_{66}\delta_{-1,0} \\
 \tilde{K}_{9,10} &= n^2 \delta_{0,0}E_{66} \\
 \tilde{K}_{9,11} &= n^2 \delta_{-1,1}E_{66} \\
 \tilde{K}_{9,12} &= n^2 \delta_{0,1}E_{66} \\
 \tilde{K}_{10,10} &= \delta_{1,0}(E_{44} + n^2 E_{66}) \\
 \tilde{K}_{10,11} &= \delta_{1,0}E_{24} + n^2 \delta_{0,1}E_{66} \\
 \tilde{K}_{10,12} &= \delta_{2,0}E_{24} + \delta_{1,1}(E_{44} + n^2 E_{66}) \\
 \tilde{K}_{11,11} &= \delta_{1,0}E_{22} + n^2 \delta_{-1,2}E_{66} \\
 \tilde{K}_{11,12} &= \delta_{2,0}E_{22} + \delta_{1,1}E_{24} + n^2 \delta_{0,2}E_{66} \\
 \tilde{K}_{12,12} &= E_{22}\delta_{3,0} + 2E_{24}\delta_{2,1} + \delta_{1,2}(E_{44} + n^2 E_{66})
 \end{aligned}$$

2. Transform the element stiffness matrix from field coordinates to grid point degrees of freedom:

$$[\tilde{K}_{pj}] = [H]^T [\tilde{K}] [H] \quad (11)$$

3. The 3 by 3 partitions $[K_{pj}]$ of $[K]$, corresponding to the pivot point p , are extracted from $[\tilde{K}_{pj}]$, then for insertion, these 3 by 3 partitions are expanded to 6 by 6 partitions:

$$[K_{pj}] = \begin{bmatrix} \tilde{K}_{pj} & \vdots & 0 \\ \vdots & \vdots & \vdots \\ 0 & \vdots & 0 \end{bmatrix} \quad (12)$$

8.22.6 Mass Matrix Calculations (Subroutine MPZDA of Module EMG)

1. Generate the consistent mass matrix in field coordinates

STRUCTURAL ELEMENT DESCRIPTIONS

$$[M] = (12 \times 12) \quad \rho \begin{bmatrix} [\Delta] & & \\ & [\Delta] & \\ & & [\Delta] \end{bmatrix} \quad (13)$$

where

$$[\Delta] = \begin{bmatrix} A_{1,0} & A_{2,0} & A_{1,1} & A_{2,1} \\ & A_{3,0} & A_{2,1} & A_{3,1} \\ -SYM & & A_{1,2} & A_{2,2} \\ & & & A_{3,2} \end{bmatrix} \quad (14)$$

and

$$\begin{aligned} \rho &= 2\pi\rho \quad \text{if } n = 0 \\ \rho &= \pi\rho \quad \text{if } n > 0 \end{aligned}$$

The integrals for the trapezoidal ring, $A_{1,0}, A_{2,0}, A_{1,1}, A_{2,1}, A_{3,0}, A_{3,1}, A_{1,2}, A_{2,2}$, and $A_{3,2}$ are computed by subroutine DKL and are equivalent to the integrals I_{ij} .

2. Transform the mass matrix to grid point degrees of freedom:

$$[\bar{M}] = [H]^T [\tilde{M}] [H] \quad (15)$$

3. The 6 by 6 partitions, $[M_{pj}]$, are calculated as in equations 22 and 23.

4. Generate the lumped mass matrix

$$\begin{aligned} \text{For } n = 0, M &= 2\pi\rho R_L A \\ \text{For } n > 0, M &= \pi\rho R_L A \end{aligned} \quad (16)$$

$$\text{where } A = 1/2(R_1(Z_2 - Z_4) + R_2(Z_3 - Z_1) + R_3(Z_4 - Z_2) + R_4(Z_1 - Z_3)) \quad (17)$$

$$\text{and } R_L = 1/4 \sum_{i=1}^4 R_i$$

8.22.7 Thermal Load calculations (Subroutine TPZTEM of Module SSG1)

1. Form the temperature gradient vector:

$$\{\Delta T\} = \{T\} - \begin{Bmatrix} T_0 \\ T_0 \\ T_0 \\ T_0 \end{Bmatrix}, \quad (18)$$

where $\{T\}$ is the vector of loading temperatures at the grid points.

2. Form the thermal expansion coefficient vector:

$$\{\alpha\} = \begin{Bmatrix} \alpha_r \\ \alpha_z \\ \alpha_\theta \\ 0 \\ 0 \\ 0 \end{Bmatrix} \quad (6 \times 1) \quad (19)$$

3. Multiply the elastic constants matrix by the thermal expansion coefficient vector:

$$\{B\} = [E_g]\{\alpha\}$$

4. Form the
- $\{Q\}$
- matrix:

$$[A]_{(12 \times 4)} = \begin{bmatrix} B_3 A_{0,0} & B_3 A_{1,0} & B_3 A_{0,1} & B_3 A_{1,1} \\ (B_1+B_3)A_{1,0} & (B_1+B_3)A_{2,0} & (B_1+B_3)A_{1,1} & (B_1+B_3)A_{2,1} \\ B_3 A_{0,1}+B_4 A_{1,0} & B_3 A_{1,1}+B_4 A_{2,0} & B_3 A_{0,2}+B_4 A_{1,1} & B_3 A_{1,2}+B_4 A_{2,1} \\ (B_1+B_3)A_{1,1}+B_4 A_{2,0} & (B_1+B_3)A_{2,1}+B_4 A_{3,0} & (B_1+B_3)A_{1,2}+B_4 A_{2,1} & (B_1+B_3)A_{2,2}+B_4 A_{3,1} \\ n B_3 A_{0,0} & n B_3 A_{1,0} & n B_3 A_{0,1} & n B_3 A_{1,1} \\ n B_3 A_{1,0} & n B_3 A_{2,0} & n B_3 A_{1,1} & n B_3 A_{2,1} \\ n B_3 A_{0,1} & n B_3 A_{1,1} & n B_3 A_{0,2} & n B_3 A_{1,2} \\ n B_3 A_{1,1} & n B_3 A_{2,1} & n B_3 A_{1,2} & n B_3 A_{2,2} \\ 0 & 0 & 0 & 0 \\ B_4 A_{1,0} & B_4 A_{2,0} & B_4 A_{1,1} & B_4 A_{2,1} \\ B_2 A_{1,0} & B_2 A_{2,0} & B_2 A_{1,1} & B_2 A_{2,1} \\ B_2 A_{2,0}+B_4 A_{1,1} & B_2 A_{3,0}+B_4 A_{2,1} & B_2 A_{2,1}+B_4 A_{1,2} & B_2 A_{3,1}+B_4 A_{2,2} \end{bmatrix} \quad (20)$$

where the integrals for the trapezoidal ring, $A_{0,0}, A_{0,1}$, etc. are computed by subroutine AIS.

STRUCTURAL ELEMENT DESCRIPTIONS

5. Compute the thermal load in field coordinates:

$$\{\tilde{F}_T\} = C [Q][H'] \{\Delta T\} \quad , \quad (21)$$

$$\text{where } AB = (R_2 - R_1) (R_3 - R_4) (Z_4 - Z_1)$$

$$\text{and } C = \pi \text{ if } n \neq 0$$

$$C = 2\pi \text{ if } n = 0$$

6. Transform the thermal load to grid point degrees of freedom:

$$\{\bar{F}_T\} = [H]^T \{\tilde{F}_T\} \quad (22)$$

7. These vectors are added to the overall load vector, $\{P_g\}$.

8.22.8 Element Stress and Force Calculations (Subroutines STPAX1, STPAX2, and STPAX3 of Module SDR2)

Phase I calculations are as follows:

1. Form the element stiffness matrix $[K]$ as in Section
2. Define a fifth "grid point" to be the average of the other four points:

$$R_{L5} = 1/4(R_{L1} + R_{L2} + R_{L3} + R_{L4}) \quad , \quad (23)$$

$$Z_{L5} = 1/4(Z_{L1} + Z_{L2} + Z_{L3} + Z_{L4}) \quad . \quad (24)$$

3. Form the matrices $[D^{(1)}]$, $[D^{(2)}]$, $[D^{(3)}]$, $[D^{(4)}]$, $[D^{(5)}]$

$$[D^{(i)}] = \begin{bmatrix} 0 & 1 & 0 & Z_i & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & R_i \\ 1/R_i & 1 & Z_i/R_i & Z_i & +n/R_i & +n & +nZ_i/R_i & +nZ_i & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & R_i & 0 & 0 & 0 & 0 & 0 & 1 & 0 & Z_i \\ -n/R_i & -n & -nZ_i/R_i & -nZ_i & -1/R_i & 0 & -Z_i/R_i & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & R_i & -n/R_i & -n & -nZ_i/R_i & +nZ_i \end{bmatrix} \quad (25)$$

where $i = 1$ to 5 denotes the five grid points.

THE TRAPAX ELEMENT

4. Compute the stress matrices for each of the five grid points in field coordinates:

$$[\tilde{S}^{(i)}] = [E_g] [D^{(i)}] \quad . \quad (26)$$

5. Transform each stress matrix to grid point degrees of freedom:

$$[\tilde{S}^{(i)}] = [\tilde{S}^{(i)}] [H] \quad . \quad (27)$$

6. Form the master stress matrix:

$$[S] = \begin{bmatrix} S^{(1)} \\ S^{(2)} \\ S^{(3)} \\ S^{(4)} \\ S^{(5)} \end{bmatrix} \quad . \quad 20 \times 12 \quad (28)$$

7. Form the thermal stress vector:

$$P_s = E_{pa} \quad (29)$$

Phase 2 Calculations are as follows:

1. The elements in the EST are ordered by harmonic and TRAPAX I.D. number. All harmonic elements for each TRAPAX are grouped together. When the harmonic, N, of an element is zero, it indicates that it is the first of a group of element. Storage space is allotted for four 35 by 14 vectors, defining the element forces at four points and stresses on its centroid and grid points.

2. Extract the displacement vector $\{\Delta\}$, at the three translational components of the grid points from the global displacement vector.

3. Calculate the element forces:

$$\{P\} = [K] \{\Delta\} \quad . \quad (30)$$

4. Calculate the element stresses:

$$\{\sigma\} = [S]_k \{\Delta\} - T_d \{T_s\} \quad . \quad (31)$$

STRUCTURAL ELEMENT DESCRIPTIONS

where

$$T_d = \begin{cases} T_i - T_0, & \text{if } i \neq 5 \text{ (} T_i \text{ is the temperature at } i^{\text{th}} \text{ point)} \\ T_{\text{avg}} - T_0, & \text{if } i = 5 \text{ (} T_{\text{avg}} \text{ is the average temperature over} \\ & \text{the four grid points)} \\ \text{if } N > 0 \text{ } T_0 = 0 \text{ (} T_0 \text{ is reference temperature)} \end{cases} \quad (32)$$

5. Sum, the harmonic element stresses

$$\sigma_h = \sigma_0 + \sum_{K=2}^{N+1} [C_0^{(sy)}] \{\sigma^{(K)}\} \quad (33)$$

where

For symmetric case $sy=0$

For antisymmetric case $sy=1$

$$[C_0^{(0)}] = \begin{bmatrix} \cos N\theta_i & 0 & 0 & 0 & 0 & 0 \\ 0 & \cos N\theta_i & 0 & 0 & 0 & 0 \\ 0 & 0 & \cos N\theta_i & 0 & 0 & 0 \\ 0 & 0 & 0 & \cos N\theta_i & 0 & 0 \\ 0 & 0 & 0 & 0 & \sin N\theta_i & 0 \\ 0 & 0 & 0 & 0 & 0 & \sin N\theta_i \end{bmatrix} \quad (34)$$

$$[C_0^{(1)}] = \begin{bmatrix} \sin N\theta_i & 0 & 0 & 0 & 0 & 0 \\ 0 & \sin N\theta_i & 0 & 0 & 0 & 0 \\ 0 & 0 & \sin N\theta_i & 0 & 0 & 0 \\ 0 & 0 & 0 & \sin N\theta_i & 0 & 0 \\ 0 & 0 & 0 & 0 & -\cos N\theta_i & 0 \\ 0 & 0 & 0 & 0 & 0 & -\cos N\theta_i \end{bmatrix} \quad (35)$$

THE TRAPAX ELEMENT

6. Sum, the harmonic element forces at each grid point

$$P_i = P_0 + \sum_{K=2}^{N+1} [C_0^{(sy)}] \{P(K)\} \quad (36)$$

where

For symmetric case $sy=0$

For antisymmetric case $sy=1$

$$[C_0^{(0)}] = \begin{bmatrix} \cos N\theta_i & 0 & 0 \\ 0 & \sin N\theta_i & 0 \\ 0 & 0 & \cos N\theta_i \end{bmatrix} \quad (37)$$

$$[C_0^{(1)}] = \begin{bmatrix} \sin N\theta_i & 0 & 0 \\ 0 & \cos N\theta_i & 0 \\ 0 & 0 & \sin N\theta_i \end{bmatrix} \quad (38)$$

THE TRIAAX ELEMENT

8.23 THE TRIAAX ELEMENT

8.23.1 Input Data for TRIAAX Element

1. The ECPT/EST entries for the axisymmetric triangular ring (TRIAAX) element are:

<u>Symbol</u>	<u>Description</u>
EID	Element ID*1000 + harmonic number +1
SIL ₁ ,SIL ₂ ,SIL ₃	Scalar indices of the N th harmonic of the three grid points.
Y	Material property orientation angle (degrees).
MAT I.D.	Material property identification number.
φ _i , i=1, 14	Angles defining points around element.
$\left. \begin{array}{l} N_1, R_1, Z_1, 0.0 \\ N_2, R_2, Z_2, 0.0 \\ N_3, R_3, Z_3, 0.0 \end{array} \right\}$	Local coordinates system number and location in basic coordinate of the three grid points.
t _μ	Element temperature for material properties.
N	Harmonic index.

For this element N_i must equal zero for i = 1, 2 and 3, and we define:

$$\{R_s\} = \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} \quad (1)$$

$$\{Z_s\} = \begin{pmatrix} Z_{s1} \\ Z_{s2} \\ Z_{s3} \end{pmatrix} = \begin{pmatrix} Z_1 \\ Z_2 \\ Z_3 \end{pmatrix} \quad (2)$$

STRUCTURAL ELEMENT DESCRIPTIONS

2. Material data

The material property identification number, Mat I.D., and the element temperature for material properties, t_μ , are used to select the following data items. For this element, material properties may be defined on a MAT1 or MAT3 bulk data card.

E_r, E_θ, E_z	Young's moduli in the radial, tangential and axial directions respectively.
$\nu_{r\theta}, \nu_{\theta z}, \nu_{zr}$	Poisson's ratios in the three directions indicated.
	Mass density.
$G_{r\theta}, G_{\theta z}, G_{rz}$	Shear moduli in the three directions indicated.
$\alpha_r, \alpha_\theta, \alpha_z$	Coefficients of thermal expansion in the three directions indicated.
T_0	Thermal expansion reference temperature.
g_e	Structural element damping coefficient.

8.23.2 General Geometric Calculations

1. Local coordinate calculations are:

$$Z_{\min} = \text{minimum of } (Z_{s1}, Z_{s2}, Z_{s3}) \quad , \quad (3)$$

$$\{R_L\} = \{R_S\} \quad , \quad (4)$$

$$\{Z_L\} = \{Z_S\} - \begin{Bmatrix} Z_{\min} \\ Z_{\min} \\ Z_{\min} \end{Bmatrix} \quad (5)$$

2. The transformation from field coordinates to grid point degrees of freedom is given by the inverse of (R_{Li} and Z_{Li} are the i^{th} components of $\{R_L\}$ and $\{Z_L\}$ respectively):

THE TRIAX ELEMENT

$$[H_{Bq}]^{-1} = \begin{bmatrix} 1 & R_{L1} & Z_{L1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & R_{L1} & Z_{L1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & R_{L1} & Z_{L1} \\ 1 & R_{L2} & Z_{L2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & R_{L2} & Z_{L2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & R_{L2} & Z_{L2} \\ 1 & R_{L3} & Z_{L3} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & R_{L3} & Z_{L3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & R_{L3} & Z_{L3} \end{bmatrix}_{9 \times 9} \quad (6)$$

8.23.3 Integral Calculations

1. The integrals over the area of the cross-section are of the form:

$$\delta_{i,j} = \iint r^i z^j dz dr \quad (7)$$

subroutine DKL calculates the values:

$$\delta_{-1,0}, \delta_{-1,1}, \delta_{-1,2}, \delta_{0,1}, \delta_{1,0}$$

8.23.4 Elastic Constants Matrix Calculations

1. Generate the transformation from material axis to element geometric axis.

$$[T_{eo}] = \begin{bmatrix} \cos^2 & \sin^2 & 0 & -2\sin \cos & 0 & 0 \\ \sin^2 & \cos^2 & 0 & 2\sin \cos & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ \sin \cos & -\sin \cos & 0 & \cos^2 - \sin^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cos & -\sin \\ 0 & 0 & 0 & 0 & \sin & \cos \end{bmatrix} \quad (8)$$

STRUCTURAL ELEMENT DESCRIPTIONS

2. Generate the matrix of elastic constants for an orthotropic body with respect to cylindrical coordinates:

$$[E_m] = \frac{1}{\Delta} \begin{bmatrix} E_r(1-\nu_{\theta z}\nu_{z\theta}) & E_r(\nu_{zr} + \nu_{z\theta}\nu_{\theta r}) & E_r(\nu_{\theta r} + \nu_{\theta z}\nu_{zr}) & 0 & 0 & 0 \\ E_r(\nu_{zr} + \nu_{z\theta}\nu_{\theta r}) & E_z(1-\nu_{\theta z}\nu_{\theta r}) & E_z(\nu_{\theta z} + \nu_{\theta r}\nu_{rz}) & 0 & 0 & 0 \\ E_r(\nu_{\theta r} + \nu_{\theta z}\nu_{zr}) & E_z(\nu_{\theta z} + \nu_{\theta r}\nu_{rz}) & E_\theta(1-\nu_{rz}\nu_{zr}) & 0 & 0 & 0 \\ 0 & 0 & 0 & G_{rz}\Delta & 0 & 0 \\ 0 & 0 & 0 & 0 & G_{r\theta}\Delta & 0 \\ 0 & 0 & 0 & 0 & 0 & G_{z\theta}\Delta \end{bmatrix} \quad \begin{matrix} \text{(Symmetric)} \\ \\ \\ \end{matrix} \quad 6 \times 6$$

where

$$\nu_{\theta r} = \nu_{r\theta} E_\theta / E_r \quad , \quad (9)$$

$$\nu_{z\theta} = \nu_{\theta z} E_z / E_\theta \quad , \quad (10)$$

$$\nu_{rz} = \nu_{zr} E_r / E_z \quad , \quad (11)$$

$$\Delta = 1 - \nu_{r\theta}\nu_{\theta r} - \nu_{\theta z}\nu_{z\theta} - \nu_{zr}\nu_{rz} - \nu_{r\theta}\nu_{\theta z}\nu_{zr} - \nu_{rz}\nu_{\theta r}\nu_{z\theta} \quad . \quad (12)$$

3. Calculate the elastic constants matrix in element geometric axes:

$$[E_g] = [T_{eo}]^T [E_m] [T_{eo}] \quad . \quad (13)$$

8.23.5 Stiffness Matrix Generation (Subroutine KTRIA of Module EMG)

1. Generate the element stiffness matrix in field coordinates:

$$[K_n] = \begin{bmatrix} \tilde{K}_{11} & \tilde{K}_{12} & \tilde{K}_{13} & \tilde{K}_{14} & \tilde{K}_{15} & \tilde{K}_{16} & \tilde{K}_{17} & \tilde{K}_{18} & \tilde{K}_{19} \\ & \tilde{K}_{22} & \tilde{K}_{23} & \tilde{K}_{24} & \tilde{K}_{25} & \tilde{K}_{26} & \tilde{K}_{27} & \tilde{K}_{28} & \tilde{K}_{29} \\ & & \tilde{K}_{33} & \tilde{K}_{34} & \tilde{K}_{35} & \tilde{K}_{36} & \tilde{K}_{37} & \tilde{K}_{38} & \tilde{K}_{39} \\ & & & \tilde{K}_{44} & \tilde{K}_{45} & \tilde{K}_{46} & \tilde{K}_{47} & \tilde{K}_{48} & \tilde{K}_{49} \\ & & & & \tilde{K}_{55} & \tilde{K}_{56} & \tilde{K}_{57} & \tilde{K}_{58} & \tilde{K}_{59} \\ & & & & & \tilde{K}_{66} & \tilde{K}_{67} & \tilde{K}_{68} & \tilde{K}_{69} \\ & & & & & & \tilde{K}_{77} & \tilde{K}_{78} & \tilde{K}_{79} \\ & & & & & & & \tilde{K}_{88} & \tilde{K}_{89} \\ & & & & & & & & \tilde{K}_{99} \end{bmatrix} \quad \begin{matrix} \\ \\ \\ \text{Symmetric} \\ \\ \\ \end{matrix} \quad (14)$$

THE TRIAX ELEMENT

where

$$\begin{aligned}
 \tilde{K}_{11} &= (E_{33} + n^2 E_{55}) \delta_{-1,0} \\
 \tilde{K}_{12} &= (E_{13} + E_{33} + n^2 E_{55}) \delta_{0,0} \\
 \tilde{K}_{13} &= (E_{33} + n^2 E_{55}) \delta_{-1,1} + E_{34} \delta_{0,0} \\
 \tilde{K}_{14} &= (E_{33} + E_{55}) n \delta_{-1,0} \\
 \tilde{K}_{15} &= n E_{33} \delta_{0,0} \\
 \tilde{K}_{16} &= (E_{33} + E_{55}) n \delta_{-1,1} - n E_{56} \delta_{0,0} \\
 \tilde{K}_{17} &= n^2 E_{56} \delta_{-1,0} \\
 \tilde{K}_{18} &= (E_{34} + n^2 E_{56}) \delta_{0,0} \\
 \tilde{K}_{19} &= E_{23} \delta_{0,0} + n^2 E_{56} \delta_{-1,1} \\
 \tilde{K}_{22} &= ((E_{11} + 2E_{13} + E_{33}) + n^2 E_{55}) \delta_{1,0} \\
 \tilde{K}_{23} &= (E_{13} + E_{33} + n^2 E_{55}) \delta_{0,1} + (E_{14} + E_{34}) \delta_{1,0} \\
 \tilde{K}_{24} &= (E_{13} + E_{33} + E_{55}) n \delta_{0,0} \\
 \tilde{K}_{25} &= (E_{13} + E_{33}) n \delta_{1,0} \\
 \tilde{K}_{26} &= (E_{13} + E_{33} + E_{55}) n \delta_{0,1} - n E_{56} \delta_{1,0} \\
 \tilde{K}_{27} &= n^2 E_{56} \delta_{0,0} \\
 \tilde{K}_{28} &= (E_{14} + E_{34} + n^2 E_{56}) \delta_{1,0} \\
 \tilde{K}_{29} &= (E_{12} + E_{23}) \delta_{1,0} + n^2 E_{56} \delta_{0,1} \\
 \tilde{K}_{33} &= (E_{33} + n^2 E_{55}) \delta_{-1,2} + E_{44} \delta_{1,0} + 2E_{34} \delta_{0,1} \\
 \tilde{K}_{34} &= (E_{33} + E_{55}) n \delta_{-1,1} + n E_{34} \delta_{0,0} \\
 \tilde{K}_{35} &= n E_{33} \delta_{0,1} + n E_{34} \delta_{1,0} \\
 \tilde{K}_{36} &= (E_{33} + E_{55}) n \delta_{-1,2} + (E_{34} - E_{56}) n \delta_{0,1} \\
 \tilde{K}_{37} &= n^2 E_{56} \delta_{-1,1} \\
 \tilde{K}_{38} &= E_{44} \delta_{1,0} + (E_{34} + n^2 E_{56}) \delta_{0,1} \\
 \tilde{K}_{39} &= E_{23} \delta_{0,1} + E_{24} \delta_{1,0} + n^2 E_{56} \delta_{-1,2} \\
 \tilde{K}_{44} &= (E_{55} + n^2 E_{33}) \delta_{-1,0} \\
 \tilde{K}_{45} &= n^2 \delta_{0,0} E_{33} \\
 \tilde{K}_{46} &= (E_{55} + n^2 E_{33}) \delta_{-1,1} - \delta_{0,0} E_{56} \\
 \tilde{K}_{47} &= n \delta_{-1,0} E_{56} \\
 \tilde{K}_{48} &= n \delta_{0,0} (E_{34} + E_{56}) \\
 \tilde{K}_{49} &= (\delta_{0,0} E_{23} + \delta_{-1,1} E_{56}) n \\
 \tilde{K}_{55} &= n^2 \delta_{1,0} E_{33}
 \end{aligned}$$

STRUCTURAL ELEMENT DESCRIPTIONS

$$\begin{aligned}
 \tilde{K}_{56} &= n^2 \delta_{0,1} E_{33} \\
 \tilde{K}_{57} &= 0 \\
 \tilde{K}_{58} &= n \delta_{1,0} E_{34} \\
 \tilde{K}_{59} &= n \delta_{1,0} E_{23} \\
 \tilde{K}_{66} &= (E_{55} + n^2 E_{33}) \delta_{-1,2} + E_{66} \delta_{1,0} - 2 \delta_{0,1} E_{56} \\
 \tilde{K}_{67} &= (\delta_{-1,1} E_{56} - \delta_{0,0} E_{66}) n \\
 \tilde{K}_{68} &= (\delta_{0,1} (E_{34} + E_{56}) - \delta_{1,0} E_{66}) n \\
 \tilde{K}_{69} &= (\delta_{0,1} (E_{23} - E_{66}) + \delta_{-1,2} E_{56}) n \\
 \tilde{K}_{77} &= n^2 \delta_{-1,0} E_{66} \\
 \tilde{K}_{78} &= n^2 E_{66} \delta_{0,0} \\
 \tilde{K}_{79} &= n^2 \delta_{-1,1} E_{66} \\
 \tilde{K}_{88} &= (E_{44} + n^2 E_{66}) \delta_{1,0} \\
 \tilde{K}_{89} &= E_{24} \delta_{1,0} + n^2 \delta_{0,1} E_{66} \\
 \tilde{K}_{99} &= E_{22} \delta_{1,0} + n^2 \delta_{-1,2} E_{66}
 \end{aligned}$$

where E_{ij} is an element of $[E_g]$. Each term must be multiplied by 2π if $n=0$ and by π if $n>0$.

2. Transform the element stiffness matrix from field coordinates to grid point degrees of freedom:

$$[\bar{K}] = [H_{Bq}]^T [\tilde{K}] [H_{Bq}] \quad (15)$$

3. The 3 by 3 partitions $[K^3_{pj}]$ of $[K]$, corresponding to the pivot point p , are extracted from $[\bar{K}]$, then for insertion, these 3 by 3 partitions are expanded to 6 by 6 partitions:

$$[K_{pj}] = \begin{bmatrix} \bar{K} & \vdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \vdots & 0 \end{bmatrix} \quad (16)$$

8.23.6 Mass Matrix Calculations (Subroutine MSTRIA of Module EMG)

1. Generate the consistent mass matrix in field coordinates:

THE TRIAAX ELEMENT

$$\tilde{M} = \rho \begin{bmatrix} [\Delta] & & \\ & [\Delta] & \\ & & [\Delta] \end{bmatrix} \quad (17)$$

(9 x 9)

where

$$[\Delta] = \begin{bmatrix} A_{1,0} & A_{2,0} & A_{1,1} \\ & A_{3,0} & A_{2,1} \\ & & A_{1,2} \end{bmatrix} \quad (18)$$

and

$$\begin{aligned} \rho &= \rho \cdot 2\pi \text{ for } N = 0 \\ \rho &= \rho \cdot \pi \text{ for } N > 0 \end{aligned}$$

The integrals of the triangular ring $A_{1,0}$, $A_{2,0}$, $A_{1,1}$, $A_{3,0}$, $A_{2,1}$ and $A_{1,2}$ are computed by subroutine DKL and are equivalent to the integrals δ_{ij} .

2. Transform the mass matrix from field coordinates to grid point degrees of freedom:

$$[\tilde{M}] = [H_{Bq}]^T [M] [H_{Bq}] \quad (19)$$

3. The 6 by 6 partitions, $[M_{pj}]$, are calculated as in Equation 16.

4. Generate the Lumped Mass

$$\begin{aligned} M &= 2\pi * \rho * RL * A \\ \text{for } N > 0 \quad M &= M/2 \end{aligned} \quad (20)$$

where

$$\begin{aligned} A &= 1/2 * (R_1 * (Z_2 - Z_3) + R_2 * (Z_3 - Z_1) + R_3 * (Z_1 - Z_2)) \\ RL &= 1/3 \sum_{i=1}^3 R_i \end{aligned}$$

STRUCTURAL ELEMENT DESCRIPTIONS

8.23.7 Thermal Load Calculations (Subroutine TRITEM of Module SSG1)

1. Form the vector of thermal strains:

$$\{\alpha\} = T_{avg} \begin{Bmatrix} \alpha_r \\ \alpha_\theta \\ \alpha_z \\ 0 \\ 0 \\ 0 \end{Bmatrix} \quad (6 \times 1) \quad (21)$$

where T_{avg} is the average loading temperature at the grid points:

$$\begin{aligned} \text{for } N = 0 \quad T_{avg} &= \left(\frac{(T_1 + T_2 + T_3)}{3} - T_0 \right) * 2\pi \\ \text{for } N > 0 \quad T_{avg} &= \frac{(T_1 + T_2 + T_3)}{3} * \pi \end{aligned}$$

2. Compute thermal load vector in grid point degrees of freedom:

$$\{\bar{F}_T\} = [H_{Bq}]^T [\tilde{D}]^T [E_g] \{\alpha\} \quad (9 \times 1) \quad (22)$$

$$[\tilde{D}]^T = \begin{bmatrix} 0 & 0 & A_{0,0} & 0 & -NA_{0,0} & 0 \\ A_{1,0} & 0 & A_{1,0} & 0 & -NA_{1,0} & 0 \\ 0 & 0 & A_{0,1} & A_{1,0} & -NA_{0,1} & 0 \\ 0 & 0 & NA_{0,0} & 0 & -A_{0,0} & 0 \\ 0 & 0 & NA_{1,0} & 0 & 0 & 0 \\ 0 & 0 & NA_{0,1} & 0 & -A_{0,1} & A_{1,0} \\ 0 & 0 & 0 & 0 & 0 & -NA_{0,0} \\ 0 & 0 & 0 & A_{1,0} & 0 & -NA_{1,0} \\ 0 & A_{1,0} & 0 & 0 & 0 & -NA_{0,1} \end{bmatrix} \quad (23)$$

The integrals of the triangular ring, $A_{0,0}$, $A_{0,1}$, and $A_{1,0}$, are computed by subroutine AIS.

3. Each partition, $\{F_T^3\}$, of length 3 of $\{F_T\}$ is transformed to global coordinates by

$$\{F_T^3\}_g = [T_i]^T \{F_T^3\} \quad (24)$$

THE TRIAX ELEMENT

These vectors are added to the overall load vector, $\{P_g\}$.

8.23.8 Element Stress and Force Calculations (STRAX1, STRAX2, STRAX3 of module SDR2)

Phase 1 calculations are as follows:

1. Form the element stiffness matrix $[K]$ as in Section 8.25.5.
2. Compute the constants:

$$R_o = 1/3 \sum_{i=1}^3 R_{2i} \quad (25)$$

$$Z_o = 1/3 \sum_{i=1}^3 Z_{2i} \quad (26)$$

3. Form $[D_o]$ matrix

$$[D_o] = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \frac{1}{R_o} & 1 & \frac{Z_o}{R_o} & \frac{N}{R_o} & N & \frac{NZ_o}{R_o} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ \frac{-N}{R_o} & -N & \frac{-NZ_o}{R_o} & \frac{1}{-R_o} & 0 & \frac{-Z_o}{R_o} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & \frac{-N}{R_o} & -N & \frac{-NZ_o}{R_o} \end{bmatrix} \quad (27)$$

4. Compute the stress matrix

$$[\tilde{S}] = [E_g][D_o][H_{Bq}] \quad (28)$$

5. Compute the thermal stress vector

$$\{T_s\} = [E_g] \{\tilde{\alpha}\} \quad (29)$$

STRUCTURAL ELEMENT DESCRIPTIONS

Phase 2 Calculations are as follows:

1. The elements in the EST are ordered by harmonic and TRIAAX I.D. number. All harmonic elements for each TRIAAX are grouped together. When the harmonic, N, of an element is zero, it indicates that it is the first of a group of elements. Storage space is allotted for 15 by 14 vectors defining the element forces at three points and stresses on its centroid.
2. Extract the displacement vector $\{\Delta\}$, at the three translational components of the grid points from the global displacement vector.
3. Calculate the element forces:

$$\{P\} = [K]\{\Delta\} \quad (30)$$

4. Calculate the element stresses:

$$\{\sigma\} = [S]\{\Delta\} - T_d\{T_s\} \quad (31)$$

where

$$\begin{aligned} \text{for } N = 0 \quad T_d &= T_{avg} - T_o \\ \text{and for } N > 0 \quad T_d &= T_{avg} \end{aligned} \quad (32)$$

5. Sum of harmonic element forces at each grid point

$$\{P_i\} = \{P_o\} + \sum_{K=2}^{N+1} [C_o^{(sy)}] \{P(K)\} \quad (33)$$

For the symmetric case $sy=0$

For antisymmetric case $sy=1$

$$[C_o^{(0)}] = \begin{bmatrix} \cos N\theta_i & 0 & 0 \\ 0 & \sin N\theta_i & 0 \\ 0 & 0 & \cos N\theta_i \end{bmatrix} \quad (34)$$

$$[C_o^{(1)}] = \begin{bmatrix} \sin N\theta_i & & \\ & -\cos N\theta_i & \\ & & \sin N\theta_i \end{bmatrix} \quad (35)$$

THE TRIAX ELEMENT

6. Sum, the harmonic element stresses computation

$$\sigma_h = \sigma_o + \sum_{k=2}^{N+1} [c_o^{(sy)}] \{\sigma^{(K)}\} \quad (36)$$

where

For symmetric case $sy=0$

For antisymmetric case $sy=1$

$$[c_o^{(0)}] = \begin{bmatrix} \cos N\theta_i & 0 & 0 & 0 & 0 & 0 \\ 0 & \cos N\theta_i & 0 & 0 & 0 & 0 \\ 0 & 0 & \cos N\theta_i & 0 & 0 & 0 \\ 0 & 0 & 0 & \cos N\theta_i & 0 & 0 \\ 0 & 0 & 0 & 0 & \sin N\theta_i & 0 \\ 0 & 0 & 0 & 0 & 0 & \sin N\theta_i \end{bmatrix} \quad (37)$$

$$[c_o^{(1)}] = \begin{bmatrix} \sin N\theta_i & 0 & 0 & 0 & 0 & 0 \\ 0 & \sin N\theta_i & 0 & 0 & 0 & 0 \\ 0 & 0 & \sin N\theta_i & 0 & 0 & 0 \\ 0 & 0 & 0 & \sin N\theta_i & 0 & 0 \\ 0 & 0 & 0 & 0 & -\cos N\theta_i & 0 \\ 0 & 0 & 0 & 0 & 0 & -\cos N\theta_i \end{bmatrix} \quad (38)$$

THE TRIM6 ELEMENT

8.24 TRIM6: LINEAR STRAIN TRIANGULAR ELEMENT

8.24.1 Input Data for TRIM6 Element

1. EST entries for TRIM6 are:

<u>Symbol</u>	<u>Description</u>
EID	Element Identification Number
SIL ₁ , SIL ₂ , . . . , SIL ₆	Scalar indices of connected grid points
θ	Anisotropic material orientation angle
Mat ID	Material Identification Number
T1, T3, T5	Thickness of corner grid points
μ	Nonstructural mass per unit area
$\left. \begin{array}{l} N_i \\ X_i \\ Y_i \\ Z_i \end{array} \right\} i = 1, 6$	Local coordinate system numbers and location coordinates in the basic system for the connected grid points
T01, T02, T03, T04, T05, T06	Temperatures at the grid points

2. Coordinate system data

The numbers N_i , X_i , Y_i and Z_i are used to calculate 3 by 3 basic-to-global coordinate transformation matrices $[T_i]$ for points $i = 1, 2, 3, 4, 5$ and 6.

3. Material data

<u>Symbol</u>	<u>Description</u>
[G]	3 x 3 stress-strain matrix
ρ	Mass density
$\alpha_x, \alpha_y, \alpha_{xy}$	Thermal expansion coefficients
T0	Reference temperature
g_e	Structural damping coefficient
$\sigma_t, \sigma_c, \sigma_s$	Stress limits for tension, compression and shear

STRUCTURAL ELEMENT DESCRIPTIONS

8.24.2 Basic Equations for TRIM6

1. The element coordinate system is defined by the following equations:

$$\{V_{13}\} = \begin{Bmatrix} X_3 - X_1 \\ Y_3 - Y_1 \\ Z_3 - Z_1 \end{Bmatrix} \quad (1)$$

$$\{V_{15}\} = \begin{Bmatrix} X_5 - X_1 \\ Y_5 - Y_1 \\ Z_5 - Z_1 \end{Bmatrix} \quad (2)$$

$$\{i\} = \frac{\{V_{13}\}}{|\{V_{13}\}|} \quad (3)$$

$$\{k\} = \frac{\{i\} \times \{V_{13}\}}{|\{i\} \times \{V_{13}\}|} \quad (4)$$

$$\{j\} = \{k\} \times \{i\} \quad (5)$$

2. The displacement transformation matrix from basic coordinates to in-plane coordinates is:

$$[E]^T = \begin{bmatrix} i_1 & i_2 & i_3 \\ j_1 & j_2 & j_3 \end{bmatrix} \quad (6)$$

3. The local (element) coordinate system of the element is as follows:

The x-axis is obtained by joining grid points 1 and 3 of the element.

The y-axis is the perpendicular from grid point 5 to the x-axis (line joining grid points 1 and 3).

Depending upon the location of grid point 5 relative to grid points 1 and 3, 3 cases of triangle orientation are possible: (refer to Figure 1)

Case I: Acute angles at grid points 1 and 3

$$c = |\{i\} \times \{V_{15}\}| \quad (7)$$

$$b = \{i\} \cdot \{V_{15}\} \quad (8)$$

$$a = |\{V_{13}\}| - b \quad (9)$$

THE TRIM6 ELEMENT

Coordinates of points are;

$$x_1 = -b, x_2 = \frac{a-b}{2}, x_3 = a, x_4 = \frac{a}{2}, x_5 = 0 \text{ and } x_6 = -\frac{b}{2} \quad (10)$$

$$y_1 = 0, y_2 = 0, y_3 = 0, y_4 = \frac{c}{2}, y_5 = c \text{ and } y_6 = \frac{c}{2} \quad (11)$$

Case II: Obtuse angle at grid point 3

$$c = |\{i\} \times \{V_{15}\}| \quad (12)$$

$$b = \{i\} \cdot \{V_{15}\} \quad (13)$$

$$a = b - |\{V_{13}\}| \quad (14)$$

Coordinates of points are;

$$x_1 = -b, x_2 = \frac{-a-b}{2}, x_3 = -a, x_4 = -\frac{a}{2}, x_5 = 0 \text{ and } x_6 = -\frac{b}{2} \quad (15)$$

$$y_1 = 0, y_2 = 0, y_3 = 0, y_4 = \frac{c}{2}, y_5 = c \text{ and } y_6 = \frac{c}{2} \quad (16)$$

Case III: Obtuse angle at grid point 1

$$c = |\{i\} \times \{V_{15}\}| \quad (17)$$

$$b = \{i\} \cdot \{V_{15}\} \quad (18)$$

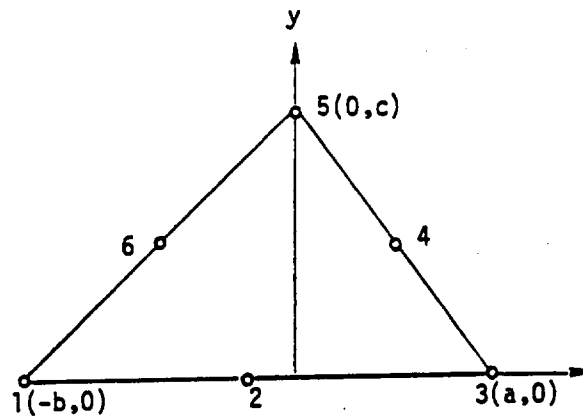
$$a = |\{V_{13}\}| + b \quad (19)$$

Coordinates of points are;

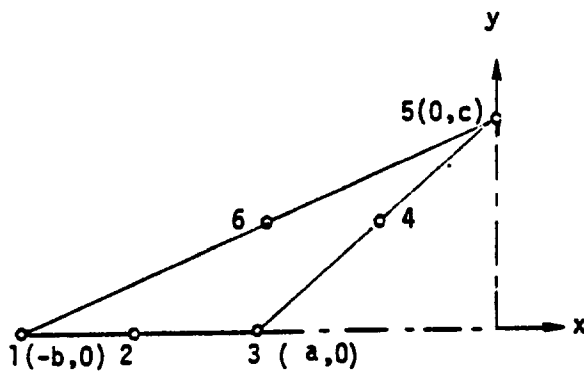
$$x_1 = b, x_2 = \frac{a+b}{2}, x_3 = a, x_4 = \frac{a}{2}, x_5 = 0 \text{ and } x_6 = \frac{b}{2} \quad (20)$$

$$y_1 = 0, y_2 = 0, y_3 = 0, y_4 = \frac{c}{2}, y_5 = c \text{ and } y_6 = \frac{c}{2} \quad (21)$$

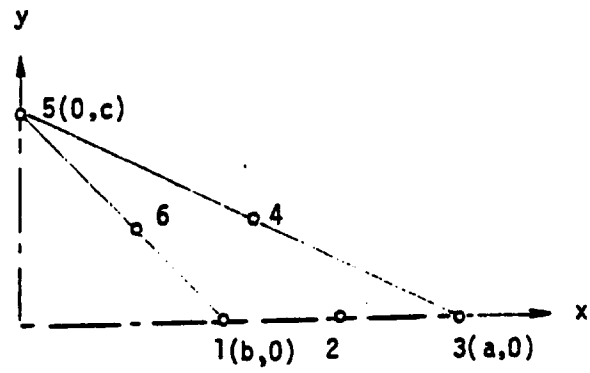
STRUCTURAL ELEMENT DESCRIPTIONS



Case I. Acute angles at grid points 1, 3 and 5.



Case II. Obtuse angle at grid point 3.



Case III. Obtuse angle at grid point 1.

Figure 1. Triangular element shapes.

THE TRIM6 ELEMENT

4. The matrix $[H]$ relating grid point displacements and the generalized coordinates (in the equation $\{u\} = [H] \{a\}$) is given by;

$$[H] = \begin{bmatrix} H_1 & & 0 \\ & \ddots & \\ 0 & & H_1 \end{bmatrix} \quad (22)$$

where;

$$[H_1] = \begin{bmatrix} 1 & x_1 & y_1 & x_1^2 & x_1 y_1 & y_1^2 \\ 1 & x_2 & y_2 & x_2^2 & x_2 y_2 & y_2^2 \\ 1 & x_3 & y_3 & x_3^2 & x_3 y_3 & y_3^2 \\ 1 & x_4 & y_4 & x_4^2 & x_4 y_4 & y_4^2 \\ 1 & x_5 & y_5 & x_5^2 & x_5 y_5 & y_5^2 \\ 1 & x_6 & y_6 & x_6^2 & x_6 y_6 & y_6^2 \end{bmatrix} \quad (23)$$

5. The matrix $[B]$ relating strain vector to the generalized coordinates (in the equation $\{\epsilon\} = [B] \{a\}$) is given by;

$$[B] = \begin{bmatrix} 0 & 1 & 0 & 2x & y & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & x & 2y \\ 0 & 0 & 1 & 0 & x & 2y & 0 & 1 & 0 & 2x & y & 0 \end{bmatrix} \quad (24)$$

8.24.3 Stiffness Matrix Calculation for TRIM6 (Subroutine KTRM6S and KTRM6D)

1. The polynomial expressions for variation of u , v and t within the element are:

$$u = \sum_{i=1}^{12} a_i x^{m_i} y^{n_i} \quad (25)$$

STRUCTURAL ELEMENT DESCRIPTIONS

$$v = \sum_{i=1}^{12} b_i x^{p_i} y^{q_i} \quad (26)$$

$$t = \sum_{i=1}^3 c_i x^{r_i} y^{s_i} \quad (27)$$

The values of $m_i, n_i, p_i, q_i, r_i, s_i$ are

$$m_1 = 0, m_2 = 1, m_3 = 0, m_4 = 2, m_5 = 1, m_6 = 0, m_7 \text{ to } m_{12} = 0, \quad (28)$$

$$n_1 = 0, n_2 = 0, n_3 = 1, n_4 = 0, n_5 = 1, n_6 = 2, n_7 \text{ to } n_{12} = 0, \quad (29)$$

$$p_1 \text{ to } p_6 = 0, p_7 = 0, p_8 = 1, p_9 = 0, p_{10} = 2, p_{11} = 1, p_{12} = 0, \quad (30)$$

$$q_1 \text{ to } q_6 = 0, q_7 = 0, q_8 = 0, q_9 = 1, q_{10} = 0, q_{11} = 1, q_{12} = 2, \quad (31)$$

$$r_1 = 0, r_2 = 1, r_3 = 0, \quad (32)$$

$$s_1 = 0, s_2 = 0, s_3 = 1, \quad (33)$$

The coefficients a_1 to a_6 and b_7 to b_{12} are generalized coordinates of the element and can be evaluated once the displacement vector is known.

$$a_7 \text{ to } a_{12} = 0; b_1 \text{ to } b_6 = 0, \quad (34)$$

The coefficients c_1, c_2 and c_3 can be evaluated from the specified thicknesses t_1, t_3 and t_5 of the three corner grid points and the geometric dimensions a, b and c of the element

$$c_1 = \frac{t_1 a + t_3 b}{(a + b)} \quad (35)$$

$$c_2 = \frac{t_3 - t_1}{(a + b)} \quad (36)$$

$$c_3 = \frac{1}{c} (t_5 - c_1) \quad (37)$$

THE TRIM6 ELEMENT

The elements of the symmetric portion of the stress-strain matrix $[G_e]$ are denoted by $G_{11}, G_{12}, G_{13}, G_{22}, G_{23}$ and G_{33} .

A formula for the integral of the type $x^m y^n$ taken over the area of the element is;

$$\iint x^m y^n dx dy = F(m,n) = c^{n+1} \{a^{m+1} - (-b)^{m+1}\} \frac{m!n!}{(m+n+2)!} \quad (38)$$

The equation used in the stiffness matrix generation in generalized coordinates is;

$$\begin{aligned} (k_{ij})_{\text{gen}} = & \sum_{k=1}^3 c_k [G_{11} m_i m_j F(m_i + m_j + r_k - 2, n_i + n_j + s_k) \\ & + G_{22} q_i q_j F(p_i + p_j + r_k, q_i + q_j + s_k - 2) \\ & + G_{33} \{n_i n_j F(m_i + m_j + r_k, n_i + n_j + s_k - 2) \\ & + p_i p_j F(p_i + p_j + r_k - 2, q_i + q_j + s_k)\} \\ & + (G_{33} n_i p_j + G_{12} m_i q_j) F(m_i + p_j + r_k - 1, n_i + q_j + s_k - 1) \\ & + (G_{33} n_j p_i + G_{12} m_j q_i) F(m_j + p_i + r_k - 1, n_j + q_i + s_k - 1) \\ & + G_{13} \{m_j n_i + m_i n_j\} F(m_i + m_j + r_k - 1, n_i + n_j + s_k - 1) \\ & + m_j p_i F(m_j + p_i + r_k - 2, n_j + q_i + s_k) \\ & + m_i p_j F(m_i + p_j + r_k - 2, n_i + q_j + s_k)\} \\ & + G_{23} \{(p_i q_j + p_j q_i) F(p_i + p_j + r_k - 1, q_i + q_j + s_k - 1) \\ & + n_i q_j F(m_i + p_j + r_k, n_i + q_j + s_k - 2) \\ & + n_j q_i F(m_j + p_i + r_k, n_j + q_i + s_k - 2)\}] \end{aligned}$$

The stiffness matrix in global coordinates is;

STRUCTURAL ELEMENT DESCRIPTIONS

$$[k] = [E] [T]^T [H]^{-1} [k]_{\text{gen}} [H]^{-1} [T] [E]^T \text{ where;}$$

$$\text{where; } [H]^{-1} = [H]^{-1}$$

For use in the overall structural matrix, the 3 x 3 k_{ij} partition of the stiffness matrix

$[k]$ corresponding to grid point i and connection point j is expanded to 6 x 6 to form;

$$k_{ij} = \begin{bmatrix} k_{ij} & \vdots & 0 \\ - & - & - \\ 0 & \vdots & 0 \end{bmatrix} \quad (41)$$

8.24.4 Mass Matrix Calculation for the TRIM6 Element (calculated in the stiffness subroutine KTRM6S and KTRM6D)

The mass is generated by the following algorithm;

$$\{V_{13}\} = \begin{Bmatrix} X_3 - X_1 \\ Y_3 - Y_1 \\ Z_3 - Z_1 \end{Bmatrix} \quad (42)$$

$$\{V_{15}\} = \begin{Bmatrix} X_5 - X_1 \\ Y_5 - Y_1 \\ Z_5 - Z_1 \end{Bmatrix} \quad (43)$$

The area is;

$$A = \frac{1}{2} |\{V_{13}\} \times \{V_{15}\}| \quad (44)$$

Volume;

$$V = c_1 F(0,0) + c_2 F(1,0) + c_3 F(0,1) \quad (45)$$

where c_1, c_2, c_3 (see equation (33), (34) and (35)) are the constants in the thickness equation of the element (equation (25)) and zero factorial has a value of 1. The mass at each point is;

$$m = \frac{1}{6} (\rho V + A u) \quad (46)$$

For each point the diagonal mass matrix in the element coordinate system at all grid points is;

THE TRIM6 ELEMENT

$$[m_i] = \begin{bmatrix} m & & & 0 \\ & m & & \\ & & 0 & \\ & & & 0 \\ 0 & & & & 0 \end{bmatrix} \quad i = 1, 2, \dots \text{ and } 6 \quad (47)$$

so that $[M_{ee}]$ the element mass matrix has $[m_i]$ matrices arranged diagonally. The mass matrix in global coordinate system is obtained as;

$$[M_{gg}] = [E] [T]^T [M_{ee}] [T] [E]^T \quad (48)$$

The consistent mass matrix is given by

$$\begin{aligned} [M_{ij}]_{\text{gen}} &= \sum_{k=1}^3 \rho \iint c_k x^{m_i+m_j+r_k} y^{n_i+n_j+s_k} dx dy \\ &+ \mu \iint x^{m_i+m_j} y^{n_i+n_j} dx dy \\ &= \rho \sum_{k=1}^3 \left\{ c_k F(m_i+m_j+r_k, n_i+n_j+s_k) \right\} + \mu F(m_i+m_j, n_i+n_j) \end{aligned} \quad (48a)$$

where $[M_{ij}]_{\text{gen}}$ represents the j^{th} element of the i^{th} row of the generalized mass matrix.

The mass matrix in global coordinates is

$$[M] = [E] [T]^T [H^{-1}]^T [M]_{\text{gen}} [H^{-1}] [T] [E]^T \quad (48b)$$

8.24.5 Element Load Calculations for TRIM6 (Subroutine TL0DM6)

The temperature within the element is assumed to vary bilinearly;

$$T = \sum_{i=1}^3 d_i x^{r_i} y^{s_i} \quad (49)$$

with;

$$r_1 = 0; r_2 = 1; r_3 = 0 \quad (50)$$

STRUCTURAL ELEMENT DESCRIPTIONS

and;

$$s_1 = 0; s_2 = 0; s_3 = 1 \quad (51)$$

The coefficients d_1 , d_2 and d_3 are evaluated from the specified temperatures T_{01} , T_{03} , and T_{05} at the three corner grid points (obtained from the GPTT data block) and the reference temperature T_0 of the element;

$$d_1 = \frac{T_{01}a + T_{03}b}{(a + b)} \quad (52)$$

$$d_2 = \frac{T_{03} - T_{01}}{(a + b)} \quad (53)$$

$$d_3 = \frac{1}{c} [T_{05} - d_1] \quad (54)$$

THE TRIM6 ELEMENT

THIS PAGE HAS BEEN LEFT BLANK INTENTIONALLY.

STRUCTURAL ELEMENT DESCRIPTIONS

The constant d_1 is modified by the reference temperature, T_0 , $d_1 = d_1 - T_0$. The i th element of the generalized load vector $\{P_{gen}\}$ is;

$$\begin{aligned} \{P_i\}_{gen} = & \sum_{k=1}^3 \sum_{\ell=1}^3 c_k d_\ell \left[G_{11}^1 m_i F(m_i + r_k + t_\ell - 1, n_i + s_k + u_\ell) \right. \\ & + G_{22}^1 q_i F(p_i + r_k + t_\ell, q_i + s_k + u_\ell - 1) \\ & + G_{33}^1 \{n_i F(m_i + r_k + t_\ell, n_i + s_i + u_\ell - 1) \\ & \left. + p_i F(p_i + r_k + t_\ell - 1, q_i + s_k + u_\ell)\} \right] \end{aligned} \quad (55)$$

where;

$$G_{11}^1 = G_{11} \alpha_2 + G_{12} \alpha_2 + G_{13} \alpha_{12} \quad (56)$$

$$G_{22}^1 = G_{12} \alpha_1 + G_{22} \alpha_2 + G_{23} \alpha_{12} \quad (57)$$

$$G_{33}^1 = G_{13} \alpha_1 + G_{23} \alpha_2 + G_{33} \alpha_{12} \quad (58)$$

The generalized equivalent load vector $\{P_{gen}\}$ is transformed to load vector $\{P_e\}$ in local element coordinates and to load vector $\{P_g\}$ in global grid-point coordinates by the following transformations:

$$\{P_e\} = [H']^T \{P_{gen}\} \text{ where } [H'] = [H]^{-1} \quad (59)$$

$$\{P_g\} = [E] [T]^T \{P_e\} \quad (60)$$

$\{P_g\}$ is a 18×1 vector.

The forces are placed in the PG load vector data block.

8.24.6 Element Stress Calculations for TRIM6 Element (Subroutine STRM61 and STRM62 of module SDR2)

1. The relationship between strain and generalized coefficients is;

THR TRIM6 ELEMENT

$$\{\epsilon\} = [B] \{a\} \quad (61)$$

where;

$$[B] = \begin{bmatrix} 0 & 1 & 0 & 2x & y & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & x & 2y \\ 0 & 0 & 1 & 0 & x & 2y & 0 & 1 & 0 & 2x & y & 0 \end{bmatrix} \quad (62)$$

The transformation from displacements to stress is:

$$[S_i] = [G_e] [B] [H]^{-1} [E]^T [T] \quad (63)$$

The temperature to stress relation is;

$$\{S_t\} = -[G_e] \{\alpha\} \quad (64)$$

where;

$$\{\alpha\} = \alpha \begin{Bmatrix} 1 \\ 1 \\ 0 \end{Bmatrix} \quad (65)$$

for isotropic materials. $\{\alpha\}$ is input by the user for anisotropic materials and corrected for material angle by;

$$a = [V] \{\alpha_m\} \quad (66)$$

2. Calculations performed by STRM62 (Phase 2 calculations) . The equation for stress is;

$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \sigma_{xy} \end{Bmatrix} = \sum_{i=1}^6 [S_i] \{u_g\} + \{S_t\} (T_j - T_0) \quad (67)$$

where T_j is the loading temperature for the point where stress is evaluated (3 corner grid points and centroid) and is obtained from the GPTT data block. The temperature of the centroid is taken as the average of the grid point temperatures.

STRUCTURAL ELEMENT DESCRIPTIONS

The principal stresses are;

$$\sigma_1 = \left(\frac{\sigma_x + \sigma_y}{2} \right) + \sqrt{\left(\frac{\sigma_x - \sigma_y}{2} \right)^2 + \sigma_{xy}^2} \quad (68)$$

$$\sigma_2 = \left(\frac{\sigma_x + \sigma_y}{2} \right) - \sqrt{\left(\frac{\sigma_x - \sigma_y}{2} \right)^2 + \sigma_{xy}^2} \quad (69)$$

$$\theta = \frac{1}{2} \arctan \left(\frac{2\sigma_{xy}}{\sigma_x - \sigma_y} \right) \text{ in degrees} \quad (70)$$

where θ is limited to: $-90^\circ \leq \theta \leq 90^\circ$.

The maximum shear is;

$$\tau = \sqrt{\left(\frac{\sigma_x - \sigma_y}{2} \right)^2 + \sigma_{xy}^2} \quad (71)$$

The stresses are output for four points for every element: three corner grid points and the centroid.

THE TRPLT1 ELEMENT

8.25 TRPLT1 HIGHER ORDER PLATE-BENDING ELEMENT

8.25.1 Input Data for TRPLT1 Element

1. EST entries for TRPLT1 are:

<u>Symbol</u>	<u>Description</u>
EID	Element Identification Number
SIL ₁ , SIL ₂ , . . . , SIL ₆	Scalar indices of connected grid points
θ	Anisotropic material orientation angle
Mat ID _b	Material Identification Number for bending
Mat ID _s	Material Identification Number for shear
I1, I3 and I5	Area moment of inertia per unit width at corner grid points $I1 = \frac{t_1^3}{12}$, $I3 = \frac{t_3^3}{12}$, $I5 = \frac{t_5^3}{12}$
TS1, TS3 and TS5	Effective thickness for transverse shear at corner grid points
u	Nonstructural mass per unit area
Z11, Z21, Z13, Z23, Z15 and Z25	Distances Z1 and Z2 for stress calculation at three corner points
$\left. \begin{matrix} N_i \\ X_i \\ Y_i \\ Z_i \end{matrix} \right\} i = 1, 6$	Local coordinate system numbers and location coordinates in the basic system for the connected grid points
TEMP	Element temperature

2. Coordinate system data

The numbers N_i , X_i and Z_i are used to calculate the three by three basic-to-global coordinate transformation matrices $[T_i]$ for points $i = 1, 2, 3, 4, 5$ and 6 (via subroutines TRANSD or TRANSS).

STRUCTURAL ELEMENT DESCRIPTIONS

3. Material data

	<u>Symbol</u>	<u>Description</u>
	[G]	3 x 3 stress-strain matrix
	ρ	Mass density
	$\alpha_x, \alpha_y, \alpha_{xy}$	Thermal expansion coefficients
	T0	Reference temperature
For Mat ID _b	g_e	Structural damping coefficient
	$\sigma_t, \sigma_c, \sigma_s$	Stress limits for tension, compression and shear
For Mat ID _s	G_s	Shear coefficient

8.25.2 Basic Equation for TRPLT1

1. The element coordinate system is defined by the following equations:

$$\{V_{13}\} = \begin{Bmatrix} X_3 - X_1 \\ Y_3 - Y_1 \\ Z_3 - Z_1 \end{Bmatrix} \quad (1)$$

$$\{V_{15}\} = \begin{Bmatrix} X_5 - X_1 \\ Y_5 - Y_1 \\ Z_5 - Z_1 \end{Bmatrix} \quad (2)$$

$$A = \frac{1}{2} |\{V_{13}\} \times \{V_{15}\}| \quad (3)$$

$$\{i\} = \frac{\{V_{13}\}}{|\{V_{13}\}|} \quad (4)$$

$$\{k\} = \frac{\{i\} \times \{V_{13}\}}{|\{i\} \times \{V_{13}\}|} \quad (5)$$

$$\{j\} = \{k\} \times \{i\} \quad (6)$$

2. The displacement transformation matrix from basic coordinates to in-plane coordinates is:

THE TRPLT1 ELEMENT

$$[E]^T = \begin{bmatrix} k_1 & k_2 & k_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & i_1 & i_2 & i_3 \\ 0 & 0 & 0 & j_1 & j_2 & j_3 \end{bmatrix} \quad (7)$$

3. The local (element) coordinate system of the element is as follows: The x-axis is obtained by joining grid points 1 and 3 of the element. The y-axis is the perpendicular from grid point 5 to the x-axis (line joining grid points 1 and 3).

Depending upon the location of grid point 5 relative to grid points 1 and 3, three cases of triangle orientation are possible: (refer to Figure 1 in Section 8.24).

Case I: Acute angles at grid points 1 and 3

$$c = |\{i\} \times \{V_{15}\}| \quad (8)$$

$$b = \{i\} \cdot \{V_{15}\} \quad (9)$$

$$a = |\{V_{13}\}| - b \quad (10)$$

Coordinates of points are;

$$x_1 = -b, x_2 = \frac{a-b}{2}, x_3 = a, x_4 = \frac{a}{2}, x_5 = 0 \text{ and } x_6 = -\frac{b}{2} \quad (11)$$

$$y_1 = 0, y_2 = 0, y_3 = 0, y_4 = \frac{c}{2}, y_5 = c \text{ and } y_6 = \frac{c}{2} \quad (12)$$

Case II: Obtuse angle at grid point 3

$$c = |\{i\} \times \{V_{15}\}| \quad (13)$$

$$b = \{i\} \cdot \{V_{15}\} \quad (14)$$

$$a = b - |\{V_{13}\}| \quad (15)$$

Coordinates of points are;

$$x_1 = -b, x_2 = -\frac{a-b}{2}, x_3 = 0, x_4 = -\frac{a}{2}, x_5 = 0 \text{ and } x_6 = -\frac{b}{2} \quad (16)$$

$$y_1 = 0, y_2 = 0, y_3 = 0, y_4 = \frac{c}{2}, y_5 = c \text{ and } y_6 = \frac{c}{2} \quad (17)$$

STRUCTURAL ELEMENT DESCRIPTIONS

Case III: Obtuse angle at grid point 1

$$c = |\{i\} \times \{V_{15}\}| \quad (18)$$

$$b = \{i\} \cdot \{V_{15}\} \quad (19)$$

$$a = |\{V_{13}\}| + b \quad (20)$$

Coordinates of points are;

$$x_1 = b, \quad x_2 = \frac{a+b}{2}, \quad x_3 = a, \quad x_4 = \frac{a}{2}, \quad x_5 = 0 \text{ and } x_6 = \frac{b}{2} \quad (21)$$

$$y_1 = 0, \quad y_2 = 0, \quad y_3 = 0, \quad y_4 = \frac{c}{2}, \quad y_5 = c \text{ and } y_6 = \frac{c}{2} \quad (22)$$

4. The matrix $[H]$ (for plates infinitely rigid in transverse shear) relating grid point displacements and the generalized coordinates (in the equation $\{u\} = [H] \{a\}$) is given by the matrix $[H]$.

[H] =

1	x_1	y_1	x_1^2	$x_1 y_1$	y_1^2	x_1^3	$x_1^2 y_1$	$x_1 y_1^2$	y_1^3	x_1^4	$x_1^3 y_1$	$x_1^2 y_1^2$	$x_1 y_1^3$	y_1^4	x_1^5	$x_1^3 y_1^2$	$x_1^2 y_1^3$	$x_1 y_1^4$	y_1^5
0	0	1	0	x_1	$2y_1$	0	x_1^2	$2x_1 y_1$	$3y_1^2$	0	x_1^3	$2x_1^2 y_1$	$3x_1 y_1^2$	$4y_1^3$	0	$2x_1^3 y_1$	$3x_1^2 y_1^2$	$4x_1 y_1^3$	$5y_1^4$
0	-1	0	$-2x_1$	$-y_1$	0	$-3x_1^2$	$-2x_1 y_1$	$-y_1^2$	0	$-4x_1^3$	$-3x_1^2 y_1$	$-2x_1 y_1^2$	$-y_1^3$	0	$-5x_1^4$	$-3x_1^2 y_1^2$	$-2x_1 y_1^3$	$-y_1^4$	0
1	x_2	y_2	x_2^2	$x_2 y_2$	y_2^2	x_2^3	$x_2^2 y_2$	$x_2 y_2^2$	y_2^3	x_2^4	$x_2^3 y_2$	$x_2^2 y_2^2$	$x_2 y_2^3$	y_2^4	x_2^5	$x_2^3 y_2^2$	$x_2^2 y_2^3$	$x_2 y_2^4$	y_2^5
0	0	1	0	x_2	$2y_2$	0	x_2^2	$2x_2 y_2$	$3y_2^2$	0	x_2^3	$2x_2^2 y_2$	$3x_2 y_2^2$	$4y_2^3$	0	$2x_2^3 y_2$	$3x_2^2 y_2^2$	$4x_2 y_2^3$	$5y_2^4$
0	-1	0	$-2x_2$	$-y_2$	0	$-3x_2^2$	$-2x_2 y_2$	$-y_2^2$	0	$-4x_2^3$	$-3x_2^2 y_2$	$-2x_2 y_2^2$	$-y_2^3$	0	$-5x_2^4$	$-3x_2^2 y_2^2$	$-2x_2 y_2^3$	$-y_2^4$	0
1	x_3	y_3	x_3^2	$x_3 y_3$	y_3^2	x_3^3	$x_3^2 y_3$	$x_3 y_3^2$	y_3^3	x_3^4	$x_3^3 y_3$	$x_3^2 y_3^2$	$x_3 y_3^3$	y_3^4	x_3^5	$x_3^3 y_3^2$	$x_3^2 y_3^3$	$x_3 y_3^4$	y_3^5
0	0	1	0	x_3	$2y_3$	0	x_3^2	$2x_3 y_3$	$3y_3^2$	0	x_3^3	$2x_3^2 y_3$	$3x_3 y_3^2$	$4y_3^3$	0	$2x_3^3 y_3$	$3x_3^2 y_3^2$	$4x_3 y_3^3$	$5y_3^4$
0	-1	0	$-2x_3$	$-y_3$	0	$-3x_3^2$	$-2x_3 y_3$	$-y_3^2$	0	$-4x_3^3$	$-3x_3^2 y_3$	$-2x_3 y_3^2$	$-y_3^3$	0	$-5x_3^4$	$-3x_3^2 y_3^2$	$-2x_3 y_3^3$	$-y_3^4$	0
1	x_4	y_4	x_4^2	$x_4 y_4$	y_4^2	x_4^3	$x_4^2 y_4$	$x_4 y_4^2$	y_4^3	x_4^4	$x_4^3 y_4$	$x_4^2 y_4^2$	$x_4 y_4^3$	y_4^4	x_4^5	$x_4^3 y_4^2$	$x_4^2 y_4^3$	$x_4 y_4^4$	y_4^5
0	0	1	0	x_4	$2y_4$	0	x_4^2	$2x_4 y_4$	$3y_4^2$	0	x_4^3	$2x_4^2 y_4$	$3x_4 y_4^2$	$4y_4^3$	0	$2x_4^3 y_4$	$3x_4^2 y_4^2$	$4x_4 y_4^3$	$5y_4^4$
0	1	0	$-2x_4$	$-y_4$	0	$-3x_4^2$	$-2x_4 y_4$	$-y_4^2$	0	$-4x_4^3$	$-3x_4^2 y_4$	$-2x_4 y_4^2$	$-y_4^3$	0	$-5x_4^4$	$-3x_4^2 y_4^2$	$-2x_4 y_4^3$	$-y_4^4$	0
1	x_5	y_5	x_5^2	$x_5 y_5$	y_5^2	x_5^3	$x_5^2 y_5$	$x_5 y_5^2$	y_5^3	x_5^4	$x_5^3 y_5$	$x_5^2 y_5^2$	$x_5 y_5^3$	y_5^4	x_5^5	$x_5^3 y_5^2$	$x_5^2 y_5^3$	$x_5 y_5^4$	y_5^5
0	0	1	0	x_5	$2y_5$	0	x_5^2	$2x_5 y_5$	$3y_5^2$	0	x_5^3	$2x_5^2 y_5$	$3x_5 y_5^2$	$4y_5^3$	0	$2x_5^3 y_5$	$3x_5^2 y_5^2$	$4x_5 y_5^3$	$5y_5^4$
0	-1	0	$-2x_5$	$-y_5$	0	$-3x_5^2$	$-2x_5 y_5$	$-y_5^2$	0	$-4x_5^3$	$-3x_5^2 y_5$	$-2x_5 y_5^2$	$-y_5^3$	0	$-5x_5^4$	$-3x_5^2 y_5^2$	$-2x_5 y_5^3$	$-y_5^4$	0
1	x_6	y_6	x_6^2	$x_6 y_6$	y_6^2	x_6^3	$x_6^2 y_6$	$x_6 y_6^2$	y_6^3	x_6^4	$x_6^3 y_6$	$x_6^2 y_6^2$	$x_6 y_6^3$	y_6^4	x_6^5	$x_6^3 y_6^2$	$x_6^2 y_6^3$	$x_6 y_6^4$	y_6^5
0	0	1	0	x_6	$2y_6$	0	x_6^2	$2x_6 y_6$	$3y_6^2$	0	x_6^3	$2x_6^2 y_6$	$3x_6 y_6^2$	$4y_6^3$	0	$2x_6^3 y_6$	$3x_6^2 y_6^2$	$4x_6 y_6^3$	$5y_6^4$
0	-1	0	$-2x_6$	$-y_6$	0	$-3x_6^2$	$-2x_6 y_6$	$-y_6^2$	0	$-4x_6^3$	$-3x_6^2 y_6$	$-2x_6 y_6^2$	$-y_6^3$	0	$-5x_6^4$	$-3x_6^2 y_6^2$	$-2x_6 y_6^3$	$-y_6^4$	0

THE TRPLT ELEMENT

STRUCTURAL ELEMENT DESCRIPTIONS

5. The matrix $[B_2]$ relating curvatures (for plates infinitely rigid in transverse shear) to the generalized coordinates in the equation $\{x_1\} = [B_2] \{a\}$ is given by: (24)

$$[B_2] = \begin{bmatrix} 0 & 0 & 0 & 2 & 0 & 0 & 6x & 2y & 0 & 0 & 12x^2 & bxy & 2y^2 & 0 & 0 & 20x^3 & 6xy^2 & 2y^3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 2x & by & 0 & 0 & 2x^2 & bxy & 12y^2 & 0 & 2x^3 & 6x^2y & 12xy^2 & 20y^3 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 4x & 4y & 0 & 0 & 6x^2 & 8xy & 6y^2 & 0 & 0 & 12x^2y & 12xy^2 & 8y^3 & 0 \end{bmatrix}$$

6. The matrix $[B_1]$ relating transverse shear strains $\{y\}$ to the generalized coordinates (in the equation $\{y\} = [B_1] \{a\}$) is a 2×20 matrix whose nonzero elements are as follows:

$$B_1(1,7) = 6A_{11} \quad (25a)$$

$$B_1(1,8) = 2A_{31} \quad (25b)$$

$$B_1(1,9) = 2A_{32} \quad (25c)$$

$$B_1(1,10) = 6A_{15} \quad (25d)$$

$$B_1(1,11) = 24A_{11}x \quad (25e)$$

$$B_1(1,12) = 6(A_{31}x + A_{11}y) \quad (25f)$$

$$B_1(1,13) = 4(A_{32}x + A_{31}y) \quad (25g)$$

$$B_1(1,14) = 6(A_{15}x + A_{32}y) \quad (25h)$$

$$B_1(1,15) = 24A_{15}y \quad (25i)$$

$$B_1(1,16) = -120(A_{11}^2 + A_{13}A_{21} - 0.5A_{11}x^2) \quad (25j)$$

$$B_1(1,17) = -12(A_{11}A_{32} + A_{13}A_{34} + A_{38}A_{31} + A_{39}A_{33} + A_{11}A_{16} + A_{15}A_{21} - 0.5A_{32}x^2 - A_{31}xy - 0.5A_{11}y^2) \quad (25k)$$

$$B_1(1,18) = -12(A_{11}A_{15} + A_{13}A_{25} + A_{38}A_{32} + A_{39}A_{34} + A_{16}A_{31} + A_{15}A_{33} - 0.5A_{15}x^2 - A_{32}xy - 0.5A_{31}y^2) \quad (25l)$$

$$B_1(1,19) = -24(A_{15}A_{38} + A_{25}A_{39} + A_{16}A_{32} + A_{15}A_{34} - A_{15}xy - 0.5A_{32}y^2) \quad (25m)$$

$$B_1(1,20) = -120(A_{15}A_{16} + A_{15}A_{25} - 0.5A_{15}y^2) \quad (25n)$$

$$B_1(2,7) = 6A_{21} \quad (25o)$$

$$B_1(2,8) = 2A_{33} \quad (25p)$$

THE TRPLT1 ELEMENT

$$B_1(2,9) = 2A_{34} \quad (25q)$$

$$B_1(2,10) = 6A_{25} \quad (25r)$$

$$B_1(2,11) = 24A_{21}x \quad (25s)$$

$$B_1(2,12) = 6(A_{33}x + A_{21}y) \quad (25t)$$

$$B_1(2,13) = 4(A_{34}x + A_{33}y) \quad (25u)$$

$$B_1(2,14) = 6(A_{25}x + A_{34}y) \quad (25v)$$

$$B_1(2,15) = 24A_{25}y \quad (25w)$$

$$B_1(2,16) = -120(A_{11}A_{21} + A_{23}A_{21} - 0.5A_{21}x^2) \quad (25x)$$

$$B_1(2,17) = -12(A_{21}A_{32} + A_{23}A_{34} + A_{40}A_{31} + A_{41}A_{33} + A_{26}A_{11} + A_{25}A_{21} - 0.5A_{34}x^2 - A_{33}xy - 0.5A_{21}y^2) \quad (25y)$$

$$B_1(2,18) = -12(A_{21}A_{15} + A_{23}A_{25} + A_{40}A_{32} + A_{41}A_{34} + A_{26}A_{31} + A_{25}A_{33} - 0.5A_{25}x^2 - A_{34}xy - 0.5A_{33}y^2) \quad (25z)$$

$$B_1(2,19) = -24(A_{15}A_{40} + A_{25}A_{41} + A_{26}A_{32} + A_{25}A_{34} - A_{25}xy - 0.5A_{34}y^2) \quad (25aa)$$

$$B_1(2,20) = -120(A_{15}A_{26} + A_{25}^2 - 0.5A_{25}y^2) \quad (25bb)$$

where;

$$\left. \begin{aligned} A_{11} &= -(J_{11}D_{11} + J_{12}D_{13}) \\ A_{12} &= -(J_{11}D_{12} + J_{12}D_{23}) \\ A_{13} &= -(J_{11}D_{13} + J_{12}D_{33}) \\ A_{14} &= -(J_{11}D_{13} + J_{12}D_{12}) \\ A_{15} &= -(J_{11}D_{23} + J_{12}D_{22}) \\ A_{16} &= -(J_{11}D_{33} + J_{12}D_{23}) \\ A_{21} &= -(J_{12}D_{11} + J_{22}D_{13}) \\ A_{22} &= -(J_{12}D_{13} + J_{22}D_{23}) \\ A_{23} &= -(J_{12}D_{13} + J_{22}D_{33}) \\ A_{24} &= -(J_{12}D_{13} + J_{22}D_{12}) \end{aligned} \right\} \quad (25cc)$$

STRUCTURAL ELEMENT DESCRIPTIONS

$$A_{25} = -(J_{12}^D D_{23} + J_{22}^D D_{22})$$

$$A_{26} = -(J_{12}^D D_{33} + J_{22}^D D_{23})$$

$$A_{31} = A_{14} + 2A_{13}$$

$$A_{32} = A_{12} + 2A_{16}$$

$$A_{33} = A_{24} + 2A_{23}$$

$$A_{34} = A_{22} + 2A_{26}$$

$$A_{35} = A_{33} + A_{11}$$

$$A_{36} = A_{34} + A_{31}$$

$$A_{37} = A_{25} + A_{32}$$

$$A_{38} = A_{13} + A_{14}$$

$$A_{39} = A_{12} + A_{16}$$

$$A_{40} = A_{23} + A_{24}$$

$$A_{41} = A_{22} + A_{26}$$

(25cc)
Con't.

7. The matrix $[B_3]$ relating $\{x_2\}$, the contribution of transverse shear to the vector of curvatures, to the generalized coordinates (in the equation $\{x_2\} = [B_3] \{a\}$) is given by;

$$B_3(1,11) = -24A_{11} \quad (26a)$$

$$B_3(1,12) = -6A_{31} \quad (26b)$$

$$B_3(1,13) = -4A_{32} \quad (26c)$$

$$B_3(1,14) = -6A_{15} \quad (26d)$$

$$B_3(1,16) = -120A_{11}x \quad (26e)$$

$$B_3(1,17) = -12(A_{32}x + A_{31}y) \quad (26f)$$

$$B_3(1,18) = -12(A_{15}x + A_{32}y) \quad (26g)$$

$$B_3(1,19) = -24A_{15}y \quad (26h)$$

$$B_3(2,12) = -6A_{21} \quad (26i)$$

$$B_3(2,13) = -4A_{33} \quad (26j)$$

THE TRPLT1 ELEMENT

$$\begin{aligned}
 B_3(2,14) &= -6A_{34} & (26k) \\
 B_3(2,15) &= -24A_{25} & (26l) \\
 B_3(2,17) &= -12(A_{33}x + A_{21}y) & (26m) \\
 B_3(2,18) &= -12(A_{34}x + A_{33}y) & (26n) \\
 B_3(2,19) &= -24(A_{25}x + A_{34}y) & (26o) \\
 B_3(2,20) &= -120A_{25}y & (26p) \\
 B_3(3,11) &= -24A_{21} & (26q) \\
 B_3(3,12) &= -6(A_{11} + A_{33}) & (26r) \\
 B_3(3,13) &= -4(A_{31} + A_{34}) & (26s) \\
 B_3(3,14) &= -6(A_{32} + A_{25}) & (26t) \\
 B_3(3,15) &= -24A_{15} & (26u) \\
 B_3(3,16) &= -120A_{21}x & (26v) \\
 B_3(3,17) &= -12 [(A_{34} + A_{31})x + (A_{33} + A_{11})y] & (26w) \\
 B_3(3,18) &= -12 [(A_{25} + A_{32})x + (A_{34} + A_{31})y] & (26x) \\
 B_3(3,19) &= -24 [A_{15}x + (A_{32} + A_{25})y] & (26y) \\
 B_3(3,20) &= -120A_{15}y & (26z)
 \end{aligned}$$

where;

A_{11} , A_{12} , . . . and A_{34} are as given in equations (25cc).

8. For plates with transverse shear flexibility the modified $[H]$ matrix, $[H']$, is given by subtracting the matrix $[B_1]$ for each of the six grid points from the respective rows of α and β of the grid points in the $[H]$ matrix.

9. For plates infinitely rigid in transverse shear,

$$[H]^{-1} = [H] \quad (27)$$

10. The two constraint equations involving the coefficients a_{16} , a_{17} , a_{18} , a_{19} and a_{20} of the quintic polynomial for transverse displacement to insure cubic edge rotation on the sloping edges of the triangular element are now entered as the 19th and 20th rows of $[H']$, i.e., the 19th and 20th rows are:

STRUCTURAL ELEMENT DESCRIPTIONS

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5b^4c & (3b^2c^3-2b^4c) & (2bc^4-3b^3c^2) & (c^5-4b^2c^3) & -5bc^4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5a^4c & (3a^2c^3-2a^4c) & (-2ac^4+3a^3c^2) & (c^5-4a^2c^3) & 5ac^4 \end{bmatrix}$$

This is now added as the 19th and 20th row of the $[H']$ matrix to form $[H'']$ matrix. $[H'']$ is a 20 x 20 square matrix that is nonsingular.¹

11. The $[H'']$ matrix is inverted. The first 18 columns of the $[H'']^{-1}$ matrix is denoted by the matrix $[\bar{H}]$ (Size 20 x 18), i.e., .

$$[\bar{H}] = \text{The first 18 columns of } [H'']^{-1} \quad (28)$$

8.25.3 Stiffness Matrix Calculation for TRPLT1 (Subroutines KTRPLS and KTRPLD)

1. The polynomial expressions, for variation of w and t within the element, are;

$$w = \sum_{i=1}^{20} a_i x^{m_i} y^{n_i} \quad (29)$$

$$t = \sum_{i=1}^3 c_i x^{r_i} y^{s_i} \quad (30)$$

The values of m_i , n_i , r_i and s_i are;

$$\begin{aligned} m_1 &= 0, m_2 = 1, m_3 = 0, m_4 = 2, m_5 = 1, \\ m_6 &= 0, m_7 = 3, m_8 = 2, m_9 = 1, m_{10} = 0, \\ m_{11} &= 4, m_{12} = 3, m_{13} = 2, m_{14} = 1, m_{15} = 0, \\ m_{16} &= 5, m_{17} = 3, m_{18} = 2, m_{19} = 1, m_{20} = 0, \\ n_1 &= 0, n_2 = 0, n_3 = 1, n_4 = 0, n_5 = 1, \\ n_6 &= 2, n_7 = 0, n_8 = 1, n_9 = 2, n_{10} = 3, \\ n_{11} &= 0, n_{12} = 1, n_{13} = 2, n_{14} = 3, n_{15} = 4, \end{aligned} \quad (31)$$

$$n_{16} = 0, n_{17} = 2, n_{18} = 3, n_{19} = 4, n_{20} = 5, \quad (32)$$

(1) A numerical experiment to verify that $[H'']$ is nonsingular for all practical element sizes is described in "New Triangular and Quadrilateral Plate-bending Finite Elements", by R. Narayana-swami, NASA TN D-7407, April, 1974.

THE TRPLT1 ELEMENT

$$r_1 = 0, p_2 = 1, p_3 = 0 \quad (33)$$

$$s_1 = 0, q_2 = 0, q_3 = 1 \quad (34)$$

The coefficients a_1 to a_{20} are generalized coordinates of the element and can be evaluated once the displacement vector is known.

The coefficients c_1 , c_2 , and c_3 can be evaluated from the specified thicknesses t_1 , t_3 and t_5 of the 3 corner grid points and the geometric dimensions of the element:

$$c_1 = \frac{t_1 a + t_3 b}{(a + b)} \quad (35)$$

$$c_2 = \frac{t_3 - t_1}{(a + b)} \quad (36)$$

$$c_3 = \frac{1}{c} (t_5 - c_1) \quad (37)$$

where t_1 , t_3 and t_5 are evaluated from the values I_1 , I_3 and I_5 respectively.

The elements of the symmetric portion of the stress-strain matrix $[G_e]$ are denoted by G_{11} , G_{12} , G_{13} , G_{22} , G_{23} and G_{33} .

A formula for the integral of the type $x^m y^n$ taken over the area of the element is;

$$\iint x^m y^n dx dy = F(m, n) = c^{n+1} \{a^{m+1} - (-b)^{m+1}\} - \frac{m!n!}{(m+n+2)!} \quad (38)$$

The equation used is the stiffness matrix generation in generalized coordinates for plates infinitely rigid in transverse shear is given by:

$$[k_{ij}]_{\text{gen}} = \frac{1}{12} \sum_{k_1=1}^3 \sum_{k_2=1}^3 \sum_{k_3=1}^3 c_{k_1} c_{k_2} c_{k_3} \\ [G_{11} m_i m_j (m_i - 1) (m_j - 1) F(m_i + m_j + r_{k_1} + r_{k_2} + r_{k_3} - 4 \\ n_i + n_j + s_{k_1} + s_{k_2} + s_{k_3}) + G_{22} n_i n_j (n_i - 1) (n_j - 1) F(m_i + m_j \\ + r_{k_1} + r_{k_2} + r_{k_3}, n_i + n_j + s_{k_1} + s_{k_2} + s_{k_3} - 4) + \quad (39)$$

STRUCTURAL ELEMENT DESCRIPTIONS

$$\begin{aligned}
 & + (4G_{33}m_i m_j n_i n_j + G_{12}\{m_i m_j (m_i - 1)(n_j - 1) + m_i n_i (m_j - 1) \\
 & \cdot (n_i - 1)\} F(m_i + m_j + r_{k_1} + r_{k_2} + r_{k_3} - 2, n_i + n_j + s_{k_1} \\
 & + s_{k_2} + s_{k_3} - 2) + 2G_{13}\{m_i m_j n_j (m_i - 1) + m_i n_i m_j (m_j - 1)\} F(m_i \\
 & + m_j + r_{k_1} + r_{k_2} + r_{k_3} - 3, n_i + n_j + s_{k_1} + s_{k_2} + s_{k_3} - 1) \\
 & + 2G_{23}\{m_j n_i n_j (n_i - 1) + m_i n_i n_j (n_j - 1)\} F(m_i + m_j + r_{k_1} + r_{k_2} \\
 & + r_{k_3} - 1, n_i + n_j + s_{k_1} + s_{k_2} + s_{k_3} - 3)]
 \end{aligned}
 \tag{39}$$

Cont.

For plates with transverse shear flexibility, the expression for generalized stiffness matrix consists not only of the closed form expression but four additional integrals, given below, that are evaluated using numerical integration, i.e.,

$$\begin{aligned}
 [K_{gen}] &= [K_{gen}]_{\text{closed form}} + \iint [B_2]^T [D] [B_3] dx dy \\
 &+ \iint [B_3]^T [D] [B_2] dx dy + \iint [B_3]^T [D] [B_3] dx dy \\
 &+ \iint [B_1]^T [G_s] [B_1] dx dy
 \end{aligned}
 \tag{40}$$

[D] matrix is obtained from the stress-strain matrix $[G_e]$ as

$$[D] = \frac{1}{12} [G_e] \left(\sum_{i=1}^3 \sum_{j=1}^3 \sum_{k=1}^3 c_i c_j c_k x^{r_i+r_j+r_k} y^{s_i+s_j+s_k} \right)
 \tag{41}$$

$$[G_s] = G t^* \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}
 \tag{42}$$

where t^* is the effective thickness for shear at the integration point and is evaluated from the user specified values TS1, TS3 and TS5. The numerical integration formulae used are the seven-point integration scheme listed in Zienkiewics¹ and are given below.

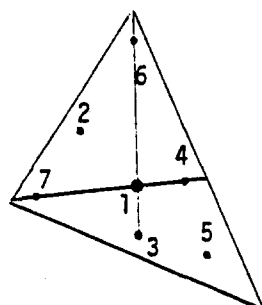
For a triangle, the integrals of the form;

- (1) O. C. Zienkiewicz, Finite Element Method on Engineering Science, New York, London, McGraw-Hill, 1971.

THE TRPLT1 ELEMENT

$$I = \int_0^1 \int_0^{1-L} f(L_1 L_2 L_3) dL_1 dL_2 = \sum_{k=1}^7 w_k f_k(L_1 L_2 L_3) \quad (43)$$

where the points (L_1 , L_2 and L_3) and the weighting factors are as follows:



Point	Triangular Coordinates L_1, L_2, L_3	Weight, $2W_k$
1	1/3, 1/3, 1/3	0.225
2	$\alpha_1 \quad \beta_1 \quad \beta_1$	0.13239415
3	$\beta_1 \quad \alpha_1 \quad \beta_1$	
4	$\beta_1 \quad \beta_1 \quad \alpha_1$	
5	$\alpha_2 \quad \beta_2 \quad \beta_2$	0.12593928
6	$\beta_2 \quad \alpha_2 \quad \beta_2$	
7	$\beta_2 \quad \beta_2 \quad \alpha_2$	

with;

$$\alpha_1 = 0.05971588 \quad \beta_1 = 0.47014206$$

$$\alpha_2 = 0.79742699 \quad \beta_2 = 0.101286505$$

The stiffness matrix in global coordinates is;

$$[k] = [E] [T]^T [\bar{H}]^T [k]_{\text{gen}} [\bar{H}] [T] [E]^T \quad (44)$$

8.25.4 Mass Matrix Calculation for TRPLT1 (Calculated in Stiffness Subroutines KTRPLS and KTRPLD)

- Two different mass matrices are used; the lumped mass and the consistent mass. The lumped mass matrix is calculated in the same manner as for TRIM6:

$$m = \frac{1}{6} (pV + A\mu) \quad (45)$$

STRUCTURAL ELEMENT DESCRIPTIONS

where

$$V, \text{ the volume of the element} = c_1 F(0,0) + c_2 F(1,0) + c_3 F(0,1) \quad (46)$$

For each point, the diagonal mass matrix in element coordinates at all the grid points is;

$$[m_i] = \begin{bmatrix} 0 & & & & & 0 \\ & 0 & & & & \\ & & m & & & \\ & & & 0 & & \\ & & & & 0 & \\ 0 & & & & & 0 \end{bmatrix} \quad i = 1, 2, \dots \text{ and } 6 \quad (47)$$

so that $[M_e]$ the element mass matrix has $[m_i]$ matrices arranged diagonally.

The mass matrix in the global coordinate system is obtained as;

$$[M_{gg}] = [E] [T]^T [M_{ee}] [T] [E]^T \quad (48)$$

If the parameter COUPMASS is set by the user, the consistent mass matrix will be formed. The j th element of the i th row of generalized mass matrix is given by:

$$[M_{ij}]_{\text{gen}} = \rho \iint \sum_{k=1}^3 c_k x^{m_i+m_j+r_k} y^{n_i+n_j+s_k} dx dy \quad (49)$$

$$+ \mu \iint x^{m_i+m_j} y^{n_i+n_j} dx dy$$

$$= \rho \sum_{k=1}^3 \{c_k F(m_i + m_j + r_k, n_i + n_j + s_k)\} \quad (50)$$

$$+ u F(m_i + m_j, n_i + n_j)$$

The mass matrix in global coordinates is;

$$[M] = [E] [T]^T [\bar{H}]^T [M_{\text{gen}}] [\bar{H}] [T] [E]^T \quad (51)$$

8.25.5 Structural Damping Matrices for the TRPLT1 Element

The structural damping matrices are;

$$[K_{ij}^4] = g_e [k_{ij}^g] \quad (52)$$

where g_e is the structural damping coefficient for the bending material referenced.

8.25.6 Stress and Element Force Calculations for the TRPLT1 Element (Subroutines STRP11 and STRP12 of Modules SDR2)

1. STRP11 is used to calculate the phase 1 stress-displacement relations.

Frequent reference will be made to the equations from sections 8.25.2 and 8.25.3.

The following data are calculated:

1. $[\bar{H}]$ - 20 x 18 Matrix relating generalized coordinates to grid point displacements.
2. $[B_2]$ - 3 x 20 Matrix relating bending curvatures and generalized coordinates.
3. $[B_3]$ - 2 x 20 Matrix relating curvature contribution of transverse shear strains and generalized coordinates.
4. $[E]$ - Element to basic coordinate transformation.
5. $[D]$ - 3 x 3 Matrix of elastic coefficients relating bending moments and curvatures.
6. $[G_s]$ - 2 x 2 Matrix relating transverse shear forces and shear strains.
7. $[T_i]$ - $i = 1, 2, \dots, 6$ - Global to basic transformations.

The following calculations are performed.

$$[S_M^*] = [D] [B_2] [\bar{H}] \quad (53)$$

$[S_M^*]$ is a 3 x 18 matrix; this is split into six 3 x 3 matrix partitions as follows:

$$[S_M^*] = \left[\begin{array}{c|c|c|c|c|c} S_{M_1}^* & S_{M_2}^* & & & & \\ \hline & & & & & \\ \hline & & & & & \\ \hline & & & & & \\ \hline & & & & & \\ \hline & & & & & S_{M_6}^* \end{array} \right] \quad (54)$$

Each of the six matrix partitions is multiplied as follows:

$$\left[S_{M_i} \right] = \left[S_{M_i}^* \right] [E]^T [T_i] \quad i = 1, 2, \dots, 6 \quad (55)$$

$$[S_G^*] = [G_s] [B_3] [\bar{H}] \quad (56)$$

$[S_G^*]$ is a 2 x 18 matrix; this is split into six 2 x 3 matrix partitions as follows:

$$[S_G^*] = \left[\begin{array}{c|c|c|c|c|c} S_{G_1}^* & S_{G_2}^* & & & & \\ \hline & & & & & \\ \hline & & & & & \\ \hline & & & & & \\ \hline & & & & & \\ \hline & & & & & S_{G_6}^* \end{array} \right] \quad (57)$$

Each of the six matrix partitions is multiplied as follows:

$$\left[S_{G_i} \right] = \left[S_{G_i}^* \right] [E]^T [T_i] \quad i = 1, 2, \dots, 6 \quad (58)$$

The 5 x 6 matrix $[S_i]$ is obtained as;

STRUCTURAL ELEMENT DESCRIPTIONS

$$[S_i] = \begin{bmatrix} S_{M_i} \\ S_{G_i} \end{bmatrix} \quad i = 1, 2, \dots, 6 \quad (59)$$

2. Phase 2

(a) The vector of forces is computed as

$$\begin{Bmatrix} M_x \\ M_y \\ M_{xy} \\ V_x \\ V_y \end{Bmatrix} = \left(\sum_{i=1}^6 [S_i] \{u_i\} \right) - \{M_t\} \quad (60)$$

where $\{M_t\}$ is the thermal moment vector. If the thermal gradient is specified,

$$\text{where } \{M_t\}_i = -[G_e] \{\alpha_e\} I_i T'_i \quad (61)$$

where I_i is the moment of inertia of the cross section and T'_i is the thermal gradient at vertex i of the element.

The stresses and forces are evaluated at the vertices of the element; in addition, the stresses are also evaluated at the centroid. The simplification is made that the thermal moment vector at the centroid is the average of that at the vertices.

(b) With no given temperatures at the stress points, the stresses are then calculated from the equations

$$\begin{aligned} \sigma_x &= \frac{-V_e M_{xz}}{I} \\ \sigma_y &= \frac{-V_e M_{yz}}{I} \\ \sigma_{xy} &= \frac{-V_e M_{xyz}}{I} \end{aligned} \quad (62)$$

M_x , M_y , M_{xy} and I relate to the appropriate points (vertices or centroid); z values for the corner grid points are as those given in the PTRPLT1 card and for the centroid, the z values are the top and bottom fibre distances; and V_e is element volume.

THE TRPLT1 ELEMENT

If mean temperature T_0 and the gradient T' are specified for the element, and the user specified temperature T at the point where outer fiber stresses are calculated is different from $(T_0 + T'Z_i)$, then

$$\begin{Bmatrix} \sigma_{x_i} \\ \sigma_{y_i} \\ \sigma_{xy_i} \end{Bmatrix} = -\frac{Z_i}{I} \begin{Bmatrix} M_x \\ M_y \\ M_{xy} \end{Bmatrix} + \frac{[D]\{\alpha\}}{I} [T'Z_i + (T_0 - T)], \quad i = 1, 2 \quad (63)$$

The principal stresses and angles are calculated using the same formula as for the membrane element TRIM6 (section 8.24.6).

8.25.7 Thermal Load Calculations for the Bending Elements (Subroutine TLODT1, TLODT2 and TLODT3 of Module SSG1)

The variation over the surface of the element of the mean temperature, T_0 , and the thermal gradient at a cross section, T' , is assumed as a bilinear ploynomial

$$T_0 = \sum_{i=1}^3 d_i x^p y^q \quad (64)$$

$$T' = \sum_{i=1}^3 d_i^1 x^p y^q \quad (65)$$

so that the temperature at any point (x, y, z) is $T = T_0 + T'$.

The constants d_i and d_i^1 are evaluated from the values at the vertices and the reference temperature T_0 of the element:

$$d_1 = \frac{T_{01}a + T_{03}b}{(a+b)} - T_0; \quad d_2 = \frac{T_{03} - T_{01}}{(a+b)}; \quad d_3 = \frac{1}{c} [T_{05} - d_1] \quad (66)$$

$$d_1^1 = \frac{T_1^1 a + T_3^1 b}{(a+b)}; \quad d_2^1 = \frac{T_3^1 - T_1^1}{(a+b)}; \quad d_3^1 = \frac{1}{c} [T_5^1 - d_1^1] \quad (67)$$

It is convenient to define the elements of $[G_e]\{\alpha_e\}$ as

$$G_{11}' = G_{11}\alpha_{e1} + G_{12}\alpha_{e2} + G_{13}\alpha_{e3} \quad (68)$$

$$G_{22}' = G_{12}\alpha_{e1} + G_{22}\alpha_{e2} + G_{23}\alpha_{e3} \quad (69)$$

STRUCTURAL ELEMENT DESCRIPTIONS

$$G'_{33} = G_{13}\alpha_{e_1} + G_{23}\alpha_{e_2} + G_{33}\alpha_{e_3} \quad (70)$$

The thermal load vector in generalized coordinates, $\{P_{gen}^t\}$, will be evaluated in two stages, vis., the closed form expression $\{P_{gen}^t\}_1$, due to the vector of curvatures in the absence of transverse shear and the numerically integrated expression $\{P_{gen}^t\}_2$ due to the contribution of transverse shear to the vector of curvatures. The i th element of $\{P_{gen}^t\}_1$ is given by:

$$\begin{aligned} [\{P_{gen}^t\}_1]_i = & \frac{1}{12} \sum_{i_1=1}^3 \sum_{i_2=1}^3 \sum_{i_3=1}^3 \sum_{i_4=1}^3 c_{i_1} c_{i_2} c_{i_3} d_j' [G'_{11} m_i (m_i - 1) F(m_i + r_i \\ & + r_{i_2} + r_{i_3} + p_j - 2, n_i + s_{i_1} + s_{i_2} + s_{i_3} + q_j) + G'_{22} n_i (n_i - 1) F(m_i \\ & + r_{i_1} + r_{i_2} + r_{i_3} + p_j, n_i + s_{i_1} + s_{i_2} + s_{i_3} + q_j - 2) \\ & + G'_{33} m_i n_i F(m_i + r_{i_1} + r_{i_2} + r_{i_3} + p_j - 1, n_i + s_{i_1} + s_{i_2} + s_{i_3} + q_j - 1)] \end{aligned} \quad (71)$$

The load vector $\{P_{gen}^t\}_2$ is evaluated using numerical integration of the following expression:

$$\{P_{gen}^t\}_2 = \frac{1}{12} \iint [B_3]^T [G_e] \{\alpha_e\} T' t^3 dx dy \quad (72)$$

The generalized thermal load vector is

$$\{P_{gen}^t\} = \{P_{gen}^t\}_1 + \{P_{gen}^t\}_2 \quad (73)$$

The thermal load vector in global coordinates is

$$\{P_g^t\} = [E] [T]^T [\bar{A}]^T \{P_{gen}^t\} \quad (74)$$

The forces are placed in the PG load vector data block.

THE TRSHL ELEMENT

8.26 TRSHL: SHALLOW SHELL TRIANGULAR ELEMENT

8.26.1 Input Data for TRSHL Element

1. EST entries for TRSHL are

<u>Symbol</u>	<u>Description</u>
EID	Element Identification Number
SIL1, . . . SIL6	Scalar indices of connected grid points
θ	Anisotropic material orientation angle
Mat ID _m	Material-Identification Number for membrane behavior
T ₁ , T ₃ , T ₅	Membrane thickness at corner grid points
Mat ID _b	Material-Identification Number for bending
I ₁ , I ₃ , I ₅	Area moments of inertia at corner grid points
Mat ID _s	Material-Identification Number for transverse shear
TS1, TS3, TS5	Thickness for transverse shear at corner grid points
μ	Nonstructural mass per unit area
Z11, Z21, Z13, Z23, Z15, Z25	Distances Z1 and Z2 for stress calculations at three corner grid points
$\left. \begin{array}{l} N_i \\ X_i \\ Y_i \\ Z_i \end{array} \right\} \quad i = 1, \dots, 6$	Local coordinate system numbers and location of coordinates in the basic system for the connected grid points
TEMP	Element temperature

2. Coordinate system data

The numbers N_i , X_i , and Z_i are used to calculate the 3 by 3 basic-to-global coordinate transformation matrices T_i for points $i = 1, 2, 3, 4, 5$ and 6 (via subroutines TRANSD or TRANSS).

STRUCTURAL ELEMENT DESCRIPTIONS

3. Material data

For mat. ID _m	{	[G]	3 x 3 stress-strain matrix
		ρ	Mass density
		$\alpha_x, \alpha_y, \alpha_{xy}$	Thermal expansion coefficients
		T0	Reference temperature
		g_e	Structural damping coefficient
		$\sigma_t, \sigma_c, \sigma_s$	Stress limits for tension, compression and shear
For ID _b	{	D	3 x 3 bending stress-strain matrix
For ID _s	{	G_s	Shear coefficient

8.26.2 Basic Equation for TRSHL

The calculations for the TRSHL element are very similar to those of TRIM6 and TRPLT1 (sections 8.24.2 and 8.25.2 respectively) and therefore, only the essential details are given here.

The displacement transformation matrix from basic coordinates to in-plane coordinates is

$$[E]^T = \begin{bmatrix} i_1 & i_2 & i_3 & 0 & 0 & 0 \\ j_1 & j_2 & j_3 & 0 & 0 & 0 \\ k_1 & k_2 & k_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & i_1 & i_2 & i_3 \\ 0 & 0 & 0 & j_1 & j_2 & j_3 \end{bmatrix} \quad (1)$$

where i , j and k are as given by equations (4), (5) and (6) of section 8.25.2. No transverse shear effects are considered for the TRSHL element.

The matrix $[H]$ relating grid point displacements and the generalized coordinates (in the equation $\begin{Bmatrix} u \\ v \\ w \end{Bmatrix} = [H] \{a\}$) is similar to that for TRIM6 and TRPLT1. The inverse matrix relating generalized coordinates to the grid point displacement vector is a 32 x 30 matrix and is given by

THE TRSHL ELEMENT

$$[H]^{-1} = \begin{bmatrix} H_1^{-1} & 0 \\ 6 \times 6 & \\ \hline & H_1^{-1} \\ & 6 \times 6 \\ \hline & & H'' \\ & & 20 \times 18 \end{bmatrix} \quad (2)$$

where H_1 is a 6 x 6 matrix given in equation (21), section 8.24.2 and H'' is a 20 x 18 matrix given in equation (28) of section 8.25.2.

STRUCTURAL ELEMENT DESCRIPTIONS

8.26.3 Stiffness Matrix Calculation for TRSHL (Subroutine KTSMLS and KTSMLD)

The polynomial expressions for variation of u , v , w and thickness t within the element are

$$u = \sum_{i=1}^{32} a_i x^{m_i} y^{n_i} \quad (3)$$

$$v = \sum_{i=1}^{32} b_i x^{p_i} y^{q_i} \quad (4)$$

$$w = \sum_{i=1}^{32} c_i x^{v_i} y^{s_i} \quad (5)$$

$$t_m = \sum_{i=1}^3 d_i x^{t_i} y^{u_i} \quad (6)$$

$$t_b = \sum_{i=1}^3 d'_i x^{t'_i} y^{u'_i} \quad (7)$$

The values of m_i , n_i , p_i , q_i , v_i , s_i , t_i , u_i , t'_i and u'_i are

$$m_i(i = 1,32): 0, 1, 0, 2, 1, 0, 26*0$$

$$n_i(i = 1,32): 0, 0, 1, 0, 1, 2, 26*0$$

$$p_i(i = 1,32): 6*0, 0, 1, 0, 2, 1, 0, 20*0$$

$$q_i(i = 1,32): 6*0, 0, 0, 1, 0, 1, 2, 20*0$$

(8)

$$v_i(i = 1,32): 12*0, 0, 1, 0, 2, 1, 0, 3, 2, 1, 0, 4,$$

$$3, 2, 1, 0, 5, 3, 2, 1, 0$$

$$s_i(i = 1,32): 12*0, 0, 0, 1, 0, 1, 2, 0, 1, 2, 3, 0,$$

$$1, 2, 3, 4, 0, 2, 3, 4, 5$$

$$t_i(i = 1,3): 0, 1, 0; t'_i: 0, 1, 0; u_i: 0, 0, 1; u'_i: 0, 0, 1$$

THE TRSHL ELEMENT

The coefficients a_i , b_i and c_i are undetermined parameters such that

$$\begin{aligned} a_i &= 0 & i &= 7 \text{ to } 32 \\ b_i &= 0 & i &= 1 \text{ to } 6 \text{ and } 13 \text{ to } 32 \\ c_i &= 0 & i &= 1 \text{ to } 12 \end{aligned} \quad (9)$$

a_i , $i = 1$ to 6 ; b_i , $i = 7$ to 12 ; and c_i , $i = 13$ to 32 can be determined once the element displacement vector is known.

The coefficients d_i and d'_i , $i = 1$ to 3 are evaluated from the user specified values of the membrane thickness and the area moments of inertia, respectively, by equations similar to those for TRIM6 and TRPLT1 elements.

The equation used in the stiffness matrix generation in generalized coordinates is (following the procedure outlined in sections 5.8.6 and 5.8.7 of the Theoretical Manual), the j th column of the i th row of the generalized stiffness matrix is obtained as

$$\begin{aligned} K_{ij} = & \sum_{k=1}^3 [G_{11} (m_i m_j d_k F(m_i + m_j + t_k - 2, n_i + n_j + u_k) \\ & - h_4' m_i d_k F(m_i + r_j + t_k - 1, n_i + s_j + u_k) \\ & - h_4 m_j d_k F(m_j + r_i + t_k - 1, n_j + s_i + u_k) \\ & + h_4^2 d_k F(r_i + r_j + t_k, s_i + s_j + u_k)) \\ & + G_{22} (q_i q_j d_k F(p_i + p_j + t_k, q_i + q_j + u_k - 2) \\ & - h_6 q_i d_k F(p_i + r_j + t_k, q_i + s_j + u_k - 1) \\ & - h_6 q_j d_k F(r_i + p_j + t_k, s_i + q_j + u_k - 1) \\ & + h_6^2 d_k F(r_i + r_j + t_k, s_i + s_j + u_k)) \\ & + G_{33} (n_i n_j d_k F(m_i + m_j + t_k, n_i + n_j + u_k - 2) \end{aligned} \quad (10)$$

STRUCTURAL ELEMENT DESCRIPTIONS

$$\begin{aligned}
 & + n_i p_j d_k F(m_i + p_j + t_k - 1, n_i + q_j + u_k - 1) \\
 & - h_5 n_i d_k F(m_i + r_j + t_k, n_i + s_j + u_k - 1) \\
 & + p_i n_j d_k F(p_i + j_m + t_k - 1, q_i + n_j + u_k - 1) \\
 & + p_i p_j d_k F(p_i + p_j + t_k - 2, q_i + q_j + u_k) \\
 & - h_5 p_i d_k F(p_i + r_j + t_k - 1, q_i + s_j + u_k) \\
 & - h_5 n_j d_k F(r_i + m_j + t_k, s_i + n_j + u_k - 1) \\
 & - h_5 p_j d_k F(r_i + p_j + t_k - 1, s_i + q_j + u_k) \\
 & + h_5^2 d_k F(r_i + r_j + t_k, s_i + s_j + u_k) \\
 & + G_{12} \left(m_i q_j d_k F(m_i p_j + t_k - 1, n_i + q_j + u_k - 1) \right. \\
 & - h_6 m_i d_k F(m_i + r_j + t_k - 1, n_i + s_j + u_k) \\
 & - h_4 q_i d_k F(r_i + p_j + t_k, s_i + q_j + u_k - 1) \\
 & + 2h_4 h_6 d_k F(r_i + r_j + t_k, s_i + s_j + u_k) \\
 & + q_i m_j d_k F(p_i + m_j + t_k - 1, q_i + n_j + u_k - 1) \\
 & - h_4 q_i d_k F(p_i + r_j + t_k, q_i + s_j + u_k - 1) \\
 & \left. - h_6 m_j d_k F(r_i + m_j + t_k - 1, s_i + n_j + u_k) \right) \\
 & + G_{13} \left(m_i n_j d_k F(m_i + m_j + t_k - 1, n_i + n_j + u_k - 1) \right. \\
 & + m_i p_j d_k F(m_i + p_j + t_k - 2, n_i + q_j + u_k) \\
 & \left. - h_5 m_i d_k F(m_i + r_j + t_k - 1, n_i + s_j + u_k) \right)
 \end{aligned}$$

(10)
Con't.

THE TRSHL ELEMENT

$$\begin{aligned}
 & - h_4 n_j d_k F(r_i + m_j + t_k, s_i + n_j + u_k - 1) \\
 & - h_4 p_j d_k F(r_i + p_j + t_k - 1, s_i + q_j + u_k) \\
 & + 2h_4 h_5 d_k F(r_i + r_j + t_k, s_i + s_j + u_k) \\
 & + n_i m_j d_k F(m_i + m_j + t_k - 1, n_i + n_j + u_k - 1) \\
 & - h_4 n_i d_k F(m_i + r_j + t_k, n_i + s_j + u_k - 1) \\
 & + p_i m_j d_k F(p_i + m_j + t_k - 2, q_i + n_j + u_k) \\
 & - h_4 p_i d_k F(p_i + r_j + t_k - 1, q_i + s_j + u_k) \\
 & - h_5 m_j d_k F(r_i + m_j + t_k - 1, s_i + n_j + u_k) \\
 & + G_{23} \left(q_i n_j d_k F(p_i + m_j + t_k, q_i + n_j + u_k - 2) \right. \\
 & + q_i p_j d_k F(p_i + p_j + t_k - 1, q_i + q_j + u_k - 1) \\
 & - h_5 q_i d_k F(p_i + r_j + t_k, q_i + s_j + u_k - 1) \\
 & - h_6 n_j d_k F(r_i + m_j + t_k, s_i + n_j + u_k - 1) \\
 & - h_6 p_j d_k F(r_i + p_j + t_k - 1, s_i + q_j + u_k) \\
 & + 2h_5 h_6 d_k F(r_i + r_j + t_k, s_i + s_j + u_k) \\
 & + n_i q_j d_k F(m_i + p_j + t_k, n_i + q_j + u_k - 2) \\
 & - h_6 n_i d_k F(m_i + r_j + t_k, n_i + s_j + u_k - 1) \\
 & + p_i q_j d_k F(p_i + p_j + t_k - 1, q_i + q_j + u_k - 1) \\
 & \left. - h_6 p_i d_k F(p_i + r_j + t_k - 1, q_i + s_j + u_k) \right)
 \end{aligned}$$

(10)
Con't.

STRUCTURAL ELEMENT DESCRIPTIONS

$$\begin{aligned}
 & - h_5 q_j d_k F(r_i + p_j + t_k, s_i + q_j + u_k - 1) \\
 & + \sum_{k_1=1}^3 \sum_{k_2=1}^3 \sum_{k_3=1}^3 \left[\frac{1}{12} d'_{k_1} d'_{k_2} d'_{k_3} (G_{11} r_i r_j (r_i - 1) (r_j - 1) \right. \\
 & \quad \cdot F(r_i + r_j + t'_{k_1} + t'_{k_2} + t'_{k_3} - 4, s_i + s_j + u'_{k_1} + u'_{k_2} + u'_{k_3}) \\
 & \quad + G_{22} s_i s_j (s_i - 1) (s_j - 1) F(r_i + r_j + t'_{k_1} + t'_{k_2} + t'_{k_3} \\
 & \quad \quad \quad \left. + s_i + s_j + u'_{k_1} + u'_{k_2} + u'_{k_3} - 4) \right. \\
 & \quad \left. + (4G_{33} r_i r_j s_i s_j + G_{12} \{r_i s_j (r_i - 1) (s_j - 1) \right. \\
 & \quad \left. + r_j s_i (r_j - 1) (s_i - 1)\}) F(r_i + r_j + t'_{k_1} + t'_{k_2} + t'_{k_3} - 2, \right. \\
 & \quad \quad \quad \left. + s_i + s_j + u'_{k_1} + u'_{k_2} + u'_{k_3} - 2) \right. \\
 & \quad \left. + 2G_{13} \{r_i r_j s_j (r_i - 1) + r_i r_j s_i (r_j - 1)\} F(r_i + r_j \right. \\
 & \quad \left. + t'_{k_1} + t'_{k_2} + t'_{k_3} - 3, s_i + s_j + u'_{k_1} + u'_{k_2} + u'_{k_3} - 1) \right. \\
 & \quad \left. + 2G_{23} \{r_j s_i s_j (s_i - 1) + r_i s_i s_j (s_j - 1)\} F(r_i + r_j \right. \\
 & \quad \left. + t'_{k_1} + t'_{k_2} + t'_{k_3} - 1, s_i + s_j + u'_{k_1} + u'_{k_2} + u'_{k_3} - 3) \right]
 \end{aligned}
 \tag{10}$$

Con't.

The generalized stiffness matrix can be transformed to the element and global coordinates by transformations similar to those for TRIM6 and TRPLT1 elements.

8.26.4 Mass Matrix Calculation for the TRSHL Element (Calculated in the stiffness subroutine KTSHLS and KTSHLD)

Two different mass matrices are calculated: lumped mass and consistent mass. The calculations are the same as for TRIM6 and TRPLT1.

8.26.5 Structural Damping Matrices for the TRSHL Element

The calculations are similar to those for TRIM6 and TRPLT1 elements.

THE TRSHL ELEMENT

8.26.6 Stress and Element Force Calculations for TRSHL Element (Subroutines STRSL1, STRSLV and STRSL2 of module SDR2)

The calculations are similar to those of TRIM6 and TRPLT1 elements.

8.26.7 Thermal Load Calculations for the TRSHL Element (Subroutines TLØDSL of module SSG1)

The calculations are similar to those for TRIM6 and TRPLT1 elements.

8.26.8 Differential Stiffness Matrix Calculations for the TRSHL Element (Subroutine TRSHL of module SSG1)

The steps leading to the calculations of the differential stiffness matrix are given in Section 7.6 of the Theoretical Manual.

8.27 THE RØD, CØNRØD AND TUBE ELEMENTS

8.27.1 Input Data for the RØD, TUBE, CØNRØD Elements

1. The ECPT/EST entries for the RØD and CØNRØD are:

<u>Symbol</u>	<u>Description</u>
SIL_a, SIL_b	Scalar indices for grid points a and b
N_a, X_a, Y_a, Z_a N_b, X_b, Y_b, Z_b	Local coordinate system number and basic coordinates of grid points
Mat I. D.	Material identification number
A	Cross-section area
J	Polar inertia
μ	Nonstructural mass per unit length
C	Shear stress coefficient
t_μ	Temperature for material properties

2. The TUBE element has the same characteristics as the RØD except for different input properties. The TUBE has d, the outside diameter, and t, the thickness, given.

The conversion to RØD properties is:

$$A = \pi(d - t)t$$

$$J = \frac{1}{4}A((d - t)^2 + t^2)$$

$$C = \frac{d}{2}$$

3. Coordinate system data

Given $N_a, X_a, Y_a, Z_a, N_b, X_b, Y_b, Z_b$ and the CSTM (Coordinate System Transformation Matrices) data block, the 3 by 3 global-to-basic coordinate transformation matrices $[T_a]$ and $[T_b]$ are calculated using the utility routine TRANSD or TRANSS.

4. Material data

Given the "MAT I.D." and t_μ , the material routine, MAT, returns the following data:

- E - Modulus of Elasticity
- G - Shear Modulus
- ν - Poisson's ratio
- ρ - Density
- α - Thermal expansion coefficient
- T_0 - Reference temperature
- q_e - Structural damping ratio
- σ_t - Stress limit, tension
- σ_c - Stress limit, compression
- σ_s - Stress limit, shear

8.27.2 Stiffness Matrix Calculation (Subroutines KRØD and KTUBE of Module SMA1)

1. Calculate the length of member, (l):

$$l = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2 + (z_a - z_b)^2} \quad (1)$$

2. Calculate a normalized direction vector $\{n\}$ in basic coordinates:

$$\begin{Bmatrix} n_1 \\ n_2 \\ n_3 \end{Bmatrix} = \frac{1}{l} \begin{Bmatrix} x_a - x_b \\ y_a - y_b \\ z_a - z_b \end{Bmatrix} \quad (2)$$

3. Form the extensional stiffness matrix, $[D_\ell]$:

$$[D_\ell] = \frac{AE}{l} \begin{bmatrix} n_1^2 & n_1 n_2 & n_1 n_3 \\ n_1 n_2 & n_2^2 & n_2 n_3 \\ n_1 n_3 & n_2 n_3 & n_3^2 \end{bmatrix} \quad (3)$$

4. Form the torsional stiffness matrix $[D_r]$:

$$[D_r] = \frac{GJ}{\ell} \begin{bmatrix} n_1^2 & n_1 n_2 & n_1 n_3 \\ n_1 n_2 & n_2^2 & n_2 n_3 \\ n_1 n_3 & n_2 n_3 & n_3^2 \end{bmatrix} . \quad (4)$$

5. $[T_a]$ and $[T_b]$ are the matrices which transform displacement components in the global coordinate system to the basic system.

6. Transforming to global coordinates and combining the results give the partitions of the element stiffness matrix:

$$[k_{aa}] = \begin{bmatrix} T_a^T D_\ell T_a & 0 \\ 0 & T_a^T D_r T_a \end{bmatrix} , \quad (5)$$

$$[k_{ab}] = \begin{bmatrix} T_a^T D_\ell T_b & 0 \\ 0 & T_a^T D_r T_b \end{bmatrix} , \quad (6)$$

$$[k_{bb}] = \begin{bmatrix} T_b^T D_\ell T_b & 0 \\ 0 & T_b^T D_r T_b \end{bmatrix} . \quad (7)$$

$$[k_{ba}] = [k_{ab}]^T . \quad (8)$$

The element damping matrices are equal to the stiffness matrices times g_e , the structural damping coefficient.

8.27.3 Lumped Mass Matrix Calculation (Subroutines MRØD and MTUBE of Module SMA2)

The total mass of the element, m , is

$$m = \rho A \ell + \mu \ell . \quad (9)$$

The partitions of the element mass matrix are:

$$[M_{aa}] = [M_{bb}] = \frac{m}{2} \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ - & - & - & - & - \\ & & & 0 & \\ & 0 & & & 0 \\ & & & & & 0 \end{bmatrix}, \quad (10)$$

$$[M_{ab}] = [M_{ba}] = [0]. \quad (11)$$

8.27.4 Element Load Calculations (Subroutines EDTL of Module SSG1)

The element loading calculations are calculated using the EST data, the element loading temperature, \bar{T} , and the enforced deformation δ .

1. Calculate λ , $\{n\}$, $[T_a]$ and $[T_b]$ as in Section 8.27.2.
2. Calculate:

$$\bar{T} = (t_a + t_b)/2 - T_0, \quad (12)$$

$$\{P_a\} = \frac{EA}{\lambda} \begin{bmatrix} [T_a]^T \\ - \\ 0 \end{bmatrix} \{n\} (\delta + \alpha \lambda \bar{T} - \alpha \lambda T_0), \quad (13)$$

$$\{P_b\} = -\frac{EA}{\lambda} \begin{bmatrix} [T_b]^T \\ - \\ 0 \end{bmatrix} \{n\} (\delta + \alpha \lambda \bar{T} - \alpha \lambda T_0). \quad (14)$$

3. $\{P_a\}$ and $\{P_b\}$ are placed in the load vector in positions corresponding to points a and b.

8.27.5 Element Stress Calculations (Subroutines SRØD1 and SRØD2 of Module SDR2)

The stress functions calculated in phase 1 (Subroutine SRØD1) are:

$$[S_a^t] = \frac{E}{\lambda} \{n\}^T [T_a], \quad (1 \times 3); \quad (15)$$

$$[S_b^t] = - \frac{E}{\ell} \{n\}^T [T_b], \quad (1 \times 3); \quad (16)$$

$$[S_a^r] = \frac{GC}{\ell} \{\eta\}^T [T_b], \quad (1 \times 3); \quad (17)$$

$$[S_b^r] = - \frac{GC}{\ell} \{n\}^T [T_b], \quad (1 \times 3). \quad (18)$$

$$S_T = - \alpha E, \quad (19)$$

$$S_\delta = - \frac{E}{\ell}. \quad (20)$$

The miscellaneous constants A , $\frac{J}{C}$, T_0 , σ_t , σ_c and σ_s are also saved for phase 2 calculations.

Note J/C is set to zero if $C = 0$. The superscripts t and r denotes translational and rotational stress matrices respectively.

The stress and force values are calculated in phase 2 (Subroutine SRØD2) using the displacement vectors $\{u_a\}$ and $\{u_b\}$, the loading temperature, \bar{T} , and the enforced deformation δ . Note that $\{u_i^t\}$ and $\{u_i^r\}$ ($i = a, b$) denote the 3 by 1 translational and rotational components of $\{u_i\}$.

1. Partition

$$\{u_a\} \Rightarrow \begin{pmatrix} u_a^t \\ u_a^r \end{pmatrix}, \quad (21)$$

$$\{u_b\} \Rightarrow \begin{pmatrix} u_b^t \\ u_b^r \end{pmatrix}. \quad (22)$$

2. The stresses are:

$$\sigma = [S_a^t] \{u_a^t\} + [S_b^t] \{u_b^t\} + S_\delta \delta + S_T [\bar{T} - T_0], \quad (23)$$

$$\tau = [S_a^r] \{u_a^r\} + [S_b^r] \{u_b^r\}. \quad (24)$$

The margins of safety in tension or compression, $M.S._t$ or $M.S._c$ respectively, are calculated as follows:

If $\sigma \geq 0$ then:

$$M.S. = \begin{cases} \frac{\sigma_t}{\sigma} - 1, & \sigma_t > 0 \\ \text{Integer "1", } & \sigma_t \leq 0 \text{ or } \sigma = 0 \end{cases} \quad (25)$$

If $\sigma < 0$, then: define $\sigma'_c = -|\sigma_c|$.

$$M.S. = \begin{cases} \frac{\sigma'_c}{\sigma} - 1, & \sigma'_c \neq 0 \\ \text{Integer "1", } & \sigma = 0 \text{ or } \sigma'_c = 0 \end{cases} \quad (26)$$

The margin of safety for torsion

$$M.S. = \begin{cases} \frac{\sigma_s}{|\tau|} - 1, & \sigma_s > 0 \text{ and } \tau \neq 0 \\ \text{Integer "1", } & \sigma_s \leq 0 \text{ or } \tau = 0 \end{cases} \quad (26a)$$

The forces are:

$$P = A\sigma, \quad (27)$$

$$T = \frac{J}{C} \tau. \quad (28)$$

8.27.6 Differential Stiffness Matrix Calculation (Subroutine DRD of Module DSMG1)

The data input from the ECPT, MPT and CSTM data blocks are as listed in Section 8.27.1. The following variables are calculated in the same manner as those for the stiffness matrix variables (Section 8.27.2).

l length of rod

$\left. \begin{matrix} n_1 \\ n_2 \\ n_3 \end{matrix} \right\}$ The direction cosines of the rod axis (+ from b to a) in the basic coordinate system

$[T_a], [T_b]$ The transformation matrices from global coordinates to the basic coordinate system at the grid points.

THE ROD, CONROD AND TUBE ELEMENTS

Only the linear translational displacements at the grid points are extracted from the displacement vector. Call these $\{u_a^t\}$ and $\{u_b^t\}$ (3x1 single precision vectors.)

1. Calculate the axial load in the element (+ implies tension):

$$\frac{F_x}{l} = \frac{AE}{l^2} \left\{ \{n\}^T \left[[T_a] \{u_a^t\} - [T_b] \{u_b^t\} \right] - \delta - \alpha l \bar{T} - T_0 \right\} . \quad (29)$$

2. A pair of axes perpendicular to the rod axis is constructed. Select the smallest component of $\{n\}$. Define n_i as the component of $\{n\}$ which is the smallest; let j and k be the other two components. Construct $\{m\}$ such that

$$m_i = 1, \quad m_j = m_k = 0. \quad (30)$$

Let

$$\{y\} = \frac{\{m\} \times \{n\}}{|\{m\} \times \{n\}|} , \quad (31)$$

and

$$\{z\} = \frac{\{n\} \times \{y\}}{|\{n\} \times \{y\}|} . \quad (32)$$

where \times denotes the cross product.

The actual partitions of the differential stiffness matrix relating to displacements in global coordinates are:

$$[K_{aa}^d] = \frac{F_x}{l} [T_a]^T [\{y\}\{y\}^T + \{z\}\{z\}^T] [T_a] , \quad (33)$$

$$[K_{ab}^d] = \frac{-F_x}{l} [T_a]^T [\{y\}\{y\}^T + \{z\}\{z\}^T] [T_b] , \quad (34)$$

$$[K_{bb}^d] = \frac{F_x}{l} [T_b]^T [\{y\}\{y\}^T + \{z\}\{z\}^T] [T_b] , \quad (35)$$

$$[K_{ba}^d] = [K_{ab}^d]^T . \quad (36)$$

The actual 6x6 partitions are formed by expanding:

$$[k_{aa}^d] \Rightarrow \begin{bmatrix} k_{aa}^d & 1 & 0 \\ - & - & - \\ 0 & 0 & 0 \end{bmatrix}, \text{ etc.} \quad (37)$$

8.27.7 Piecewise Linear Analysis Calculations (Subroutine PSR0D of Module PLA3 and Subroutine PKR0D of Module PLA4)

The additional ECPTNL and ESTNL entries are:

- ϵ_0^* - The previously computed strain value once removed.
- ϵ^* - The previously computed strain value.
- E^* - The previously computed modulus of elasticity.
- T^* - The previously computed torsional moment (present in the ESTNL entry only).

All of the above values are initially zero with the exception of E^* , which is initially the original modulus of elasticity present on a MAT1 card.

For both stress calculation and stiffness matrix generation, the quantities l and $\{n\}$ are generated as in Section 8.27.2.

Using $\{\Delta u_a^t\}$ and $\{\Delta u_b^t\}$, the 3x1 translational displacement vectors, calculate the increment of strain:

$$\Delta \epsilon = \frac{1}{l} \{n\}^T \left[[T_a] \{\Delta u_a^t\} - [T_b] \{\Delta u_b^t\} \right] , \quad (38)$$

$$\Delta \epsilon^* = \epsilon^* - \epsilon_0^* . \quad (39)$$

Define the following terms:

$$\epsilon_1 = \epsilon^* + \Delta \epsilon, \quad (\text{current strain}); \quad (40)$$

$$\epsilon_2 = \epsilon_1 + \gamma(\Delta \epsilon) , \quad (41)$$

(estimated next strain);

where γ is the ratio of the next load increment to the present load increment.

We calculate:

$$\sigma_1 = f(\epsilon_1), \quad (42)$$

$$\sigma_2 = f(\epsilon_2), \quad (43)$$

where f is the tabular stress-strain function. (When $\epsilon^* = 0$, define $\sigma_1 = E_0 \epsilon_1$)

For stiffness matrix generation, the new material properties are:

$$E = \begin{cases} \frac{\sigma_2 - \sigma_1}{\epsilon_2 - \epsilon_1}, & \text{if } \epsilon_2 \neq \epsilon_1 \\ E^*, & \text{if } \epsilon_2 = \epsilon_1 \end{cases}; \quad (44)$$

and

$$G = \frac{E}{E_0} G_0, \quad (45)$$

where E_0 and G_0 are elastic moduli obtained from the MAT1 bulk data card via subroutine MAT.

For plastic element stresses and forces the values are:

$$\sigma = \sigma_1, \quad (46)$$

$$P = A\sigma_1, \quad (47)$$

$$T = \frac{JG^*}{2} \{n\}^T ([T_a]\{\Delta u_a^r\} - [T_b]\{\Delta u_b^r\}) + T^*, \quad (48)$$

$$\tau = \frac{CT}{J} \quad (\tau = 0 \text{ if } C = 0 \text{ or } J = 0), \quad (49)$$

where

$$G^* = \frac{E^*}{E_0} G_0. \quad (50)$$

STRUCTURAL ELEMENT DESCRIPTIONS

The new ESTNL and ECPTNL entries are:

$$\epsilon_{on}^* = \epsilon^* , \quad (51)$$

$$\epsilon_n^* = \epsilon_1 , \quad (52)$$

$$E_n^* = \frac{\sigma_2 - \sigma_1}{\epsilon_2 - \epsilon_1} , \quad (53)$$

$$T_n^* = T . \quad (54)$$

8.27.8 Coupled Mass Matrix Calculation (Subroutine MCRØD of Module SMA2)

1. The length of the element, l , the normalized direction vector, $\{n\}$, and the mass of the element, m , are calculated as in Equations 1 and 2 in Section 8.27.2 and Equation 9 in Section 8.27.3, respectively.

2. The 3 by 3 matrix

$$[\Delta M] = \frac{m}{12} \begin{bmatrix} n_1^2 & n_1 n_2 & n_1 n_3 \\ n_1 n_2 & n_2^2 & n_2 n_3 \\ n_1 n_3 & n_2 n_3 & n_3^2 \end{bmatrix} \quad (55)$$

is calculated.

3. The 3 by 3 element mass matrices in basic coordinates are

$$[m_{aa}] = [m_{bb}] = \begin{bmatrix} m/2 & 0 & 0 \\ 0 & m/2 & 0 \\ 0 & 0 & m/2 \end{bmatrix} - [\Delta M], \quad (56)$$

and

$$[m_{ab}] = [m_{ba}] = [\Delta M]. \quad (57)$$

4. In global coordinates, the 6 by 6 mass matrix partitions are:

$$[M_{ij}] = \begin{bmatrix} T_i^T m_{ij} T_j & 0 \\ 0 & 0 \end{bmatrix}, \quad (58)$$

for $i = a$ or b , $j = a$ or b , and where $[T_i]$ is the global-to-basic coordinate transformation matrix for point i .

8.27.9 Thermal Analysis Calculations for the RØD Elements

If a "stiffness" matrix for thermal analysis is to be generated in subroutine KRØD, word 56 in CØMMØN data block SYSTEM is +1. The length, l , of the element is calculated as in Section 8.27.2. The thermal material coefficient, k , is obtained by calling subroutine HMAT, rather than MAT. The matrix terms are:

STRUCTURAL ELEMENT DESCRIPTIONS

For the pivot point, $i = 1$ or 2 :

$$K_{ii} = \frac{k A}{l}.$$

For $j \neq i$,

$$K_{ij} = - \frac{k A}{l}.$$

The "mass" matrix for thermal analysis is calculated in subroutine MRØD. The thermal capacity, C_p , is given by subroutine HMAT. The matrix terms, to be placed in matrix BGG, are:

$$B_{ii} = \frac{C_p A l}{2} \quad i = 1, 2.$$

The heat flux and gradient "stress" recovery is performed by subroutines SDHTF1, SDHTFF, and SDHTF2 in module SDR2. In Phase 1, the value H is extracted with subroutine HMAT and the output is:

$$K_1 = H$$

$$[C] = \frac{1}{l} [-1 \ 1].$$

In Phase 2, the gradient and flux are:

$$\Delta T = [C] \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix}$$

$$Q = - K_1 \Delta T$$

9. EIGENVALUE EXTRACTION METHODS

9.1 DETERMINANT METHOD OF EIGENVALUE EXTRACTION

NASTRAN contains two double precision versions of the determinant method. One version is for the solution of real eigenvalue problems; and one version is for the solution of complex eigenvalue problems. The specifications for both versions are discussed in this document. Real arithmetic is used for the real problem, complex for the complex problem.

9.1.1 Fundamentals of the Determinant Method

The basic notion employed in the determinant method of eigenvalue extraction is very simple. If the elements in a matrix $[A]$ are functions of the operator p , then the determinant of $[A]$ can be expressed as:

$$D([A]) = (p-p_1) (p-p_2) \dots (p-p_n), \quad (1)$$

where $p_1, p_2, p_3 \dots p_n$ are the eigenvalues of the matrix. The value of the determinant vanishes for $p = p_i, i = 1, 2, \dots, n$.

In the determinant method, the determinant is evaluated for trial values of p , selected according to some iterative procedure, and a criteria is established to determine when $D([A])$ is sufficiently small or when p is sufficiently close to an eigenvalue. The eigenvector is then found by solution of the equation:

$$[A] \{u\} = 0, \quad (2)$$

with one of the elements of $\{u\}$ set to unity.

9.1.2 Evaluation of Determinant

The most convenient procedure for evaluating the determinant of a matrix employs the triangular decomposition of the matrix: $[A] = [L] [U]$ where $[L]$ is a lower unit triangle (unit values on the diagonal). The determinant of $[A]$ is equal to the product of the diagonal terms of $[U]$. The matrix $[A]$ may be expressed as

$$[A] = -p[M] + [K], \quad (3)$$

for real eigenvalue problems and as

$$[A] = p^2[M] + p[B] + [K], \quad (4)$$

for complex eigenvalue problems.

9.1.3 Iteration Algorithm

Consider a series of determinants $D_{k-2}^{(i)}$, $D_{k-1}^{(i)}$, $D_k^{(i)}$ evaluated for trial values of the eigenvalue $p = p_{k-2}$, p_{k-1} , p_k . Then a better approximation to the eigenvalue is obtained from the following calculations:

Let

$$h_k = p_k - p_{k-1}, \quad (5)$$

$$\lambda_k = h_k / h_{k-1}, \quad (6)$$

$$\delta_k = 1 + \lambda_k. \quad (7)$$

Then

$$h_{k+1} = \lambda_{k+1} h_k, \quad (8)$$

$$p_{k+1} = p_k + h_{k+1}, \quad (9)$$

where

$$\lambda_{k+1} = \frac{-2 D_k^{(i)} \delta_k}{g_k \pm [g_k^2 - 4 D_k^{(i)} \delta_k \lambda_k (D_{k-2}^{(i)} \lambda_k - D_{k-1}^{(i)} \delta_k + D_k^{(i)})]^{1/2}}, \quad (10)$$

$$g_k = D_{k-2}^{(i)} \lambda_k^2 - D_{k-1}^{(i)} \delta_k^2 + D_k^{(i)} (\lambda_k + \delta_k). \quad (11)$$

The (+) or (-) sign in Equation 10 is selected to minimize the absolute value of λ_{k+1} . In the case where p_k , p_{k-1} and p_{k-2} are all arbitrarily selected initial values (starting points), the starting points should be arranged such that $|D_k| \leq |D_{k-1}| \leq |D_{k-2}|$ and the (+) or (-) sign in Equation 10 should be selected to minimize the distance from p_{k+1} to all 3 starting points.

In a real eigenvalue analysis it is possible to calculate a complex λ_{k+1} . In order to

preclude the use of complex arithmetic in a real eigenvalue analysis problem, only the real part of the λ_{k+1} should be used to estimate a p_{k+1} .

9.1.4 Scaling

In calculating the determinant of $[A]$, the determinant is scaled by powers of 10 since the accumulated product will rapidly overflow or underflow the floating point size of a digital computer. All operations using the determinant are calculated in scaled arithmetic.

9.1.5 Sweeping of Previously Extracted Eigenvalues

Once an eigenvalue has been found to satisfactory accuracy, a return to that eigenvalue by the iteration algorithm can be prevented by dividing the determinant by the factor $(p-p_i')$ where p_i' is the accepted approximation to p in all subsequent calculations.

Thus:

$$D^{(1)}([A]) = \frac{D([A])}{p-p_1'} \quad (12)$$

should be used in place of $D([A])$ after the first eigenvalue has been found. In general, the reduced determinant used for finding the $i+1^{st}$ eigenvalue is:

$$D^{(i)}([A]) = \frac{D^{(i-1)}([A])}{(p-p_1')} = \frac{D([A])}{(p-p_1')(p-p_2')\cdots(p-p_{i-1}')} \quad (13)$$

This sweeping procedure is quite satisfactory provided that all p_i' have been calculated to an accuracy that is limited only by round-off error.

For problems in which zero is an eigenvalue, the number of such eigenvalues is specified by the user. In using the determinant method, zero eigenvalues should be eliminated from the determinant by a preliminary operation,

$$D^{(0)}([A]) = \frac{D([A])}{p^m}, \quad (14)$$

where m is the multiplicity of the zero eigenvalue.

For problems with conjugate complex eigenvalues (complex eigenvalue analysis with real matrices) the conjugates of extracted eigenvalues should also be swept from the determinant. Thus

$$D^{(i)}([A]) = \frac{D^{(i-1)}([A])}{(p-p_i')(p-\bar{p}_i')} , \quad (15)$$

where \bar{p}_i' is the conjugate of p_i' . It should be noted that the sweeping equations are indeterminate for $p = p_i'$. This situation will occur when a root coincides with a starting point or a new estimate with a root already extracted. When the first situation occurs, the starting point should be moved away from the root. When the second situation occurs, D_{k+1} should be set equal to D_k .

9.1.6 Convergence Criteria

Convergence criteria are based on successive values of the increment h_k in the estimated eigenvalue. No tests on the magnitude of the determinant or any of the diagonal terms of the triangular decomposition are necessary or desirable. Wilkinson⁽¹⁾ shows that for h_k sufficiently small, the magnitude of h_k is approximately squared for each successive iteration when using Muller's method. This is an extremely rapid rate of convergence. In a very few iterations the "zone of indeterminacy" is reached within which h_k remains small but exhibits random behavior due to round-off error. Wilkinson states that if it is desired to calculate the root to the greatest possible precision, the convergence criterion for accepting p_k as a root should be:

$$|h_{k+1}| > |h_k|. \quad (16)$$

The determinant method accepts this advice, tempered by practical considerations. The first of these is that Equation 16 may be falsely satisfied during the first few iterations while the root tracking algorithm is picking up the "scent". Thus it must, in addition, be required that $|h_k|$, $|h_{k-1}|$ and $|h_{k-2}|$ be reasonably small. The second practical consideration is that we may waste a few iterations within the zone of indeterminacy while waiting for Equation 16 to be satisfied. This is avoided by accepting p_k if $|h_k|$ is sufficiently small. Finally, if the number of iterations becomes excessively large without satisfying a convergence criterion, the Determinant Method assumes the existence of one iteration loop, gives up and proceeds to

DETERMINANT METHOD OF EIGENVALUE EXTRACTION

a new set of starting points.

Figure 1 is a flow diagram of a set of tests which meet the requirements discussed above for real eigenvalue problems. The tests are based on calculated values of \overline{H}_1 , \overline{H}_2 , and \overline{H}_3 which are defined as:

$$\overline{H}_1 = |h_{k-1}| / \sqrt{|p_k|}. \quad (17)$$

$$\overline{H}_2 = |h_k| / \sqrt{|p_k|}. \quad (18)$$

$$\overline{H}_3 = |h_{k+1}| / \sqrt{|p_k|}. \quad (19)$$

where $p_k = k^{\text{th}}$ estimate of an eigenvalue and $h_k = p_k - p_{k-1}$.

A similar set of tests is described in Figure 2 for complex eigenvalue problems. In this case \overline{H}_1 , \overline{H}_2 and \overline{H}_3 are defined as:

$$\overline{H}_1 = |h_{k-1}|. \quad (20)$$

$$\overline{H}_2 = |h_k|. \quad (21)$$

$$\overline{H}_3 = |h_{k+1}|. \quad (22)$$

The magnitude of the convergence criterion ϵ should be selected as a compromise between running time and accuracy. Currently $\epsilon = 10^{-11}$. If failure occurs because the number of iterations exceeds the iteration limit, NIT, for two successive sets of starting points, ϵ is increased by a factor of 10. If successive pairs of failures still occur, ϵ is again increased until the number of permissible changes in ϵ is exceeded. The user is informed of the reduced precision of the calculations.

Since eigenvalues are swept out after they are found, all sets of starting points will eventually lead to failure by the preliminary range checks or through successive iteration failure. When this occurs it is presumed that all eigenvalues within the range of interest have been found and the calculations are halted. If for some reason this does not occur the calculation must still be halted. The one remaining avenue for the computer to continue calculations indefinitely is if it continues to find roots. To block this avenue, the calcula-

tions are stopped if the number of roots found exceeds the maximum number of N_{EVM} . As a safeguard the order in which the roots are found is indicated to the user.

9.1.7 Extraction of Eigenvectors

Once an approximate eigenvalue p_j has been accepted, the eigenvector is determined by back substitution into the previously computed triangular decomposition of $[A(p_j)]$. Now since

$$[A(p_j)]\{u\} = [L(p_j)][U(p_j)]\{u\} = 0, \quad (23)$$

we work only with $[U(p_j)]$. Because partial pivoting (row interchanges) have been used, the last diagonal term in $[U(p_j)]$ will normally be the only term with a very small value. The normal appearance of $[U(p_j)]$ is as follows, for $n = 7$:

$$\begin{bmatrix} X & & & & 0 & 0 & 0 \\ & X & & & X & 0 & 0 \\ & & X & & X & X & 0 \\ & & & X & X & X & X \\ & 0 & & & X & X & X \\ & & & & & X & X \\ & & & & & & \delta \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{Bmatrix} = \{0\}. \quad (24)$$

The terms in the upper right corner are zero due to bandwidth. δ is a very small number. The eigenvector may be extracted by assigning an arbitrary value (such as 1.0) to u_7 and solving successively for u_6 , u_5 etc., from the preceeding rows. Note that this is equivalent to placing a load vector $\{F\}$ on the right-hand side that is null except for the last term which is set equal to δ .

Situations may occur in which U_{nn} is not the smallest diagonal term. Let U_{ii} be the smallest diagonal term with $i < n$. The most common reason for this occurrence is that the degrees of freedom u_{i+1} , u_{i+2} , ..., u_n are, for some reason, uncoupled to the preceeding degrees of freedom. In this case all of the elements in the i^{th} row of $[U(p_j)]$ will be very small as shown for $i = 4$, $n = 7$:

DETERMINANT METHOD OF EIGENVALUE EXTRACTION

$$\begin{bmatrix}
 X & X & X & X & 0 & 0 & 0 \\
 & X & X & X & X & 0 & 0 \\
 & & X & X & X & X & 0 \\
 & & & \delta_{44} & \delta_{45} & \delta_{46} & \delta_{47} \\
 & 0 & & & X & X & X \\
 & & & & & X & X \\
 & & & & & & X
 \end{bmatrix}
 \begin{Bmatrix}
 u_1 \\
 u_2 \\
 u_3 \\
 u_4 \\
 u_5 \\
 u_6 \\
 u_7
 \end{Bmatrix}
 = \{0\}. \quad (25)$$

In the event of multiple or pathologically close eigenvalues two or more rows of $[U(p_j)]$ will consist of very small values, exhibited below for the very exceptional case where the n^{th} row is not very small:

$$\begin{bmatrix}
 X & X & X & X & 0 & 0 & 0 \\
 & X & X & X & X & 0 & 0 \\
 & & X & X & X & X & 0 \\
 & & & \delta_{44} & \delta_{45} & \delta_{46} & \delta_{47} \\
 & 0 & & & X & X & X \\
 & & & & & \delta_{66} & \delta_{67} \\
 & & & & & & X
 \end{bmatrix}
 \begin{Bmatrix}
 u_1 \\
 u_2 \\
 u_3 \\
 u_4 \\
 u_5 \\
 u_6 \\
 u_7
 \end{Bmatrix}
 = \{0\}. \quad (26)$$

In order to accommodate the exceptional cases described above with the more general case of $\delta = U_{nn}$, a full load vector $\{F\}$ is used for the eigenvector calculations. The load vector will also contain elements of the same order of magnitude as the smallest diagonal element of the triangularized matrix $[U(p_j)]$ in order to prevent digital overflow when the eigenvector is calculated. In addition, a distinct load vector is formed for each eigenvalue to ensure that independent eigenvectors are calculated for multiple or pathologically close eigenvalue problems. The following equations are used for $\{F\}$. For real eigenvalues, we have

$$F_i = \frac{\delta(-1)^{ij}}{1 + (1 - \frac{1}{n})j} ; \quad (27)$$

EIGENVALUE EXTRACTION METHODS

For complex eigenvalues,

$$\text{Re}(F_i) = \frac{|\delta| (-1)^{ij}}{1.0 + (1.0 - \frac{1}{n}) j}, \quad (28)$$

$$\text{Im}(F_i) = 0.0, \quad (29)$$

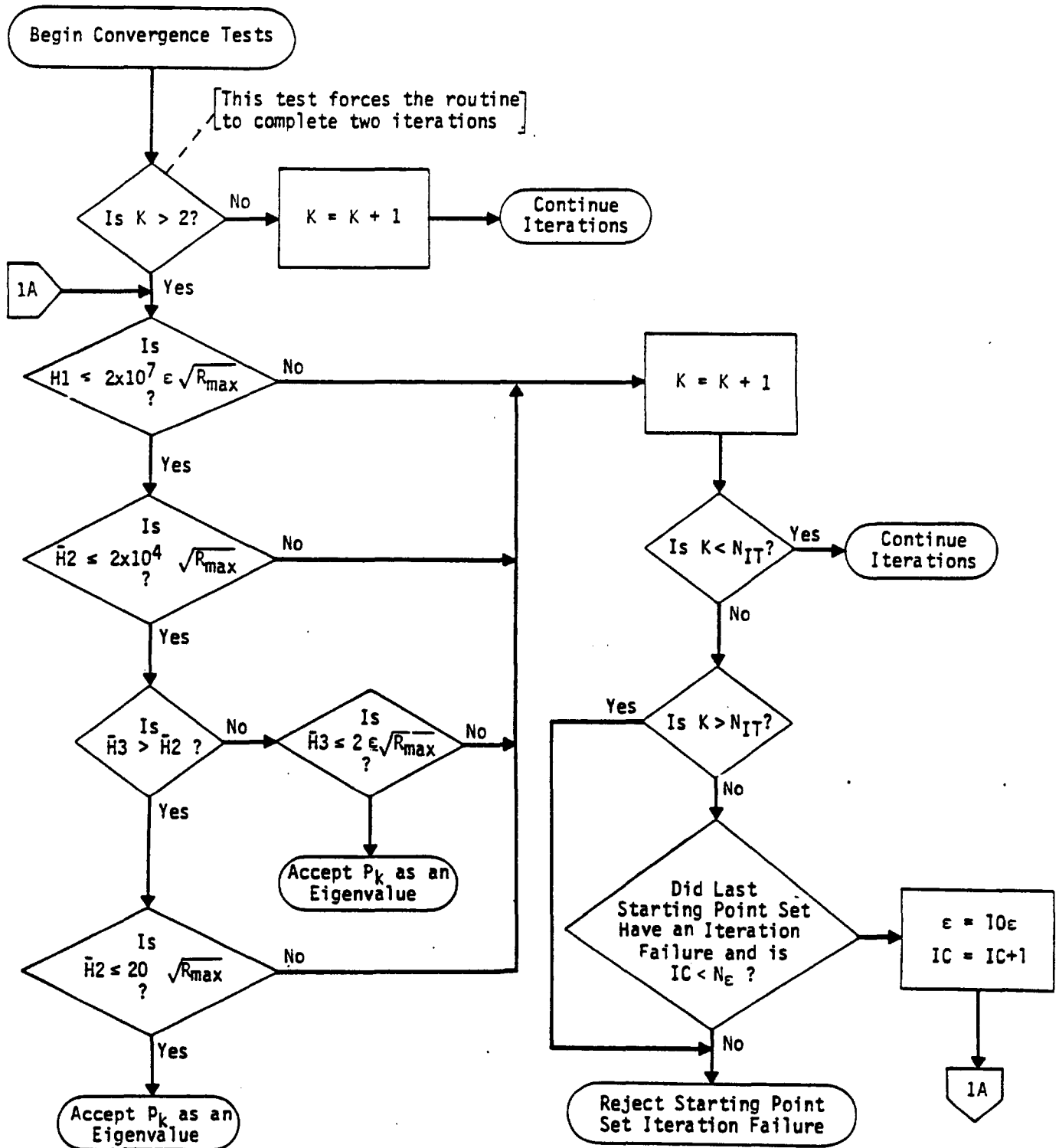
where δ = smallest U_{ii} , j = eigenvalue count, $i = i^{\text{th}}$ element of $\{F\}$ and n = number of rows.

There is a possibility that the smallest diagonal element of $[U]$ may be exactly zero for some eigenvalue. This will be the case when the accepted eigenvalue (p_j) is exactly equal to an eigenvalue of the problem. When this occurs, $\delta = 1.0 \times 10^{-8}$. The calculated eigenvectors are normalized to a unit maximum real element value.

REFERENCE

1. Wilkinson, J.H., "The Algebraic Eigenvalue Problem", Clarendon Press, Oxford, 1965.

DETERMINANT METHOD OF EIGENVALUE EXTRACTION



ϵ - Convergence Criterion

K - Iteration Counter

IC - Criterion Change Counter

$$H1 = |h_{k-1}| / \sqrt{|p_k|}$$

$$H2 = |h_k| / \sqrt{|p_k|}$$

$$H3 = |h_{k+1}| / \sqrt{|p_k|}$$

Figure 1. Real eigenvalue problems convergence tests

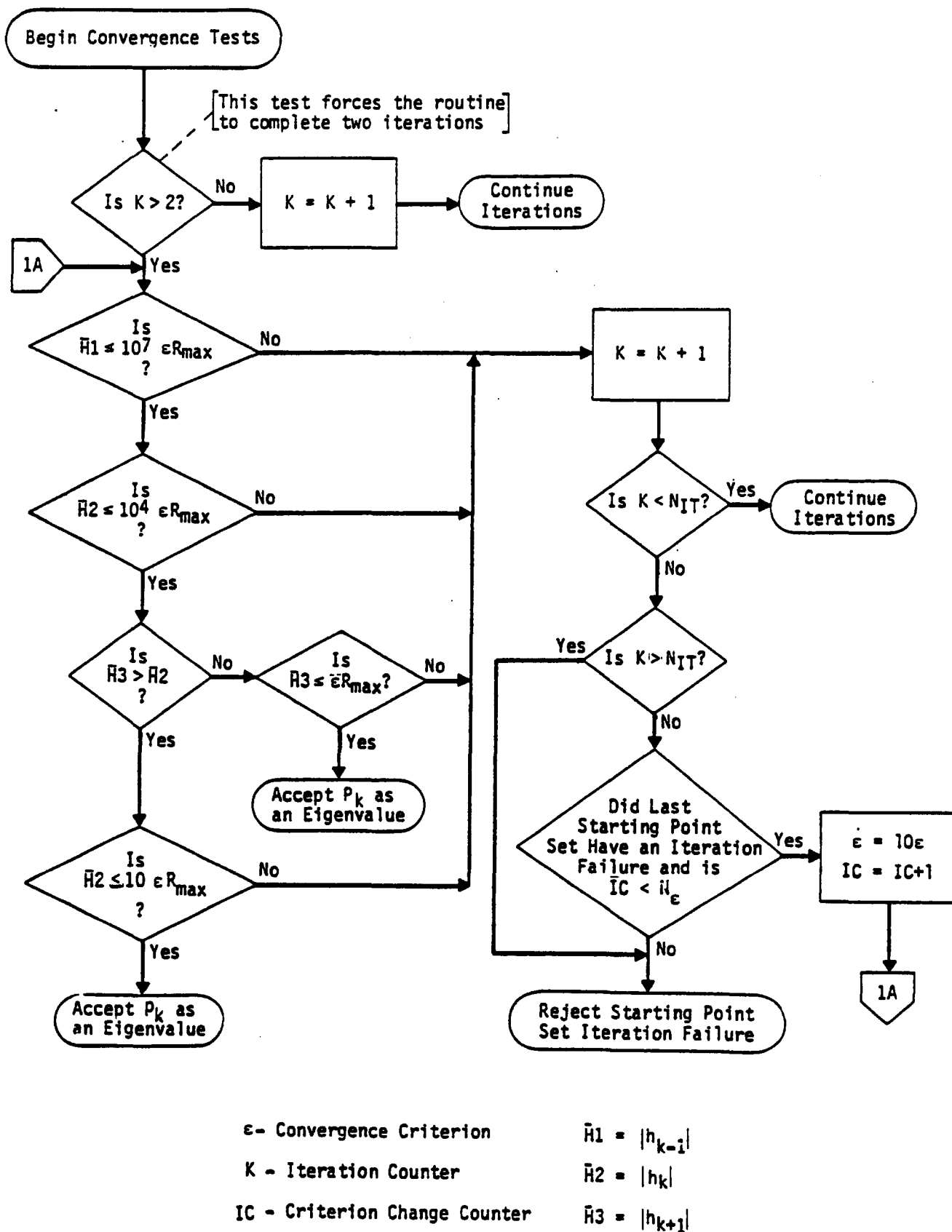


Figure 2. Complex eigenvalue problems convergence tests

EIGENVALUE EXTRACTION METHODS

9.2 GIVENS METHOD OF EIGENVALUE EXTRACTION

9.2.1 General

The most stable methods of solution of eigenproblems for large symmetric matrices are based on the tridiagonalization techniques of Givens (Reference 1), Lanczos (Reference 2), and Householder (Reference 3) followed by some method of determination of the eigensolution for a tridiagonal matrix. The Givens method depends on orthogonal transformations $[T] [A] [T]^T$ of a symmetric matrix $[A]$ of order $n \times n$. An orthogonal transformation is one whose matrix $[T]$ satisfies

$$[T] [T]^T = [T]^T [T] = [I] \quad (1)$$

The eigenvalues of a matrix are preserved under orthogonal transformation since

$$[T] ([A] - \lambda[I]) [T]^T = [T] [A] [T]^T - \lambda[I] \quad (2)$$

and if $\det ([A] - \lambda[I])$ vanishes, then so does $\det ([T] [A] [T]^T - \lambda[I])$.

The effect of a series of orthogonal transformations on the eigenvectors of a matrix is that of a succession of multiplications by orthogonal matrices. For if

$$[A] \{x\} = \lambda \{x\} \quad (3)$$

and if $[T_1], [T_2], \dots, [T_r]$ are orthogonal matrices then,

$$[T_r] [T_{r-1}] \cdots [T_2] [T_1] [A] \{x\} = \lambda [T_r] [T_{r-1}] \cdots [T_2] [T_1] \{x\} \quad (4)$$

Write $\{x\} = [T_1]^T [T_2]^T \cdots [T_{r-1}]^T [T_r]^T \{y\}$. Then

$$\begin{aligned} & [T_r] [T_{r-1}] \cdots [T_2] [T_1] [A] [T_1]^T [T_2]^T \cdots [T_{r-1}]^T [T_r]^T \{y\} \\ &= \lambda [T_r] [T_{r-1}] \cdots [T_2] [T_1] [T_1]^T [T_2]^T \cdots [T_{r-1}]^T [T_r]^T \{y\} \\ &= \lambda \{y\} \end{aligned} \quad (5)$$

It follows that $\{y\}$ is an eigenvector of the transformed matrix

$$[T_r] [T_{r-1}] \cdots [T_2] [T_1] [A] [T_1]^T [T_2]^T \cdots [T_{r-1}]^T [T_r]^T$$

and that $[x]$ may be obtained from $[y]$ by

$$\{x\} = [T_1]^T [T_2]^T \cdots [T_{r-1}]^T [T_r]^T \{y\} \quad (6)$$

9.2.2 Reduction to Triangular Form.

Givens method uses orthogonal matrices $[T]$ which are identical with the unit matrix $[I]$ except for the four elements

$$\left. \begin{aligned} t_{i+1,i+1} &= t_{j,j} = \cos \theta_{i+1,j} \\ t_{i+1,j} &= -t_{j,i+1} = \sin \theta_{i+1,j} \end{aligned} \right\} \quad (7)$$

The orthogonal transformation $[T] [A] [T]^T$ leaves unchanged all the elements of $[A]$ except those in the $i+1^{\text{th}}$ and j^{th} rows and columns, the so-called plane of rotation.

The four pivotal elements then become $\alpha_{i+1,i+1}$, $\alpha_{j,j}$, $\alpha_{i+1,j}$, and $\alpha_{j,i+1}$ given by

$$\left. \begin{aligned} \alpha_{i+1,i+1} &= a_{i+1,i+1} \cos^2 \theta_{i+1,j} + a_{j,j} \sin^2 \theta_{i+1,j} + a_{i+1,j} \sin 2\theta_{i+1,j} \\ \alpha_{j,j} &= a_{i+1,i+1} \sin^2 \theta_{i+1,j} + a_{j,j} \cos^2 \theta_{i+1,j} - a_{i+1,j} \sin 2\theta_{i+1,j} \\ \alpha_{i+1,j} &= \alpha_{j,i+1} = a_{i+1,j} \cos 2\theta_{i+1,j} - (a_{i+1,i+1} - a_{j,j}) \sin 2\theta_{i+1,j} / 2 \end{aligned} \right\} \quad (8)$$

and the other elements of the $i+1^{\text{th}}$ and j^{th} rows and columns are:

$$\left. \begin{aligned} \alpha_{i+1,s} &= \alpha_{s,i+1} = a_{i+1,s} \cos \theta_{i+1,j} + a_{j,s} \sin \theta_{i+1,j} \\ \alpha_{j,s} &= \alpha_{s,j} = -a_{i+1,s} \sin \theta_{i+1,j} + a_{j,s} \cos \theta_{i+1,j} \end{aligned} \right\} \quad (9)$$

Givens chooses $\theta_{i+1,j}$ such that $\alpha_{i,j}$ will vanish, which happens when

$$\tan \theta_{i+1,j} = a_{i,j} / a_{i+1,i} \quad (10)$$

The Givens' process, which is the calculation of $\theta_{i+1,j}$ followed by the orthogonal transformation

$$[A^{(r)}] = [T_r] [A^{(r-1)}] [T_r]^T, \quad (11)$$

GIVENS METHOD OF EIGENVALUE EXTRACTION

is carried out for $r = 1, 2, \dots$ with $[A^{(0)}] = [A]$, the values of i used are $1, 2, 3, \dots, (n - 2)$; and for each i , a set of $n - i - 1$ such transformations is performed, with j taking values $i + 2, i + 3, \dots, n$ before the next value of i is used. The result is that elements in the matrix positions $(1, 3), (1, 4), \dots, (1, n), (2, 4), (2, 5), \dots, (2, n), \dots, (n-2, n)$ are successively reduced to zero, together with their transposes, the $(3, 1), (4, 1), \dots, (n, n-2)$ elements. The final transformation of the set thus reduces the matrix to triple diagonal form.

Wilkinson (Reference 4) modifies the Givens' form by grouping together the $(n - i - 1)$ transformations which produce zeros in the i^{th} row and column. This process is particularly advantageous when the matrix $[A]$ is so large that the elements cannot all be left in the computing store at one time; it requires only $n - 2$ transfers of the matrix to and from auxiliary storage in place of the $(n-1)(n-2)/2$ transfers required by Givens' method. The method requires $4n$ words for working space and these are divided into four groups of n words each. The first $(i-1)$ rows and columns play no part in the i^{th} major step. This step has five main stages:

1. The i^{th} row of $[A]$ is transferred to the words in group 1.
2. The values of $\cos\theta_{i+1, i+2}, \sin\theta_{i+1, i+2}, \dots, \cos\theta_{i+1, n}, \sin\theta_{i+1, n}$ are computed successively from

$$\left. \begin{aligned} \cos\theta_{i+1, j} &= \frac{a_{i, i+1}^{(j-1)}}{\sqrt{(a_{i, i+1}^{(j-1)})^2 + a_{i, j}^2}} \\ \sin\theta_{i+1, j} &= \frac{a_{i, j}}{\sqrt{(a_{i, i+1}^{(j-1)})^2 + a_{i, j}^2}} \end{aligned} \right\} \quad (12)$$

where

$$\left. \begin{aligned} a_{i, i+1}^{(i+1)} &= a_{i, i+1} \\ a_{i, i+1}^{(j)} &= \sqrt{(a_{i, i+1}^{(j-1)})^2 + a_{i, j}^2} \end{aligned} \right\} \quad (13)$$

The $\cos\theta_{i, j}$ may be overwritten on the $a_{i, j}$, which are no longer required, and the $\sin\theta_{i+1, j}$ are stored in the group 2 words.

3. The $i+1^{\text{th}}$ row is transferred to the words in group 3. Only those elements on and above the diagonal are used in this and succeeding rows.

For $k = i + 2, i + 3, \dots, n$ in turn, the operations in Stages 4 and 5 are carried out.

4. The k^{th} row is transferred to the words in group 4. The elements $a_{i+1,i+1}$, $a_{i+2,k}$ and $a_{k,k}$ are subjected to the row and column operations involving $\cos\theta_{i,k}$ and $\sin\theta_{i,k}$.

For $\ell = k + 1, k + 2, \dots, n$ in turn, the part of the row transformation involving $\cos\theta_{i+1,k}$ and $\sin\theta_{i+1,k}$ is performed on $a_{i+1,\ell}$ and $a_{k,\ell}$.

For $\ell = k + 1, k + 2, \dots, n$ in turn, the part of the column transformation involving $\cos\theta_{i+1,\ell}$ and $\sin\theta_{i+1,\ell}$ is performed on $a_{i+1,k}$ and $a_{k,\ell}$.

Now all the transformations involving the i^{th} major step have been performed on all the elements of row k and on elements $i + 2, i + 3, \dots, n$ of row $i + 1$.

5. The completed k^{th} row is transferred to the backing store and we return to 4. with the next value of k .

When stages 4 and 5 have been completed for all appropriate values of k , the whole work on row $i + 1$ has also been completed. The values of $\cos\theta_{i+1,k}$ and $\sin\theta_{i+1,k}$ for $k = i + 2, i + 3, \dots, n$ and the modified elements of the i^{th} row (i.e., the first two elements of the codiagonal form) are written on auxiliary storage and row $i + 1$ is transferred to the group 1 words. Since the $i + 1^{\text{th}}$ row plays the same part in the next major step as did the i^{th} in the step just described, everything is then ready for stage 2 in the next major step. Stage 1 is, in fact, only required in the first major step because the appropriate row is already in the computing store in subsequent steps.

9.2.3 Computation of Eigenvalues for a Tridiagonal Matrix.

The Q-R transformation of Francis (Reference 5) is an orthogonal transformation $[Q_r]^T [A_r]$ of a matrix $[A_r]$ to give a matrix $[A_{r+1}]$, the orthogonal matrix $[Q_r]$ being such that $[A_r]$ may be factored into the product $[Q_r] [R_r]$ where $[R_r]$ is an upper triangular matrix. Thus

$$[A_r] = [Q_r] [R_r] \quad (14)$$

and

GIVENS METHOD OF EIGENVALUE EXTRACTION

$$\begin{aligned} [A_{r+1}] &= [Q_r]^T [A_r] [Q_r] \\ &= [Q_r]^T [Q_R] [R_r] [Q_r] \end{aligned}$$

that is

$$[A_{r+1}] = [R_r] [Q_r] \quad (15)$$

It follows that $[A_{r+1}]$ may be determined from $[A]$ by performing in succession the decomposition of equation (14) and the multiplication of equation (15). Francis has shown that if a matrix $[A] = [A_1]$ is non-singular, then the limiting case of $[A_r]$ as $r \rightarrow \infty$ will be an upper triangular matrix; because eigenvalues are preserved under orthogonal transformation, it follows that the diagonal elements of this limiting matrix are the eigenvalues of the original matrix $[A]$. In the case of the symmetric matrices, the matrix $[A_r]$ will, of course, tend to a diagonal form as $r \rightarrow \infty$.

Ortega and Kaiser (Reference 6) have developed a modification of the Q-R transformation for evaluation of eigenvalues of tridiagonal matrices such as those resulting from the application of Givens' method. The advantage of this approach over Francis' original method is that the transformations from $[A_r]$ to $[A_{r+1}]$ do not require the square rooting associated with the original method. If we write

$$\begin{aligned} [A_r] &= \begin{bmatrix} a_1 & b_2 & & & \\ b_2 & a_2 & b_3 & & \\ & & & b_{n-1} & a_{n-1} & b_n \\ & & & & b_n & a_n \end{bmatrix} \\ [A_{r+1}] &= \begin{bmatrix} \bar{a}_1 & \bar{b}_2 & & & \\ \bar{b}_2 & \bar{a}_2 & \bar{b}_3 & & \\ & & & \bar{b}_n & \bar{a}_n \end{bmatrix} \end{aligned}$$

and

$$[R_r] = \begin{bmatrix} r_1 & q_1 & t_1 & & \\ & r_2 & q_2 & t_2 & \\ & & & & \\ & & & & \\ & & & & r_n \end{bmatrix} \quad (16)$$

and denote the cosine and sine of the j^{th} rotation by c_j and s_j , then

$$\left. \begin{aligned} s_j &= \frac{b_{j+1}}{\left(p_j^2 + b_{j+1}^2\right)^{\frac{1}{2}}} \\ c_j &= \frac{b_j}{\left(p_j^2 + b_{j+1}^2\right)^{\frac{1}{2}}} \end{aligned} \right\} \quad j = 1, 2, \dots, n-1 \quad (17)$$

where

$$\left. \begin{aligned} p_1 &= a_1, \\ p_2 &= c_1 a_2 - s_1 b_2, \\ p_j &= c_{j-1} a_j - s_{j-1} c_{j-2} b_j, \end{aligned} \right\} \quad j = 3, 4, \dots, n \quad (18)$$

and

$$\left. \begin{aligned} r_j &= c_j p_j + s_j b_{j+1}, & j &= 1, 2, \dots, n-1 \\ r_n &= p_n \\ q_1 &= c_1 b_2 + s_1 a_2 \\ t_j &= s_j b_{j+2}, & j &= 1, 2, \dots, n-2 \end{aligned} \right\} \quad (19)$$

GIVENS METHOD OF EIGENVALUE EXTRACTION

On the other hand, from recombination we have

$$\left. \begin{aligned} \bar{a}_1 &= c_1 r_1 + s_1 q_1 \\ \bar{a}_j &= c_{j-1} c_j r_j + s_j q_j, & j &= 2, 3, \dots, n-1 \\ \bar{a}_n &= c_{n-1} r_n \\ \bar{b}_{j+1} &= s_j r_{j-1}, & j &= 1, 2, \dots, n-1 \end{aligned} \right\} \quad (20)$$

If we introduce the quantities g_j defined by

$$\left. \begin{aligned} g_1 &= b_1 \\ g_j &= c_{j-1} b_j, & j &= 2, 3, \dots, n \end{aligned} \right\} \quad (21)$$

Then from equations (19) and (20) we have

$$\left. \begin{aligned} \bar{a}_j &= (1 + s_j^2) g_j + s_j^2 a_{j+1}, \\ \bar{b}_{j+1}^2 &= s_j^2 (p_{j+1}^2 + b_{j+2}^2), & j &= 1, 2, \dots, n-2 \\ \bar{b}_n^2 &= s_{n-1}^2 p_n^2 \end{aligned} \right\} \quad (22)$$

while for the g_j and p_j^2 we have

$$\left. \begin{aligned} g_1 &= b_1 = a_1 \\ g_j &= a_j - s_{j-1}^2 (a_j + g_{j-1}) & j &= 2, 3, \dots, n \\ p_1^2 &= a_1^2 \\ p_j^2 &= g_j^2 / c_{j-1}^2 & \text{if } c_{j-1} &\neq 0 \\ &= c_{j-2}^2 b_j^2 & \text{if } c_{j-1} &= 0 \end{aligned} \right\} \quad (23)$$

In practice in the method of Ortega and Kaiser convergence of the tridiagonal matrix to a diagonal form is speeded up by origin shifting. Wilkinson (Reference 7) has shown that when eigenvalues are real, the best shift strategy is to reduce each diagonal element of $[A]$, and hence each eigenvalue, by a_n .

Another useful device in speeding up the determination of eigenvalues of tridiagonal matrices is to partition the matrix $[A]$ into the form

$$\left[\begin{array}{ccc|ccc|c}
 a_1 & b_1 & & & & & \\
 b_1 & a_2 & b_2 & & & & \\
 & & & & & & \\
 & & & a_{j-2} & b_{j-2} & & \\
 & & & b_{j-2} & a_{j-1} & & \\
 \hline
 & & & a_j & b_j & & \\
 & & & b_j & a_{j+1} & b_{j+1} & \\
 & & & & & & \\
 & & & & & b_{m-2} & a_{m-1} & b_{m-1} \\
 & & & & & b_{m-1} & a_m & \\
 \hline
 & & & & & & & a_{m+1} \\
 & & & & & & & \\
 & & & & & & & \\
 & & & & & & & a_n
 \end{array} \right] \quad (24)$$

In this matrix the lower right hand partition contains no off-diagonal elements so that all the diagonal elements are eigenvalues and the i^{th} row is the next lowest in which the term, b_{j-1} , below the diagonal vanishes. Now the eigenvalues of the matrix in the central block may be obtained separately. In the program implementation of this variant of the Ortega and Kaiser method, we write $D(i)$ for the diagonal element a_i , $\emptyset(i)$ for the square b_{i+1}^2 of the off-diagonal element, P for p_i^2 and later as $P + \emptyset(i)$, S for the current value of s_{i-1}^2 and of $1 - c_{i-1}^2$ and $S1$ for the previous value of S .

Then the algorithm for the first row of the partition is:

$$\begin{aligned}
 P &= D(J) \times D(J) \\
 S &= \emptyset(J)/(P + \emptyset(J)) \\
 S1 &= 0.0 \\
 U &= S \times (D(J) + D(J+1)) \\
 D(J) &= D(J) + U
 \end{aligned} \tag{25}$$

and for subsequent rows when $I = J + 1, M - 1$

$$\begin{aligned}
 G &= D(I) - U \\
 \text{If } S &= 1 \text{ then } P = (1 - S1) \times \emptyset(I - 1) \\
 &\quad \text{else } P = G^2/(1 - S) \\
 P &= P + \emptyset(I) \\
 \emptyset(I-1) &= P \times S \\
 S1 &= S \\
 S &= \emptyset(I)/P \\
 U &= S \times (G + D(I+1)) \\
 D(I) &= U + G
 \end{aligned} \tag{26}$$

This iteration described by equations (25) and (26) is continued until $\emptyset(M-2)$ vanishes and then $D(M-1)$ is an eigenvalue of the shifted matrix. The process is then continued until all the eigenvalues of the partitioned matrix have been extracted.

9.2.4 Determination of Eigenvectors

The eigenvector corresponding with any eigenvalue, λ , of the codiagonal matrix may be determined by solving $(n-1)$ of the equations:

$$\begin{aligned}
 (a_1 - \lambda)x_1 + b_2x_2 &= 0 \\
 b_2x_1 + (a_2 - \lambda)x_2 + b_3x_3 &= 0 \\
 b_{n-1}x_{n-2} + (a_{n-1} - \lambda)x_{n-1} + b_nx_n &= 0 \\
 b_nx_{n-1} + (a_n - \lambda)x_n &= 0
 \end{aligned} \tag{27}$$

EIGENVALUE EXTRACTION METHODS

The most convenient sets of equations to solve are either the first $(n - 1)$ or the last $(n - 1)$; if the first $(n - 1)$ equations are used, the solution is obtained by taking x_1 to be unity and back substituting in the equations to obtain values of x_2, x_3, \dots, x_n . Wilkinson (Reference 7) has shown that this method is unstable and has suggested the following approach. The equations may be written in the form of equation (26) but with right-hand sides b_1, b_2, \dots, b_n .

These equations are solved by the successive elimination of the variables x_1, x_2, \dots, x_{n-1} in their natural order but taking, as 'pivotal' row at each stage, that equation which has the largest coefficient of the variable which is being eliminated. At each stage there will be only two equations containing that variable. Because the equations are not necessarily used in their natural order the resulting equations must be written

$$\begin{aligned}
 p_1 x_1 + q_1 x_2 + r_1 x_3 &= c_1 \\
 p_2 x_2 + q_2 x_3 + r_2 x_4 &= c_2 \\
 &\vdots \\
 p_{n-2} x_{n-2} + q_{n-2} x_{n-1} + r_{n-2} x_n &= c_{n-2} \\
 p_{n-1} x_{n-1} + q_{n-1} x_n &= c_{n-1} \\
 p_n x_n &= c_n
 \end{aligned}
 \tag{28}$$

though usually many of the r_i will be zero. The c_i are derived from the b_i which may still be chosen arbitrarily. Wilkinson (Reference 7) shows that if the b_i are chosen so that each c_i is unity, in which case computation of the right hand side is avoided, then there are good reasons for expecting that the vector $\{b\}$ will not be deficient in the eigenvector being calculated. Figures 1 and 2 describes the algorithm developed for the solution of equations (27). Figure 1 presents the algorithm that sets up the equations in P,Q, and R form. These equations are then solved by the method of Figure 2.

GIVENS METHOD OF EIGENVALUE EXTRACTION

In the case of a double eigenvalue, the above method gives one eigenvector $\{x_1\}$. If we start with any $\{b\}$ orthogonal to $\{x_1\}$ and apply the previous algorithm convergence to $\{x_2\}$, the other eigenvector, will result. In this case if $\{x_1\}$ is given by

$$\{x_1\} = \{\alpha_1, \alpha_2, \dots, \alpha_n\} \quad (29)$$

we may take b as

$$\{b\} = - \sum_{s=2}^n (\alpha_s) / \alpha_1, 1, 1, \dots, 1 \quad (30)$$

In the more general case of multiple eigenvalues, if eigenvectors $\{x_1\}, \{x_2\}, \dots, \{x_m\}$ given by

$$\{x_s\} = \{\alpha_{1s}, \alpha_{2s}, \dots, \alpha_{ns}\}$$

have been obtained, a starter vector $\{b\}$ orthogonal to each of these eigenvectors may be obtained by taking the components $b_{m+1}, b_{m+2}, \dots, b_n$ as unity and solving the simultaneous equations

$$\begin{aligned} b_1 a_{11} + b_2 a_{21} + \dots + b_m a_{m1} &= - \sum_{s=m+1}^n (a_{s1}) \\ b_1 a_{12} + b_2 a_{22} + \dots + b_m a_{m2} &= - \sum_{s=m+1}^n (a_{s2}) \\ &\cdot \\ &\cdot \\ &\cdot \\ b_1 a_{1m} + b_2 a_{2m} + \dots + b_m a_{mm} &= - \sum_{s=m+1}^n (a_{sm}) \end{aligned} \quad (31)$$

Accumulated rounding errors in the algorithms to obtain the eigenvectors $\{x_1\}, \{x_2\}, \dots, \{x_k\}$ corresponding with k repeated eigenvalues will result in their not being exactly orthogonal to one another. The following Gram-Schmidt algorithm described by Beodwig (Reference 8) will produce an orthonormal set of k eigenvectors $\{y_s\}$ from the almost orthogonal set $\{x_s\}$:

$$\{y_1\} = \{x_1\} / || \{x_1\} ||$$

For $s = 2, k$

$$\{z_s\} = \{x_s\} - \sum_{t=1}^{s-1} (\{x_s\} \cdot \{y_t\}) y_t$$

$$\{y_s\} = \{z_s\} / || \{z_s\} ||$$

where $|| \{z_s\} ||$ denoted the Euclidean norm

$$\sqrt{z_{s1}^2 + z_{s2}^2 + \dots + z_{sn}^2} \quad (32)$$

of a vector $\{z_{s1}, z_{s2}, \dots, z_{sn}\}$, and where $(\{x_s\} \cdot \{y_t\})$ is the scalar product of the vectors $\{x_s\}$ and $\{y_t\}$.

Finally, when all the eigenvectors of the tridiagonal matrix have been extracted, the matrix multiplications of equation (6) are carried out to obtain the eigenvectors of the original matrix $[A]$.

9.2.5 Computer Program Flow

The procedures described in the preceding sections are broken into computer routines as follows:

Overall Control	-- VALVEC
Tridiagonalize Matrix	-- TRIDI
Computation of Eigenvalues	-- QRITER
Computation of Eigenvectors	-- WILVEC

Flow diagrams for VALVEC, TRIDI, QRITER and WILVEC are included here in Figure 3 through 7.

GIVENS METHOD OF EIGENVALUE EXTRACTION

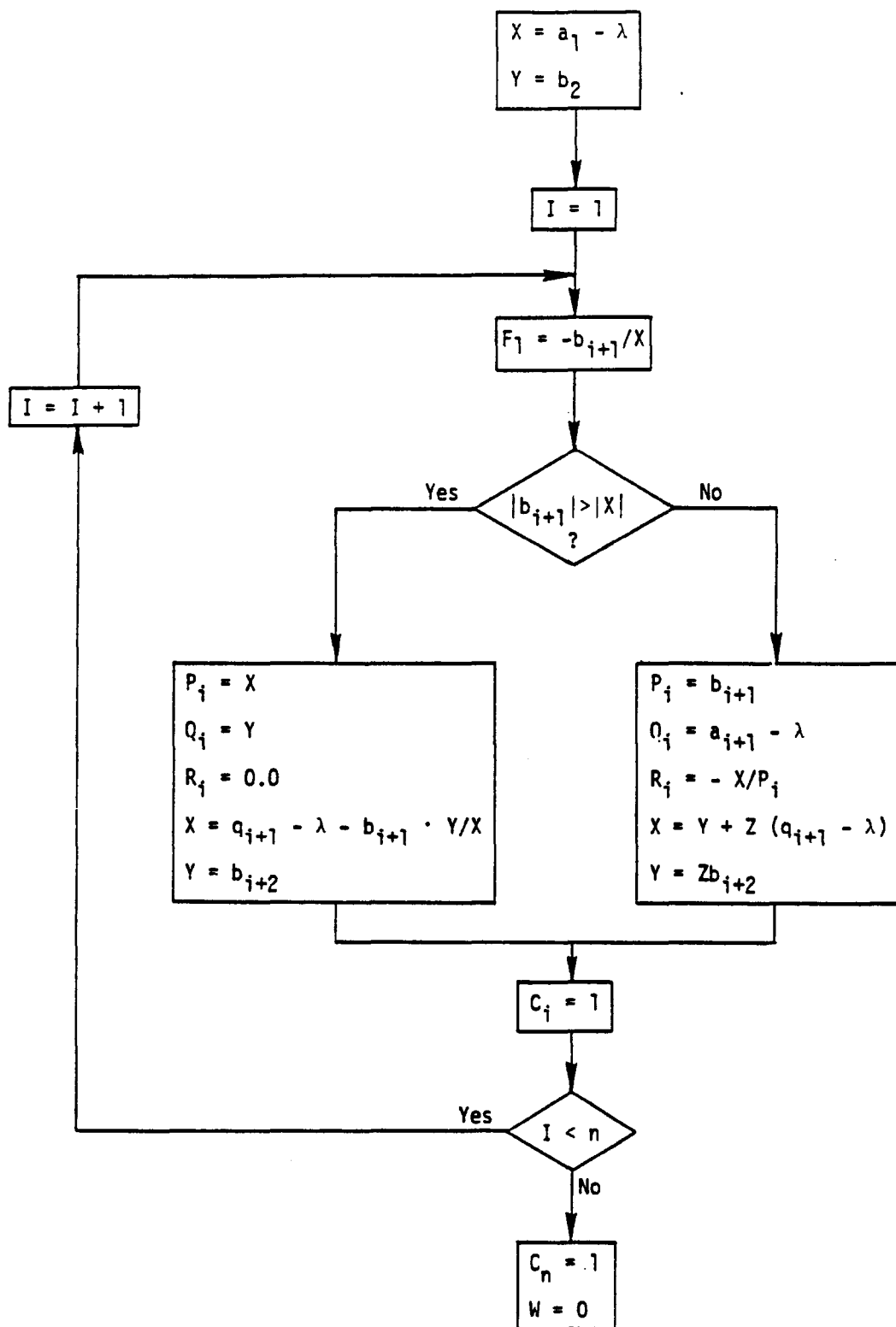


Figure 1. Eigenvalue solution algorithm equation definition.

EIGENVALUE EXTRACTION METHODS

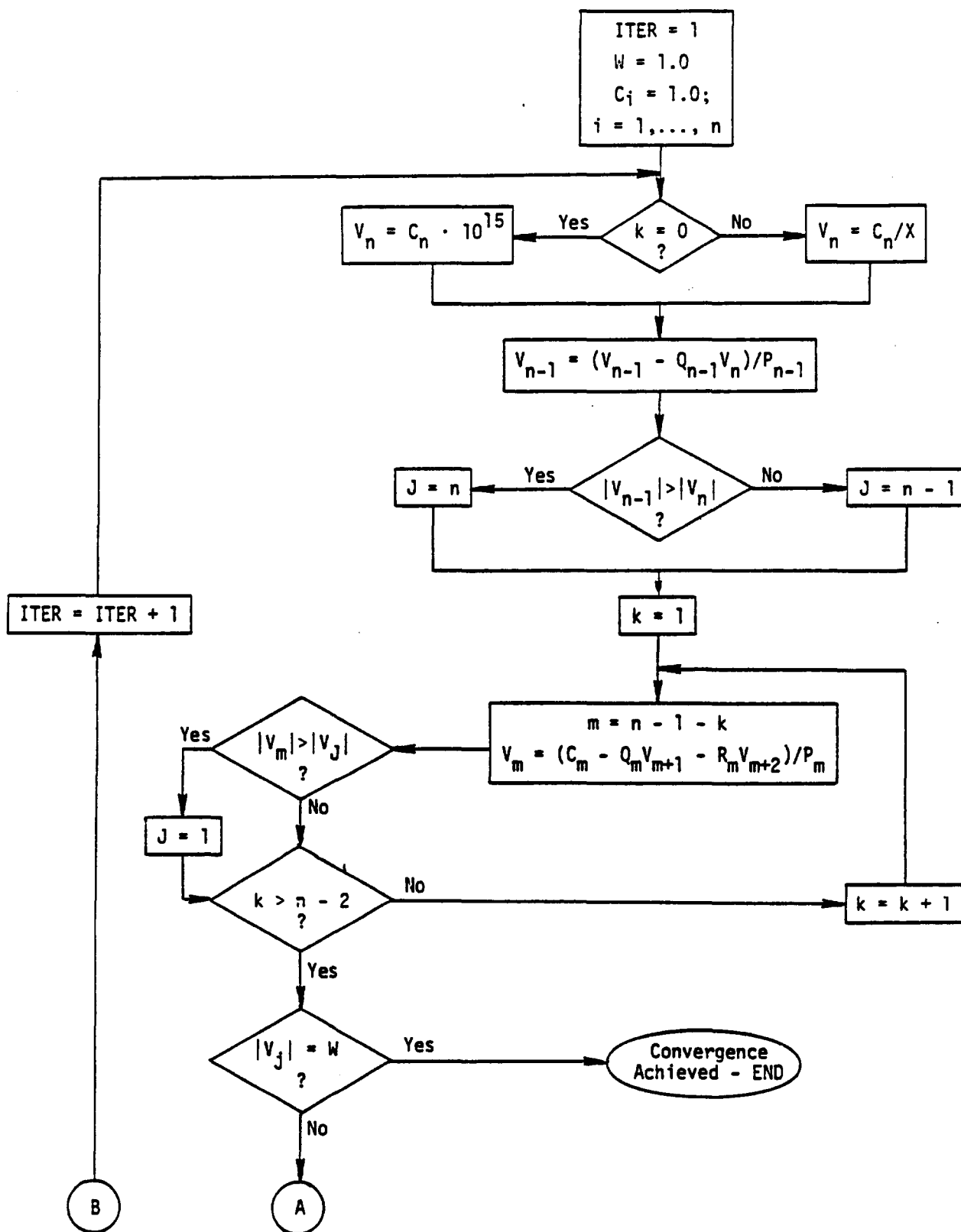


Figure 2. Eigenvalue solution algorithm equation solution.

GIVENS METHOD OF EIGENVALUE EXTRACTION

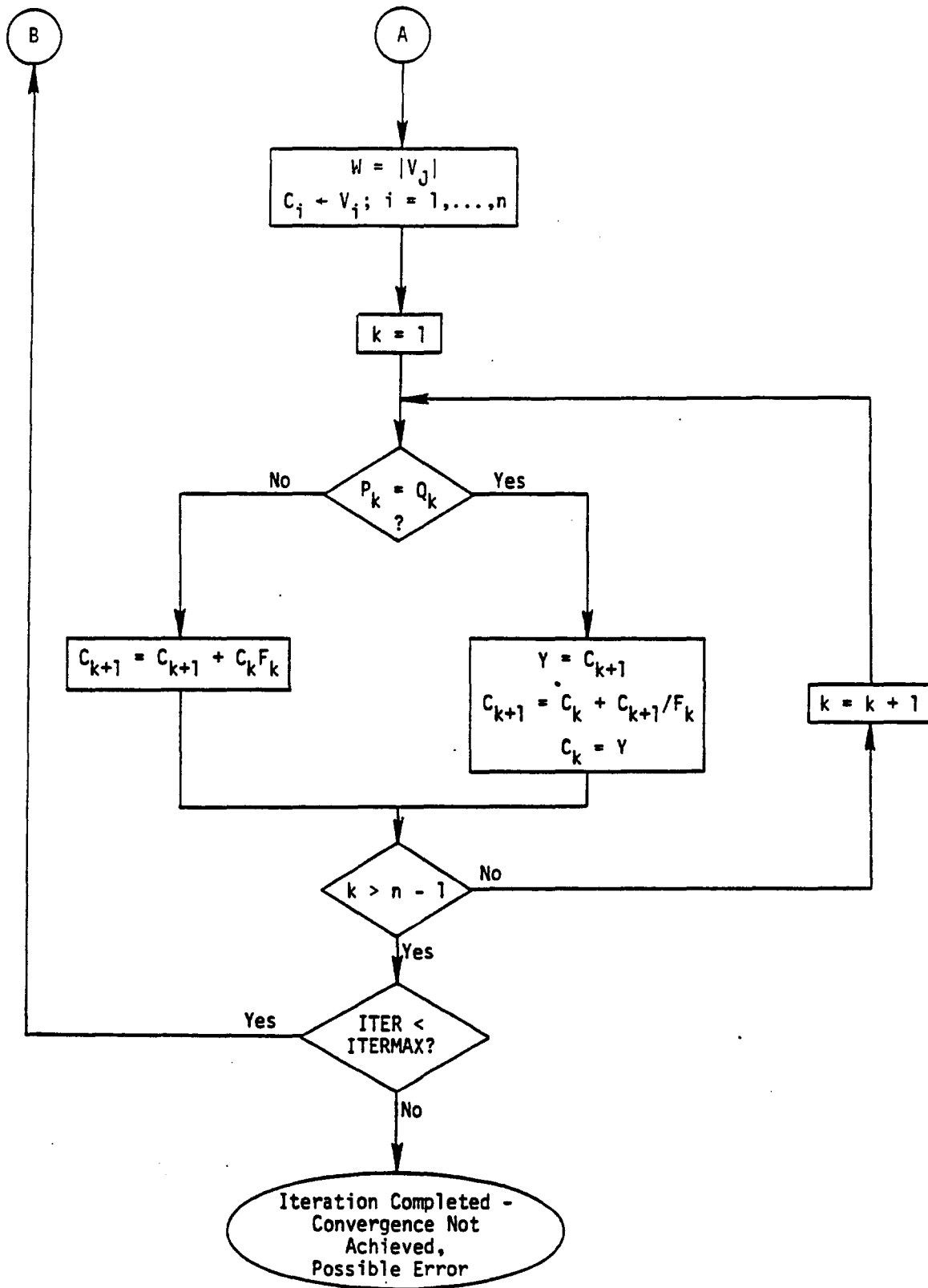


Figure 2. Eigenvalue solution algorithm equation solution (Cont'd).

EIGENVALUE EXTRACTION METHODS

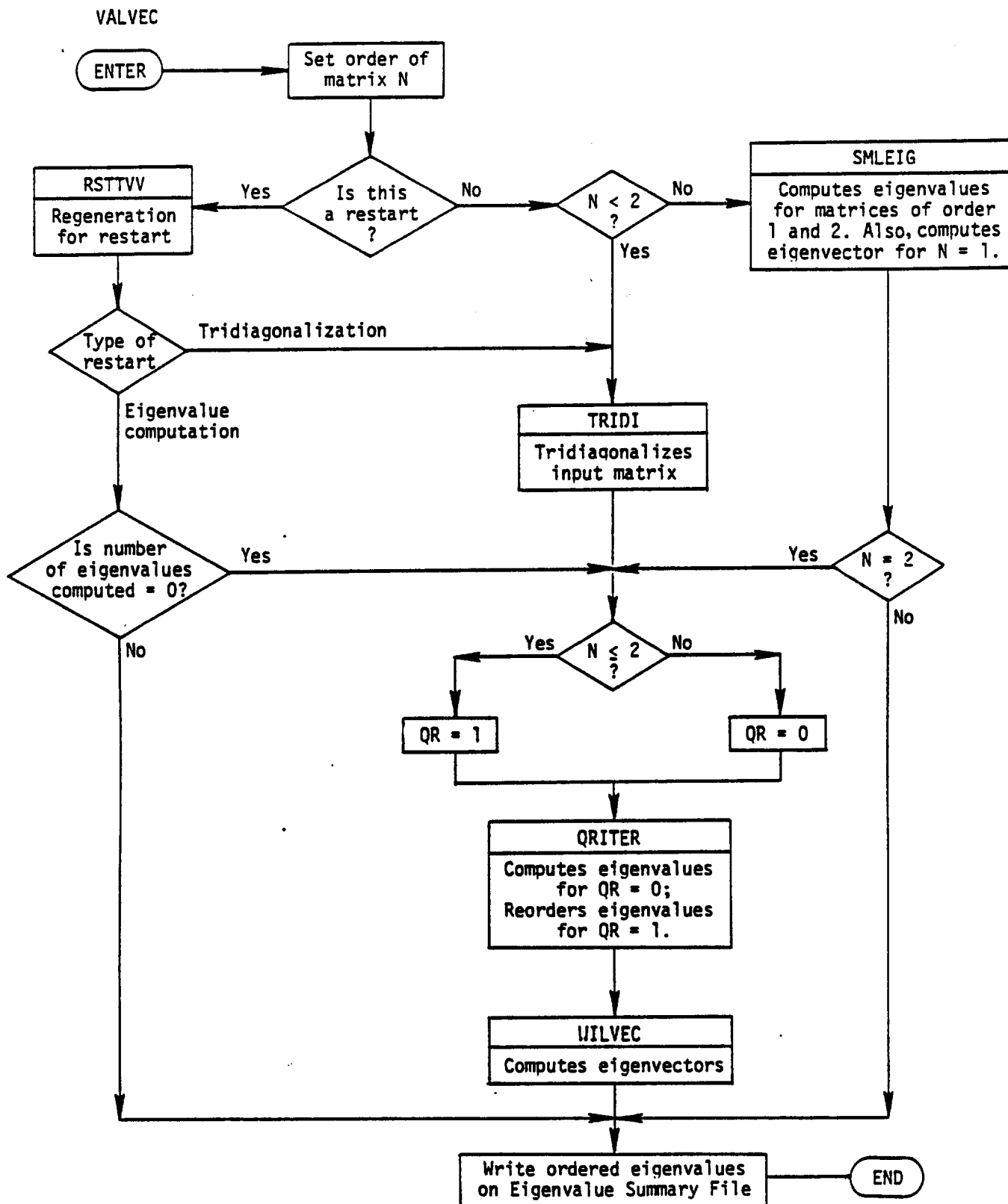


Figure 3. Subroutine VALVEC - module control.

GIVENS METHOD OF EIGENVALUE EXTRACTION

Arguments:

1. Location of Diagonals and Sines (Area D)
2. Location of Off Diagonals (Area \emptyset)
3. Location of Sines
4. Location of Available Storage

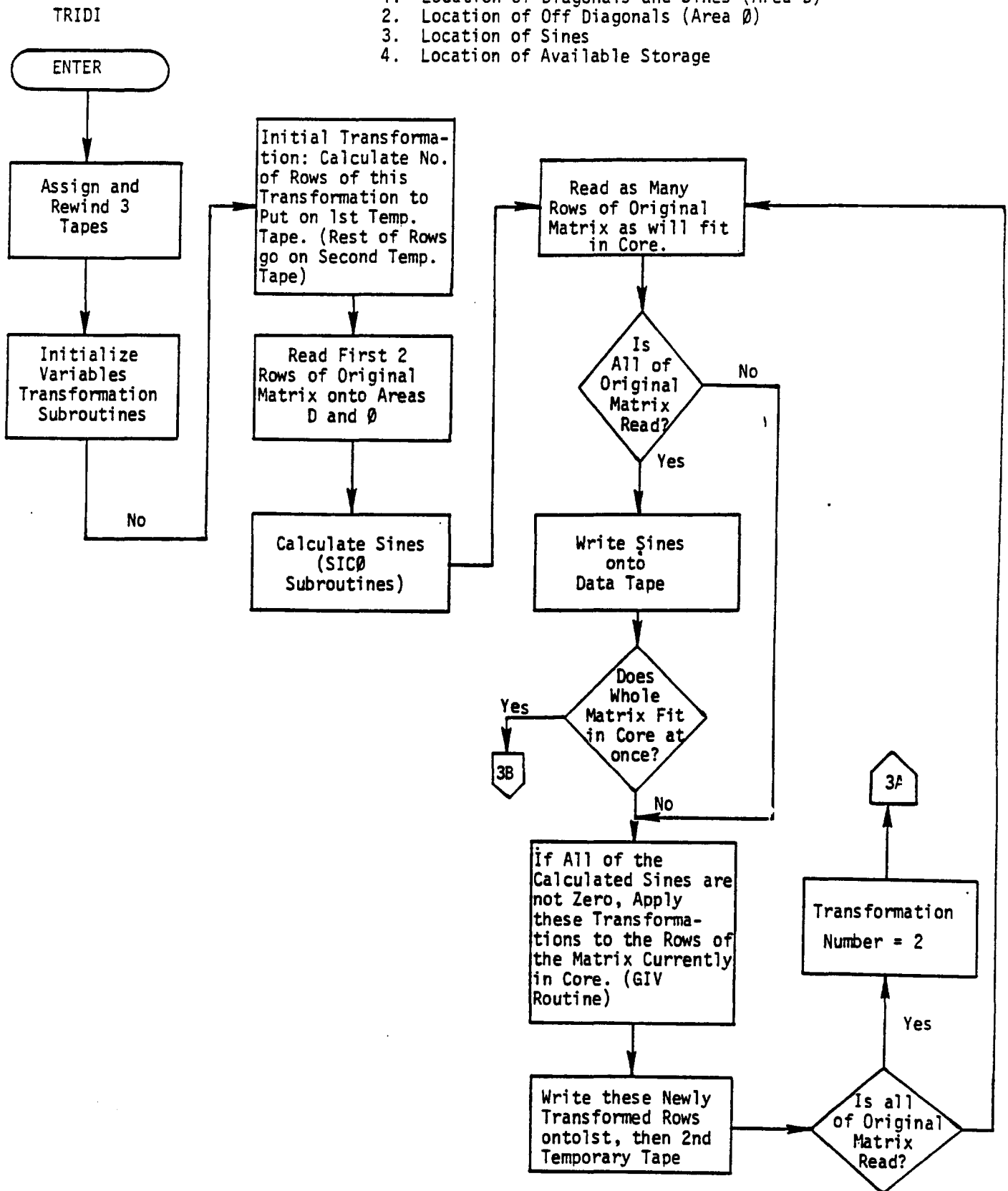


Figure 4. Tridiagonalization routine

EIGENVALUE EXTRACTION METHODS

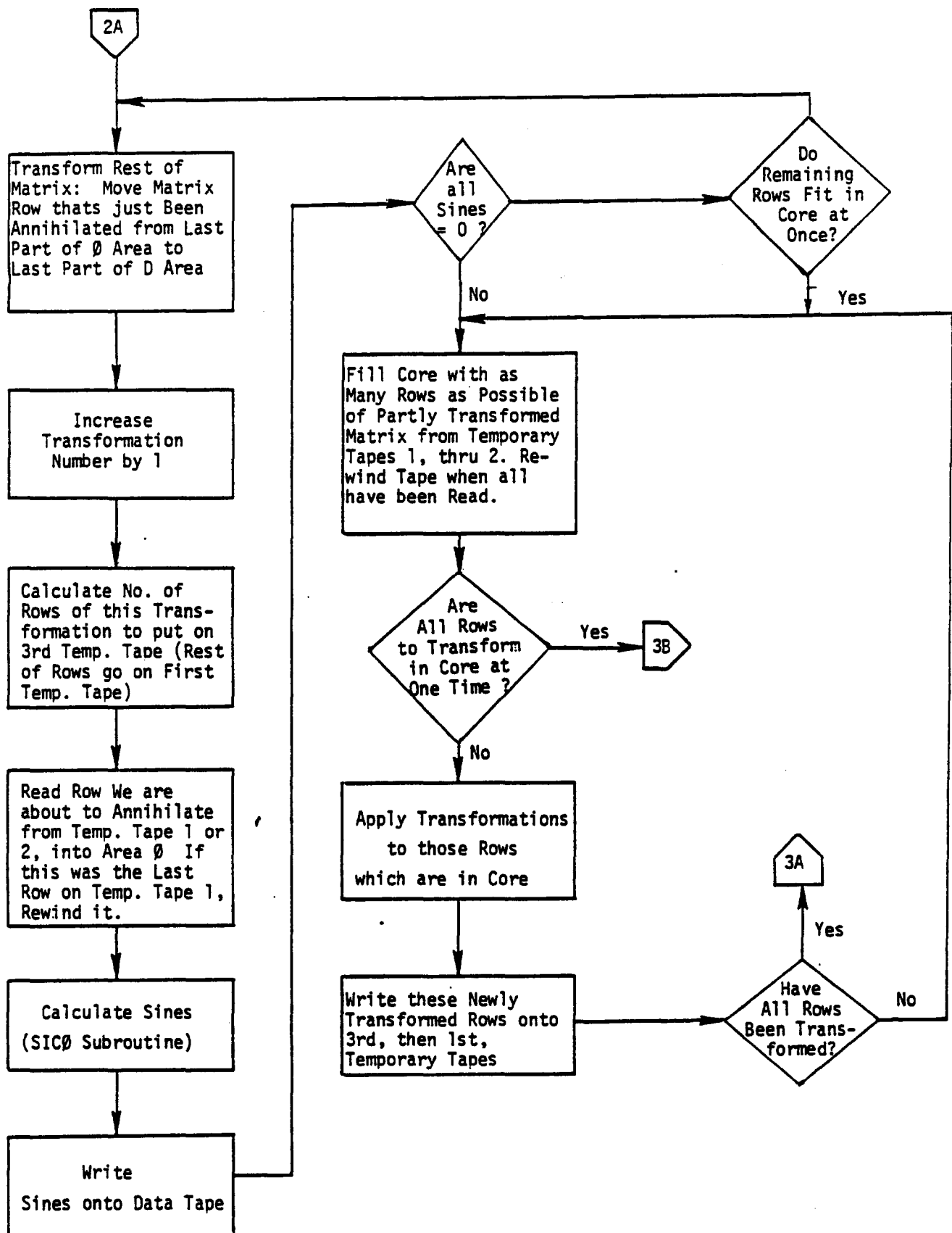


Figure 4. Tridiagonalization routine (continued)

GIVENS METHOD OF EIGENVALUE EXTRACTION

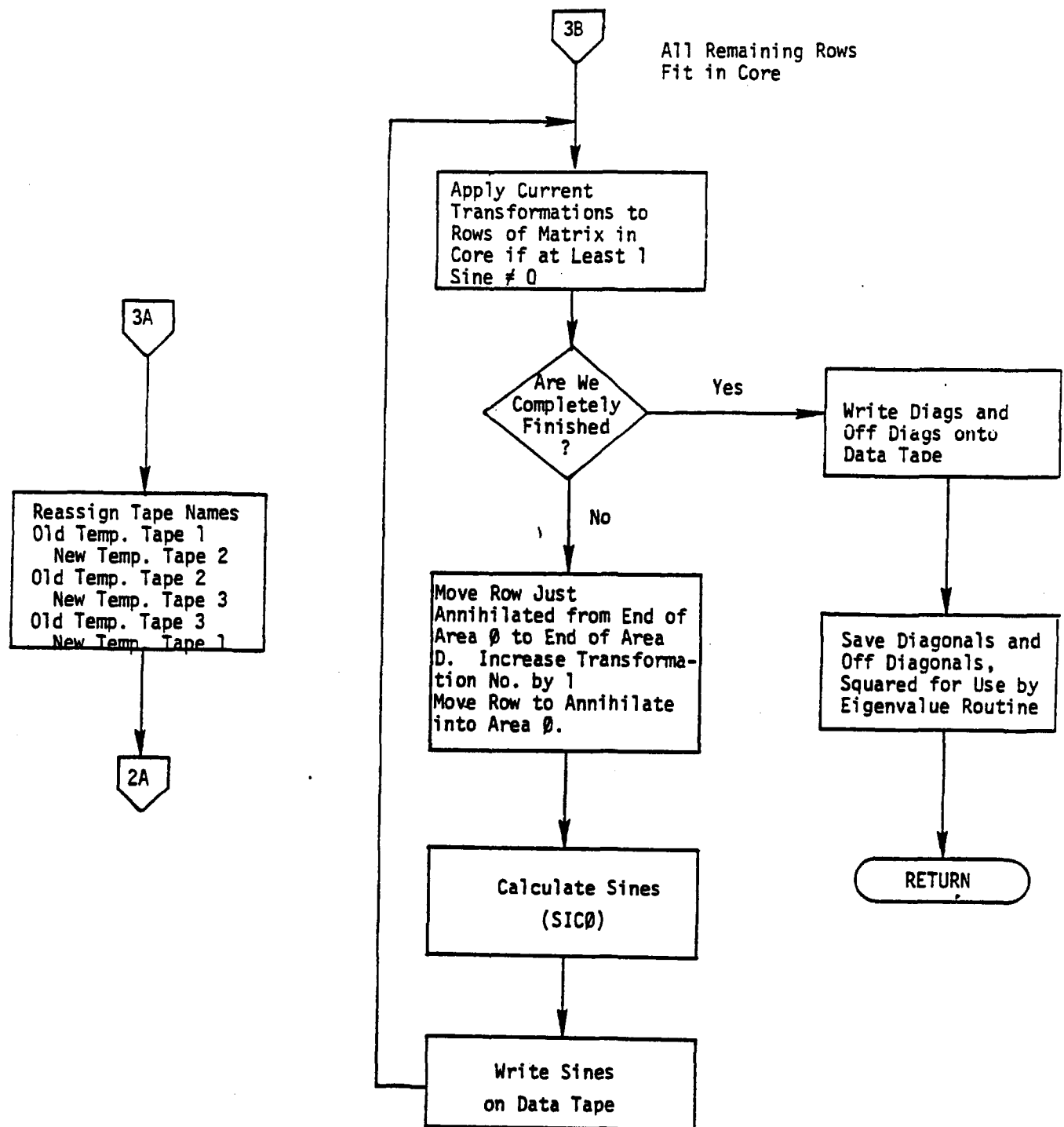


Figure 4. Tridiagonalization routine (continued)

EIGENVALUE EXTRACTION METHODS

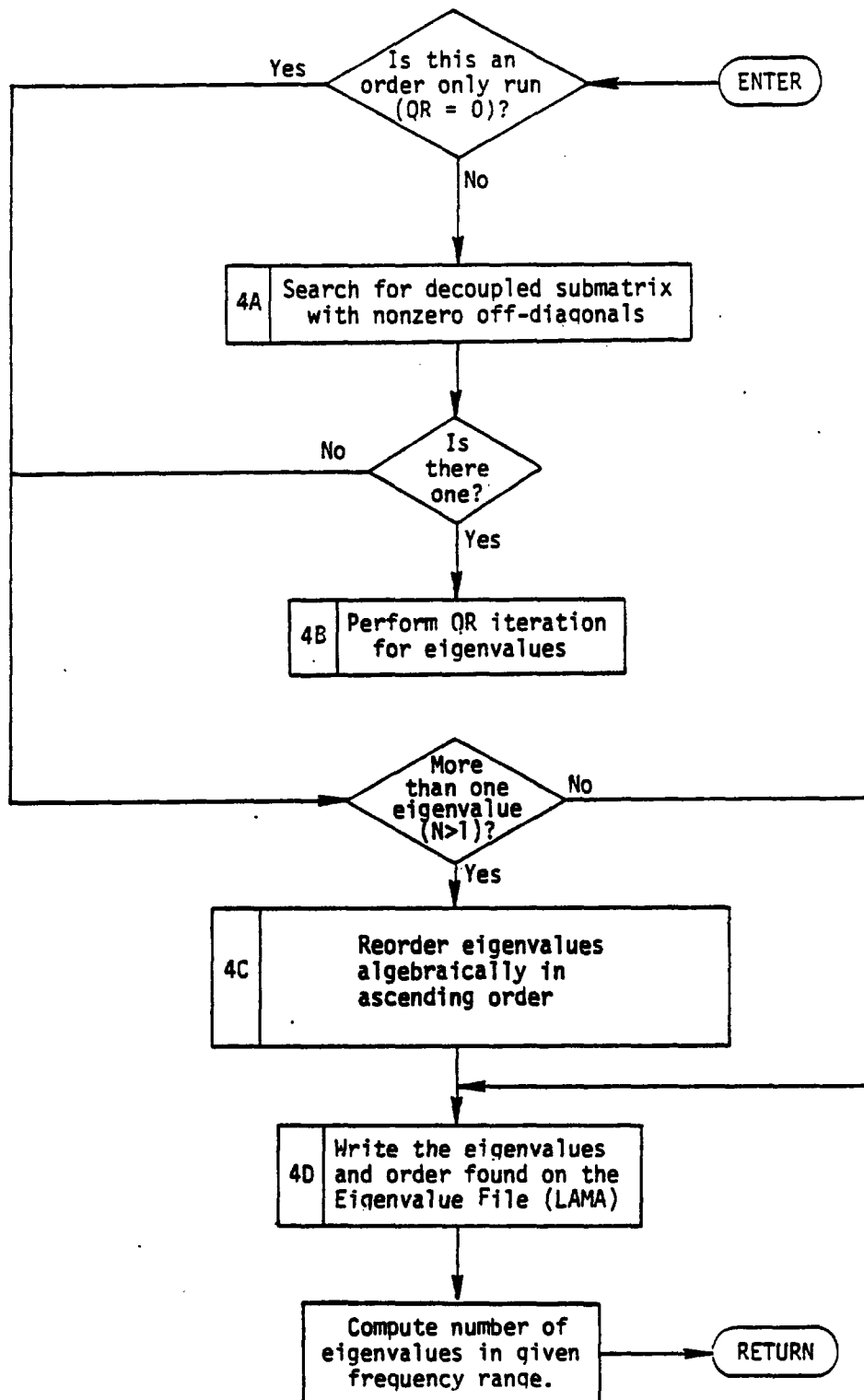


Figure 5. Gross logical flow of subroutine QRITER.

GIVENS METHOD OF EIGENVALUE EXTRACTION

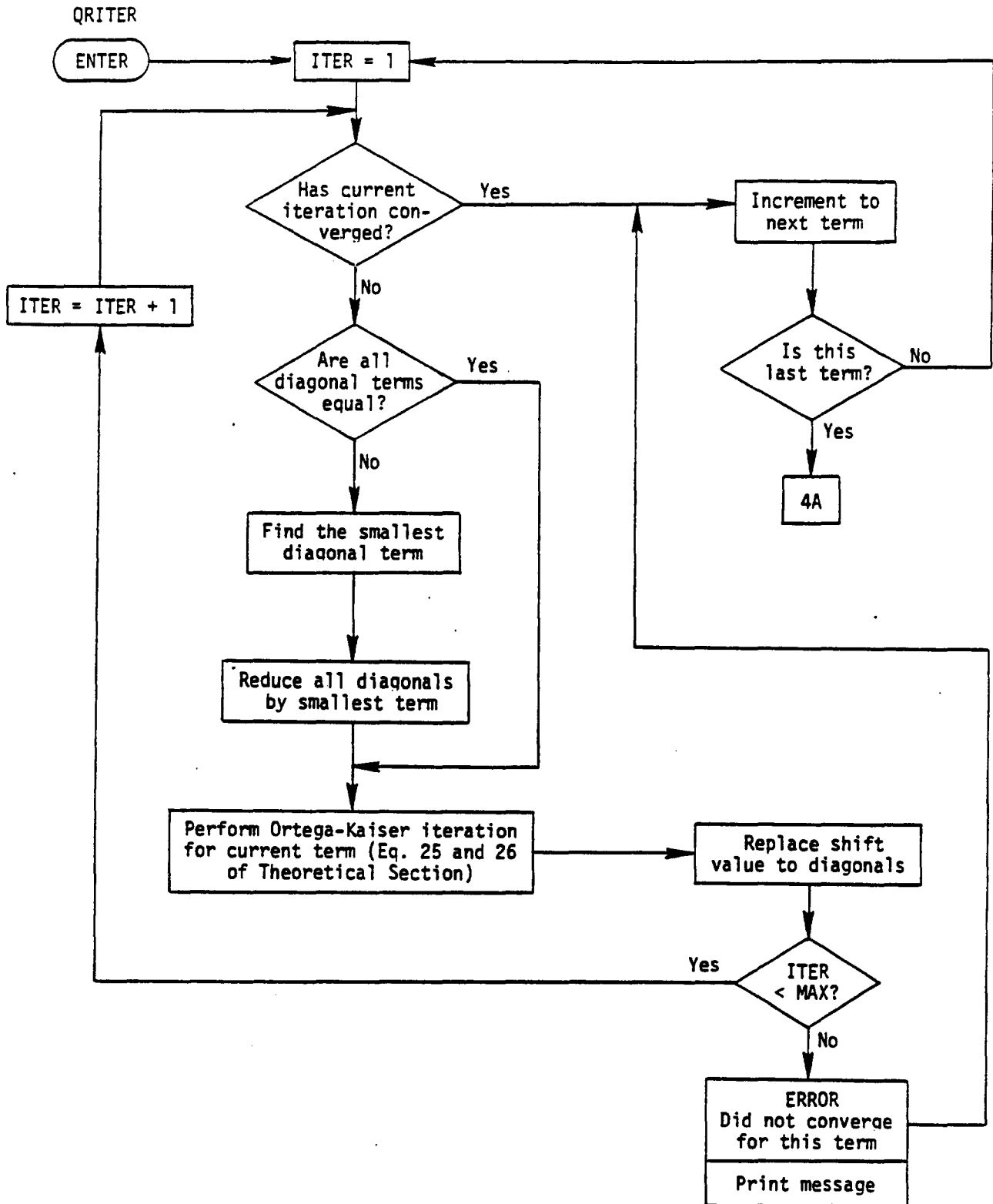


Figure 6. Subroutine QITER - eigenvalue iteration.

EIGENVALUE EXTRACTION METHODS

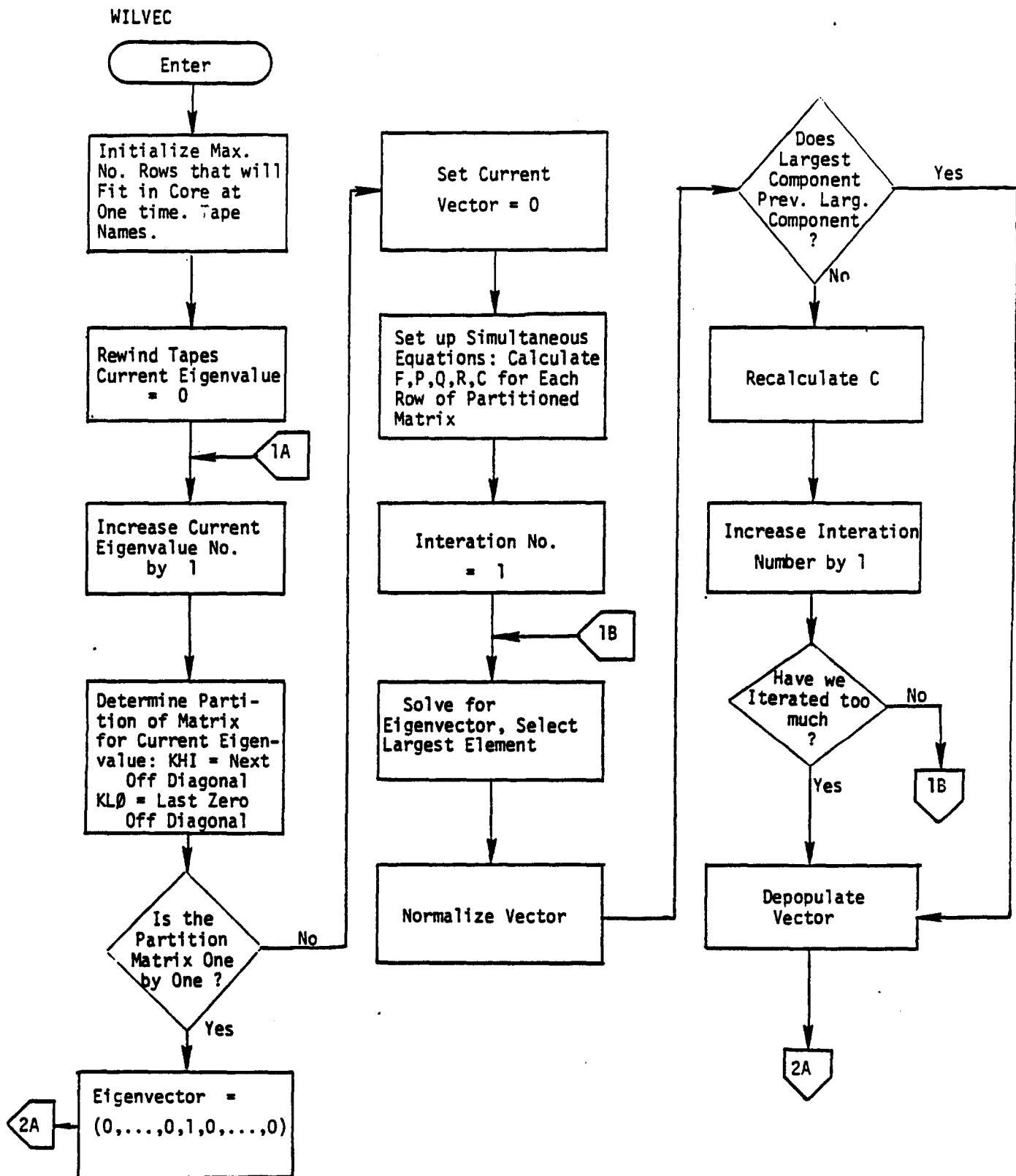


Figure 7. Eigenvector solution subroutine WILVEC

GIVENS METHOD OF EIGENVALUE EXTRACTION

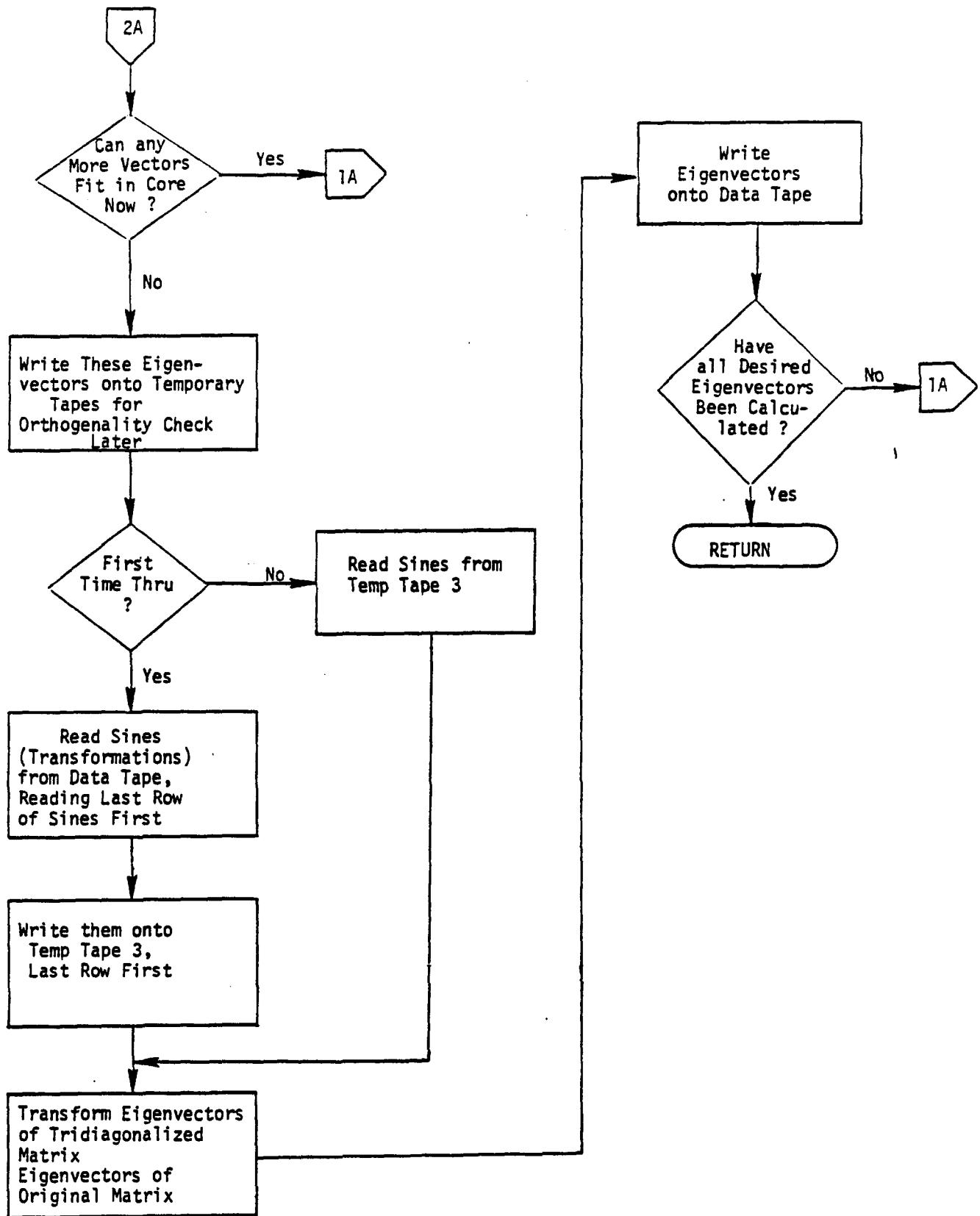


Figure 7. Eigenvector solution subroutine (continued)

REFERENCES

1. Givens, W., "Numerical Computation of the Characteristic Values of a Real Symmetric Matrix," Oak Ridge National Lab., ORNL-1574, 1954.
2. Lanczos, C., "An Iteration Method for the Solution of the Eigenvalue Problem of Linear Differential and Integral Operators," J. Res. Nat. Bur. Stand., Vol. 45, pp. 255-282, 1950.
3. Householder, A. S., and F. L. Bauer, "On Certain Methods for Expanding the Characteristic Polynomial," Numerische Mathematik, Vol. 1, pp. 29-37, 1959.
4. Wilkinson, J. H., "The Algebraic Eigenvalue Problem," Oxford University Press, 1965, pp. 506-510.
5. Francis, J. G. F., "The QR Transformation, a Unitary Analogue to the LR Transformation," Computer Journal, Vol. 4, No. 3 (Oct. 1961) and No. 4 (Jan. 1962).
6. Ortega, J. M. and H. F. Kaiser, "The LLT and QR Methods for Symmetric Tridiagonal Matrices," Computer Journal, Vol. 6, No. 1, (Jan. 1963), pp. 99-101.
7. Wilkinson, J. H., "The Calculation of the Eigenvectors of Codiagonal Matrices," The Computer Journal, Vol. I, 1958, pp. 90.
8. Bodewig, E., "Matrix Calculus," Interscience Publisher, 1959.

EIGENVALUE EXTRACTION METHODS

9.3 INVERSE POWER METHOD WITH SHIFTS

9.3.1 Summary of Procedures for Real Eigenvalue Analysis

The algorithm for finding the eigenvalues and eigenvectors of

$$([K] - \lambda[M])[\phi] = 0, \quad (1)$$

is given as follows.

The iteration algorithm is given by the following equations.

$$([K] - \lambda_0[M])\{W_n\} = [M]\{u_{n-1}\}, \quad (2)$$

$$\{\bar{u}_n\} = \frac{1}{C_n} \{W_n\}, \quad (3)$$

where C_n is the absolute value of the maximum component of $\{W_n\}$ and $\lambda = \lambda_0 + \lambda_1$. The value of $1/C_n$ converges to the shifted eigenvalue, λ_1 , nearest to the shift point λ_0 and $\{\bar{u}_n\}$ converges to the corresponding eigenvector. A triangular decomposition is required of the matrix $([K] - \lambda_0[M])$ in order to evaluate $\{W_n\}$ in Equation 2.

$$\{u_n\} = \{\bar{u}_n\} - \sum_i (\{\phi_i\}^T [M] \{\bar{u}_n\}) \{\phi_i\}, \quad (4)$$

The sum over i extends over all previously extracted eigenvectors.

For each iteration form

$$\{F_n\} = [M]\{u_n\}. \quad (5)$$

Compute the normalization factor

$$\alpha_n = (\{u_n\}^T \{F_n\})^{1/2}. \quad (6)$$

Compute the normalized difference of successive trial vectors

$$\{\delta u_n\} = \frac{\{u_n\}}{\alpha_n} - \frac{\{u_{n-1}\}}{\alpha_{n-1}}. \quad (7)$$

EIGENVALUE EXTRACTION METHODS

Compute

$$\{\delta F_n\} = [M]\{\delta u_n\} = \frac{\{F_n\}}{\alpha_n} - \frac{\{F_{n-1}\}}{\alpha_{n-1}} . \quad (8)$$

Compute the approximate eigenvalue

$$\lambda_1 = \frac{1}{C_n} \frac{\alpha_{n-1}}{\alpha_n} . \quad (9)$$

For the rapid convergence test,

$$\eta < A\epsilon \gamma \left| 1 + \frac{\lambda_0}{\lambda_1} \right| = A\epsilon \left| \frac{\lambda_2 - \lambda_1}{\lambda_1} \right| , \quad (10)$$

where A is a factor less than unity and ϵ is the mass-orthogonality criteria, compute

$$\eta = |\{\delta u_n\}^T \{\delta F_n\}|^{1/2} . \quad (11)$$

For normal convergence,

$$\sqrt{\delta_n} < A\epsilon , \quad (12)$$

Compute

$$\frac{\lambda_2}{\lambda_1} = \frac{\{\delta u_n\}^T \{\delta F_{n-1}\}}{\eta^2} , \quad (13)$$

and

$$\delta_n = (\{\delta u_n\}^T \{\delta F_n\}) / (1 - \frac{\lambda_2}{\lambda_1})^2 . \quad (14)$$

For the shift decision test,

$$h_{1,n} = \frac{\lambda_{1,n} - \lambda_{1,n-1}}{R_0} > \epsilon_1 . \quad (15)$$

INVERSE POWER METHOD OF EIGENVALUE EXTRACTION

Compute $h_{1,n}$ and the estimate of the required number of iterations, k , defined by the equation,

$$k = \frac{\log \left(\frac{\sqrt{\delta_n}}{A\epsilon} \right)}{\log (\lambda_2/\lambda_1)} . \quad (16)$$

For the λ_2 reliability test,

$$\epsilon_2 > |h_{2,n}| > |h_{2,n-1}| . \quad (17)$$

compute

$$h_{2,n} = \frac{\lambda_{2,n} - \lambda_{2,n-1}}{R_0} . \quad (18)$$

The above equations, along with the proper logic given in Figure 1 form the basis for the Inverse Power Method.

9.3.2 Summary of Procedures for Complex Eigenvalue Analysis

The procedures for complex eigenvalue analysis are very similar, and in many instances identical, to those for real eigenvalue analysis. The only major change in the flow diagrams for real eigenvalue analysis (Figures 1 and 2) is the calculation of the left eigenvector after passage of the convergence tests. Changes in procedure are indicated below according to the numbered blocks in Figures 4 and 5.

1. Compute Distribution of Shift Points

Regions as specified by the user are set up in the complex plane as shown in Figure 2. There may be any number of search regions and the search regions may overlap. Each region is divided into square subregions with sides of length ℓ_i . The number of subregions is,

$$\left\lceil \frac{|B_i - A_i|}{\ell_i} \right\rceil \leq N_r < \left\lfloor \frac{|B_i - A_i|}{\ell_i} \right\rfloor + 1 \quad (19)$$

The sum of the subregions, symmetrical with respect to the center of the original search region, redefines the search region as shown in Figure 3. The shift points, P_{s_i} , are at the center of each subregion.

2. Select a Starting Point

The shift point closest to the origin is used as the first starting point, λ_0 . Thereafter, the next closest shift point is used as λ_0 .

4. Compute One Eigenvalue and One Eigenvector

See Figures 5 and 6 for this block flow diagram.

4.1 Decompose Dynamic Matrix

The matrix to be decomposed is

$$[\lambda_0^2 M + \lambda_0 B + K] , \quad (20)$$

which is, in general, complex and nonsymmetric and where λ_0 is either a starting point, λ_s , a moved starting point, λ'_s , or a shifted shift point, λ'_0 . Double-precision arithmetic with partial pivoting is used.

4.2 One Vector Iteration

The iteration algorithm is, for $B \neq 0$,

$$[\lambda_0^2 M + \lambda_0 B + K]\{w_n\} = -[B + \lambda_0 M]\{u_{n-1}\} - [M]\{v_{n-1}\} , \quad (21)$$

$$\{\bar{u}_n\} = \frac{1}{c_n} \{w_n\} , \quad (22)$$

$$\{\bar{v}_n\} = \lambda_0 \{\bar{u}_n\} + \frac{1}{c_n} \{u_{n-1}\} , \quad (23)$$

$$\{u_n\} = \{\bar{u}_n\} - \sum_j \alpha_j \{\phi_j\} , \quad (24)$$

$$\{v_n\} = \{\bar{v}_n\} - \sum_j \alpha_j \lambda_j \{\phi_j\} , \quad (25)$$

where

$$\alpha_j = \frac{\{\bar{\phi}_j\}^T \{\lambda_j M \bar{u}_n + M \bar{v}_n + B \bar{u}_n\}}{\{\bar{\phi}_j\}^T [2\lambda_j M + B] \{\phi_j\}} \quad (26)$$

INVERSE POWER METHOD OF EIGENVALUE EXTRACTION

C_n = largest element (in magnitude) of $\{w_n\}$,

$\{\phi_j\}$ = previously found eigenvector ,

$\{\bar{\phi}_j\}$ = previously found left eigenvector ,

λ_j = previously found eigenvalue ,

λ_0 = shift point .

The sweeping operation, Equations 24 and 25, is also applied to starting vectors. The sum on (j) extends over all previously found eigenvalues.

For the special case, $[B] = 0$, Equation 25 is replaced by

$$\{v_n\} = \lambda_0 \{u_n\} + \frac{1}{C_n} \{u_{n-1}\} , \quad (27)$$

and Equation 26 is replaced by

$$\alpha_j = \frac{\{\bar{\phi}_j\}^T [M] \{\bar{u}_n\}}{\{\bar{\phi}_j\}^T [M] \{\phi_j\}} .$$

4.3 Convergence Tests (see Figure 6)

The convergence tests are identical to those for real eigenvalue analysis except that F_n is formed in the following manner

$$\{F_n\} = \left[M + \frac{1}{2\lambda_0 + \lambda_{1,n-1} + \lambda_{2,n-1}} B \right] \{u_n\} \quad (29)$$

where $\lambda_{1,n-1}$ and $\lambda_{2,n-1}$ are the eigenvalue estimates of $\lambda_{1,n}$ and $\lambda_{2,n}$ from the previous iteration.

4.8 Compute Left Eigenvector

Left eigenvectors satisfy the equation

$$[\lambda_j^2 M^T + \lambda_j B^T + K^T] \{\bar{\phi}_j\} = 0 , \quad (30)$$

where λ_j is identically equal to the eigenvalue for the given problem, i.e.,

$$[\lambda_j^2 M + \lambda_j B + K]\{\phi_j\} = 0. \quad (31)$$

The left eigenvector $\bar{\phi}_j$ is evaluated after the corresponding right eigenvector, ϕ_j , and the eigenvalue, λ_j , have been found. The procedure is to form the lower and upper triangular factors $[L_\ell]$ and $[U_\ell]$ of the dynamic matrix at the accepted value of the eigenvalue,

$$[L_\ell][U_\ell] = [M\lambda_j^2 + B\lambda_j + K], \quad (32)$$

and then to solve the equation

$$[M^T\lambda_j^2 + B^T\lambda_j + K]\{\bar{\phi}_j\} \equiv [U_\ell]^T[L_\ell]^T\{\bar{\phi}_j\} = \{F\}, \quad (33)$$

for $\{\bar{\phi}_j\}$ by forward and backward substitution (see Section 2.3). The excitation vector, $\{F\}$, may be specified arbitrarily since $[U_\ell]$ is nearly singular.

5. Have All the Eigenvalues Been Found?

The answer is "yes" if the number of eigenvalues found equals the total number estimated to be in the problem. The number found should include conjugate eigenvalues for problems with real M, B, and K matrices. The total number estimated to be in the problem is

$$N_t = 2N_{eq} - N_{oM}, \quad (34)$$

for problems in which $[B] \neq 0$, and

$$N_t = 2N_{eq} - 2N_{oM}, \quad (35)$$

for problems in which $[B] = 0$, and where

N_{eq} = number of rows in dynamic matrix

N_{oM} = number of columns of the mass matrix that are null.

6. Is Eigenvalue Outside Range? (see Figure 3)

The test is:

$$|\lambda_j - \lambda_s| > R_c : \text{Outside Range}, \quad (36)$$

INVERSE POWER METHOD OF EIGENVALUE EXTRACTION

where R_c is the distance from the starting point λ_s to the corners of the starting point's search region, see Figure 3.

7. Select Next-to-Last Trial Vector as Starting Vector

- a. If there was no shift in finding the previous eigenvalue

$$\bar{u}_{i+1,0} = u_{i,N-1} \quad (37)$$

$$\bar{v}_{i+1,0} = v_{i,N-1} \quad (38)$$

- b. If there were one or more shifts: Select $\bar{u}_{i+1,0}$ and $\bar{v}_{i+1,0}$ equal to the last vector before the first shift.

8. Too Close to Starting Point?

The criterion is

$$\epsilon \left| \frac{R_c}{\lambda_i - \lambda_s} \right| > \epsilon_3 : \text{Too close} \quad (39)$$

where

ϵ = criterion specified by user for convergence
of eigenvectors (Default value = 10^{-4})

λ_i = eigenvalue just found

λ_s = starting point

R_c = range.

The value stored in the program for ϵ_3 is .05.

EIGENVALUE EXTRACTION METHODS

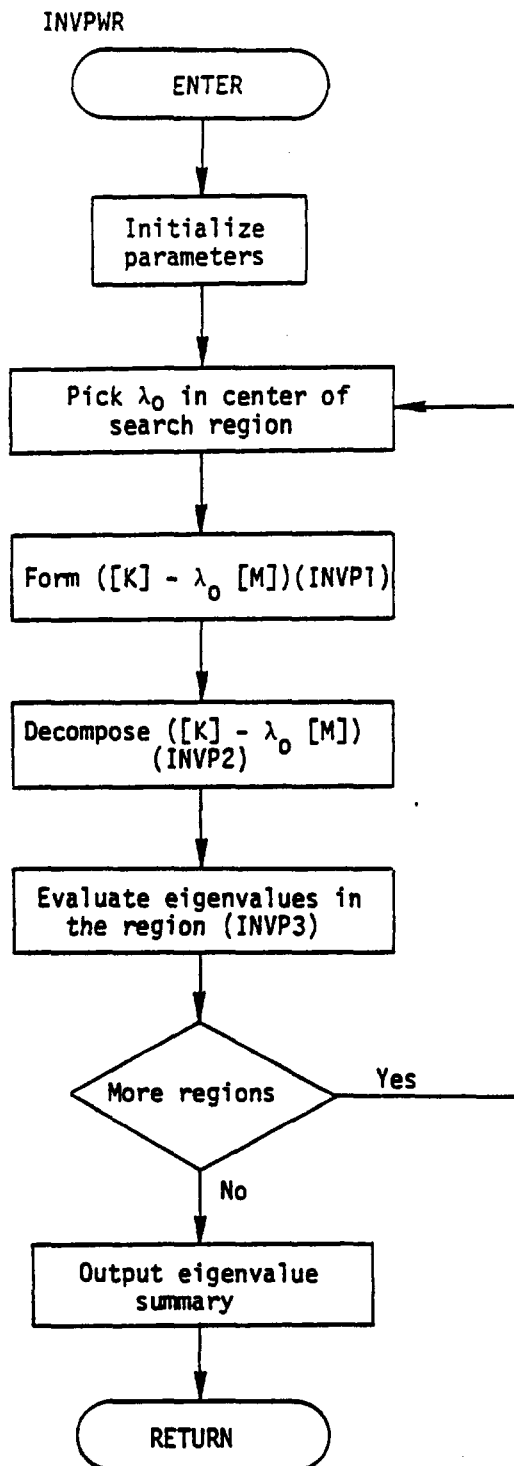


Figure 1. Flowchart for Real Inverse Power Method with Shifts

INVERSE POWER METHOD OF EIGENVALUE EXTRACTION

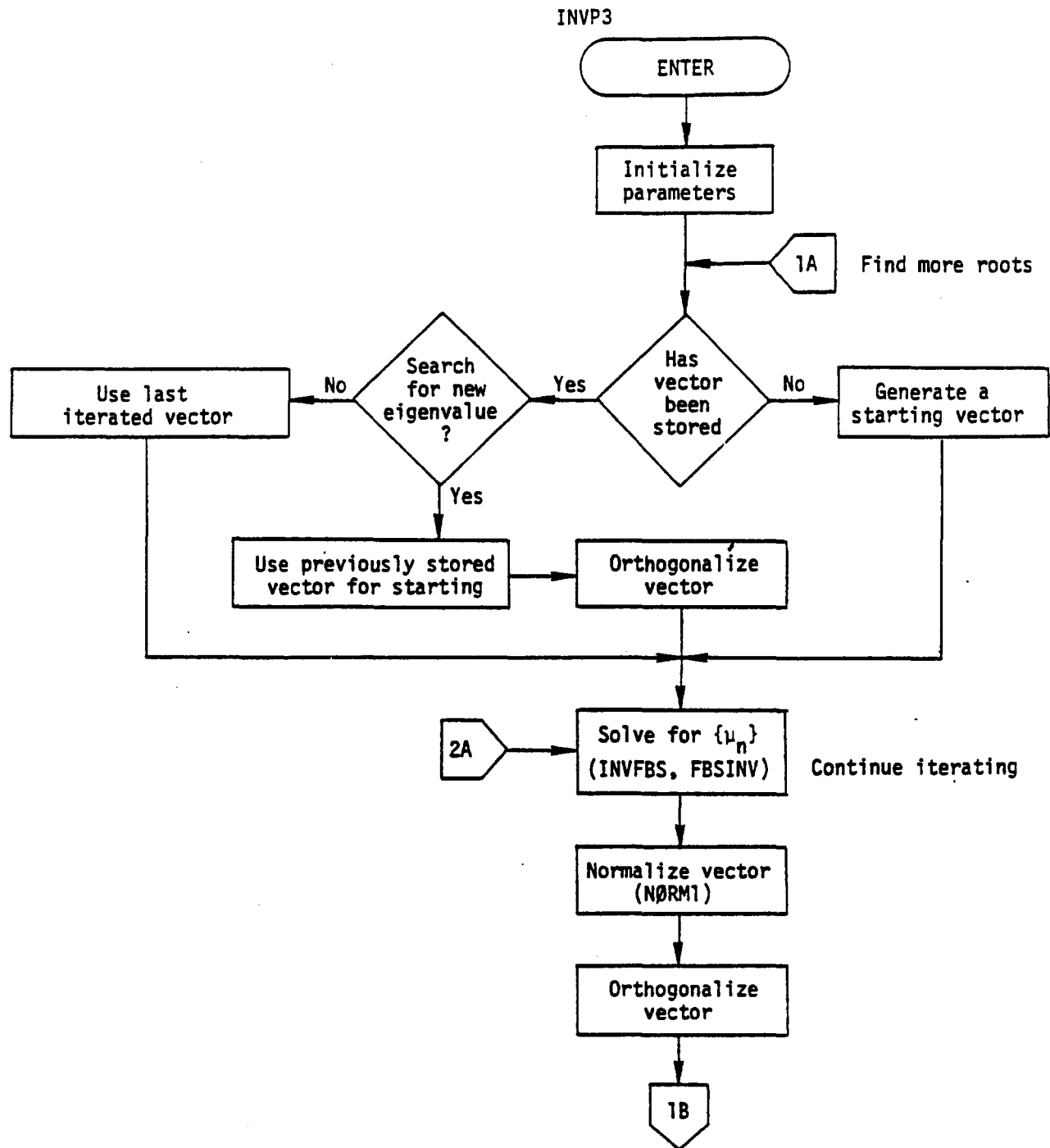


Figure 2.(a) Flowchart for Real Inverse Power Eigenvalue Evaluation

EIGENVALUE EXTRACTION METHODS

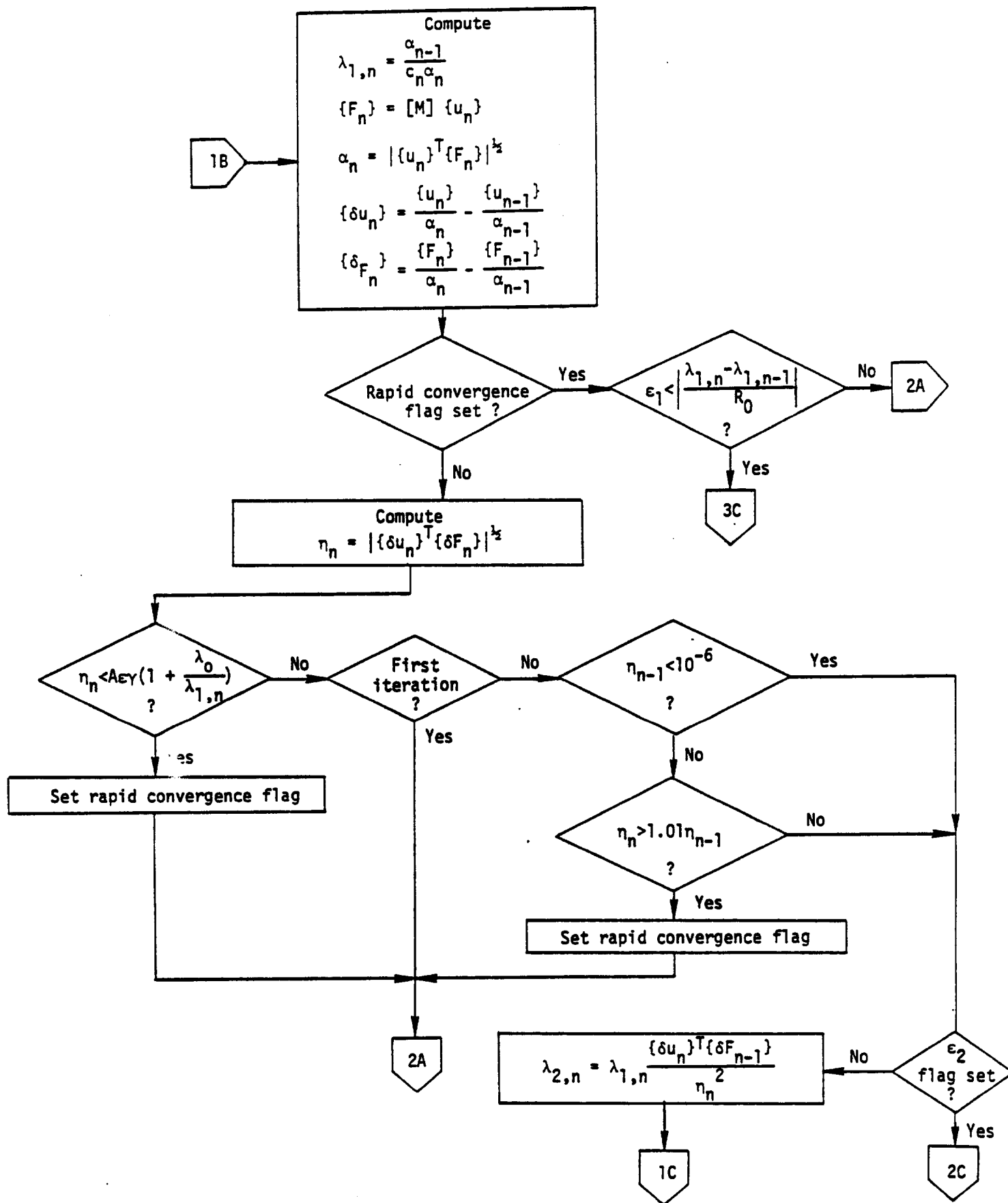


Figure 2.(b) Flowchart for Real Inverse Power Eigenvalue Evaluation

INVERSE POWER METHOD OF EIGENVALUE EXTRACTION

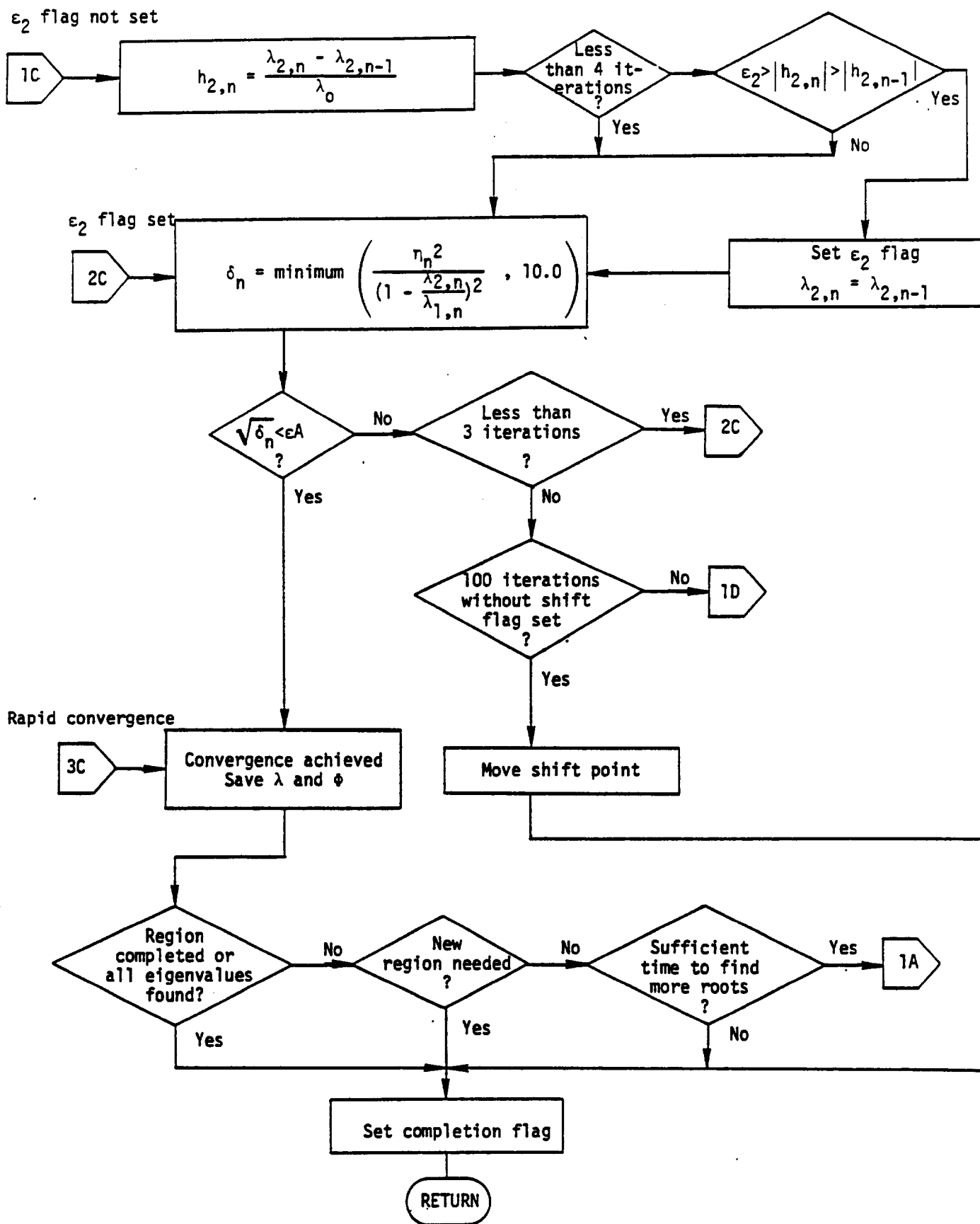


Figure 2.(c) Flowchart for Real Inverse Power Eigenvalue Evaluation

EIGENVALUE EXTRACTION METHODS

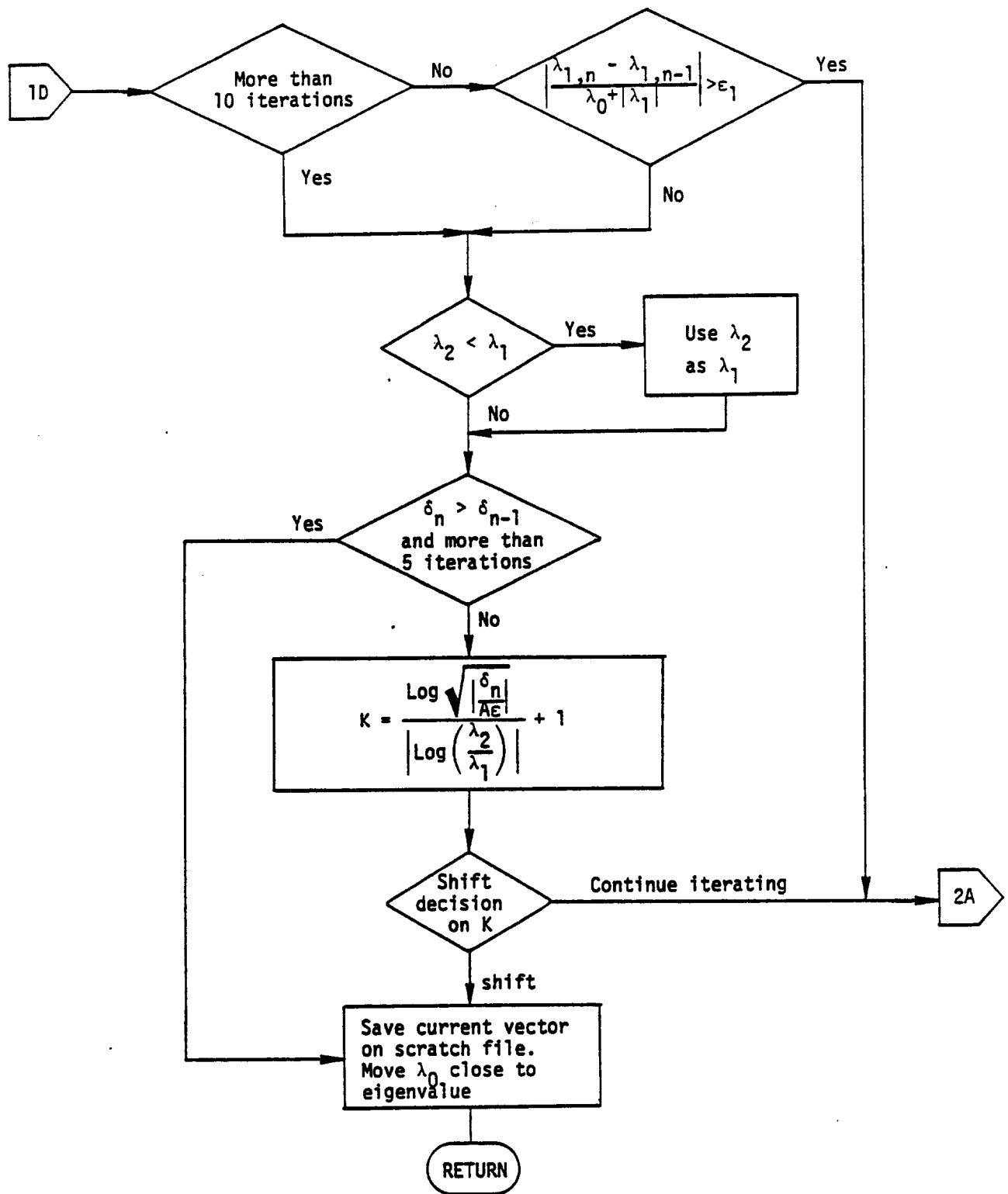


Figure 2.(d) Flowchart for Real Inverse Power Eigenvalue Evaluation

INVERSE POWER METHOD OF EIGENVALUE EXTRACTION

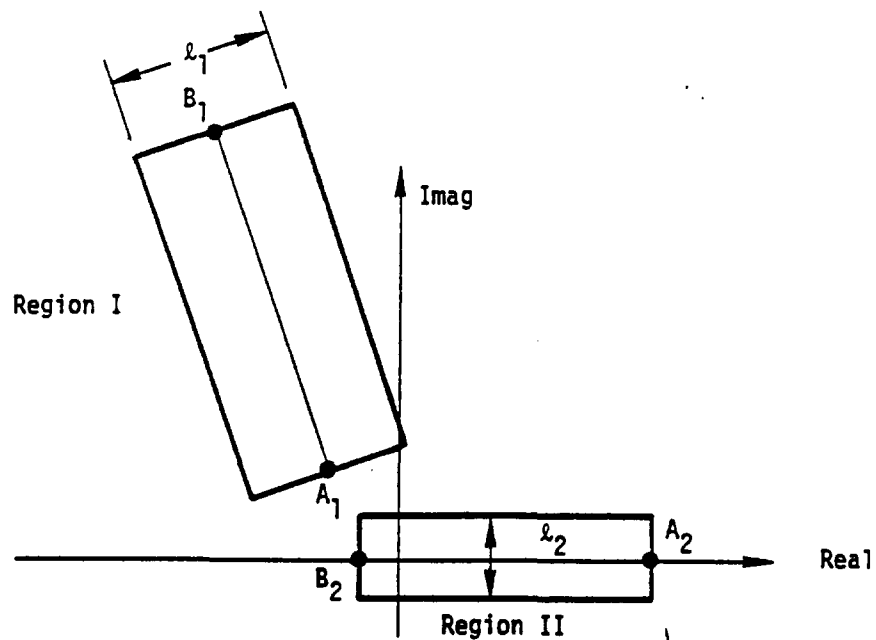


Figure 2. Search regions

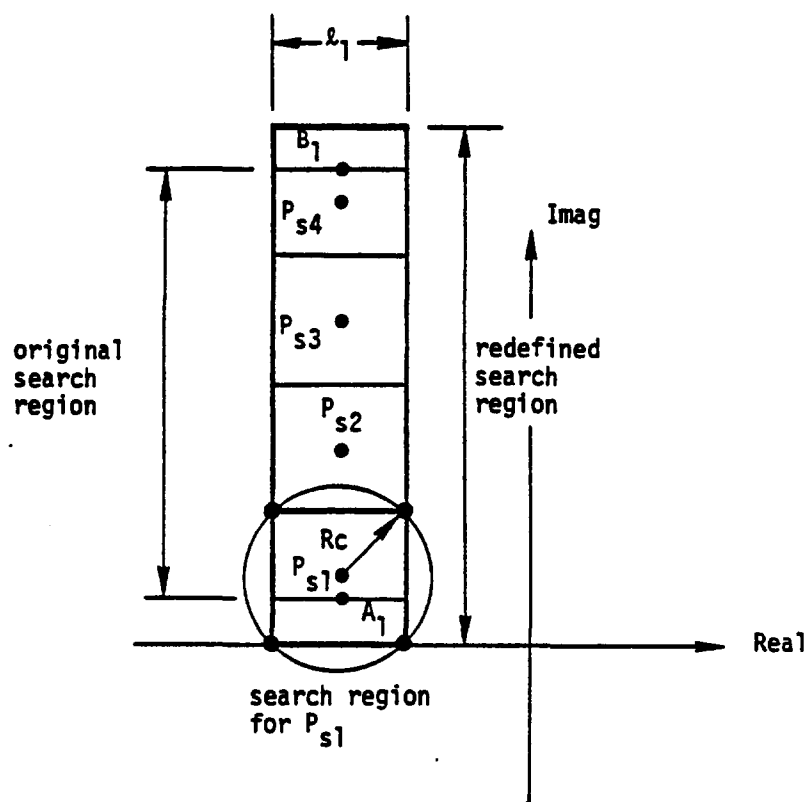


Figure 3. Distribution of starting points within a search region.

EIGENVALUE EXTRACTION METHODS

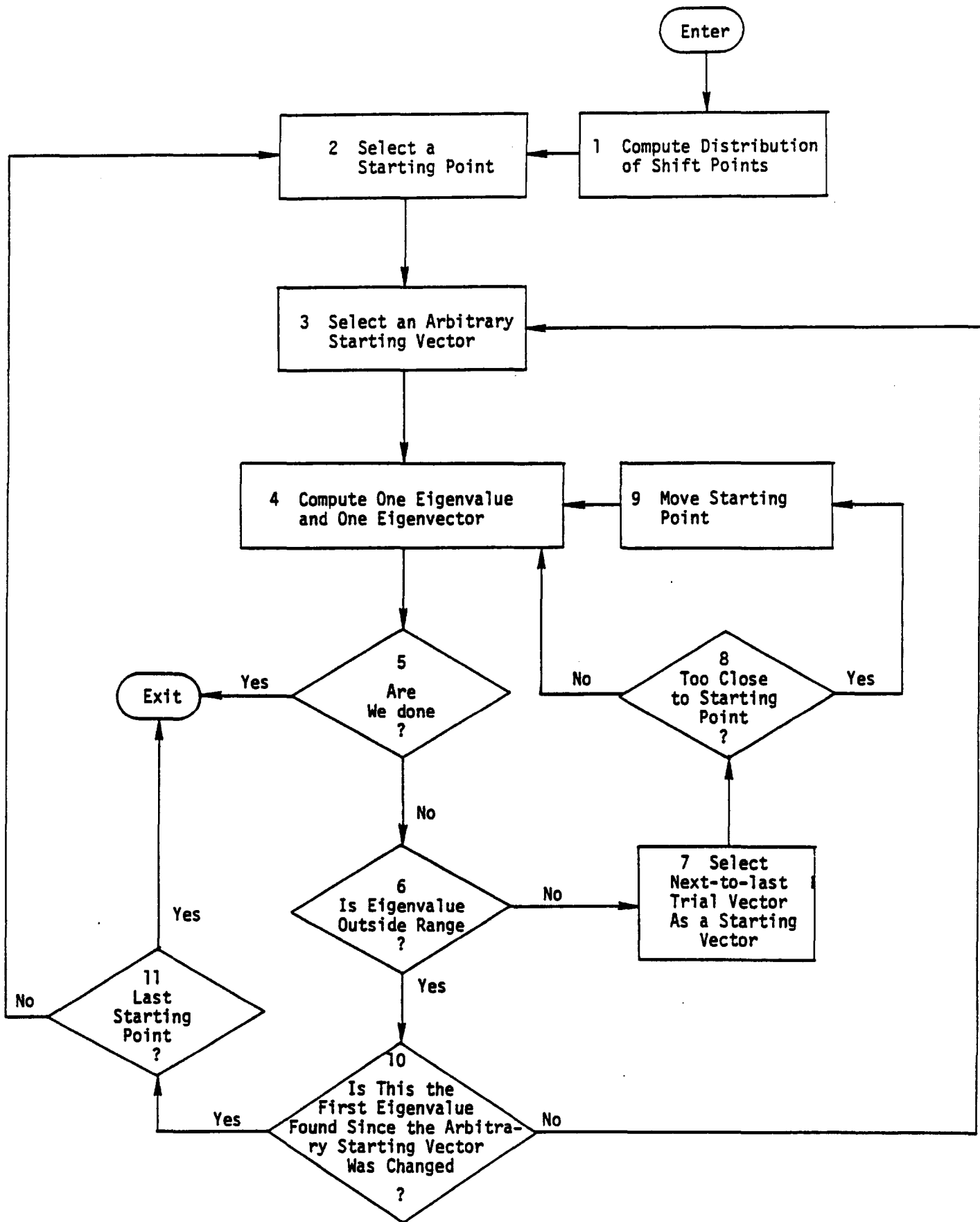


Figure 4. Flowchart for Complex Inverse Power Method with Shifts

INVERSE POWER METHOD OF EIGENVALUE EXTRACTION

CINVP3

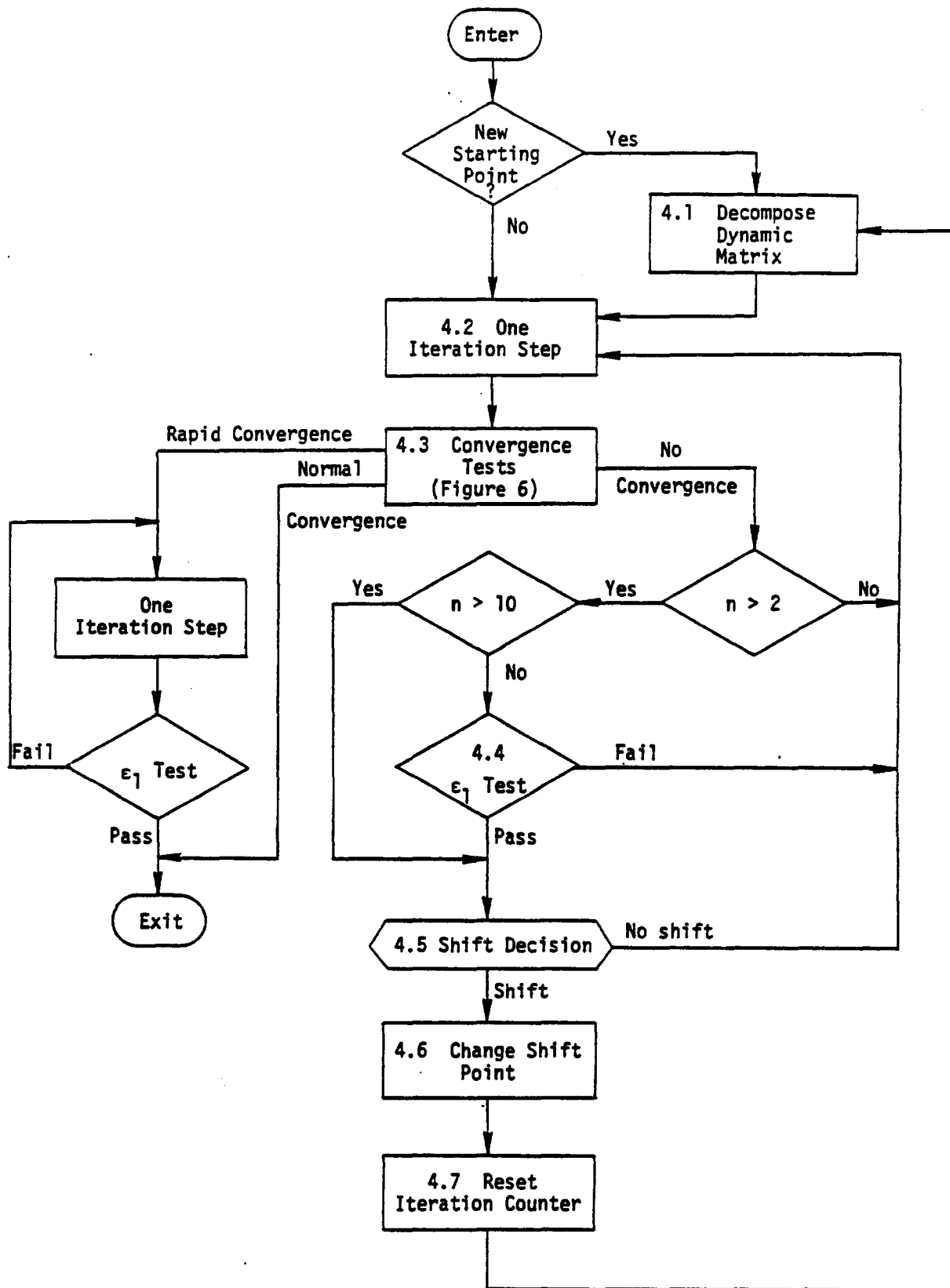


Figure 5. Flow diagram for block 4, compute one eigenvalue and one eigenvector

EIGENVALUE EXTRACTION METHODS

Convergence Tests

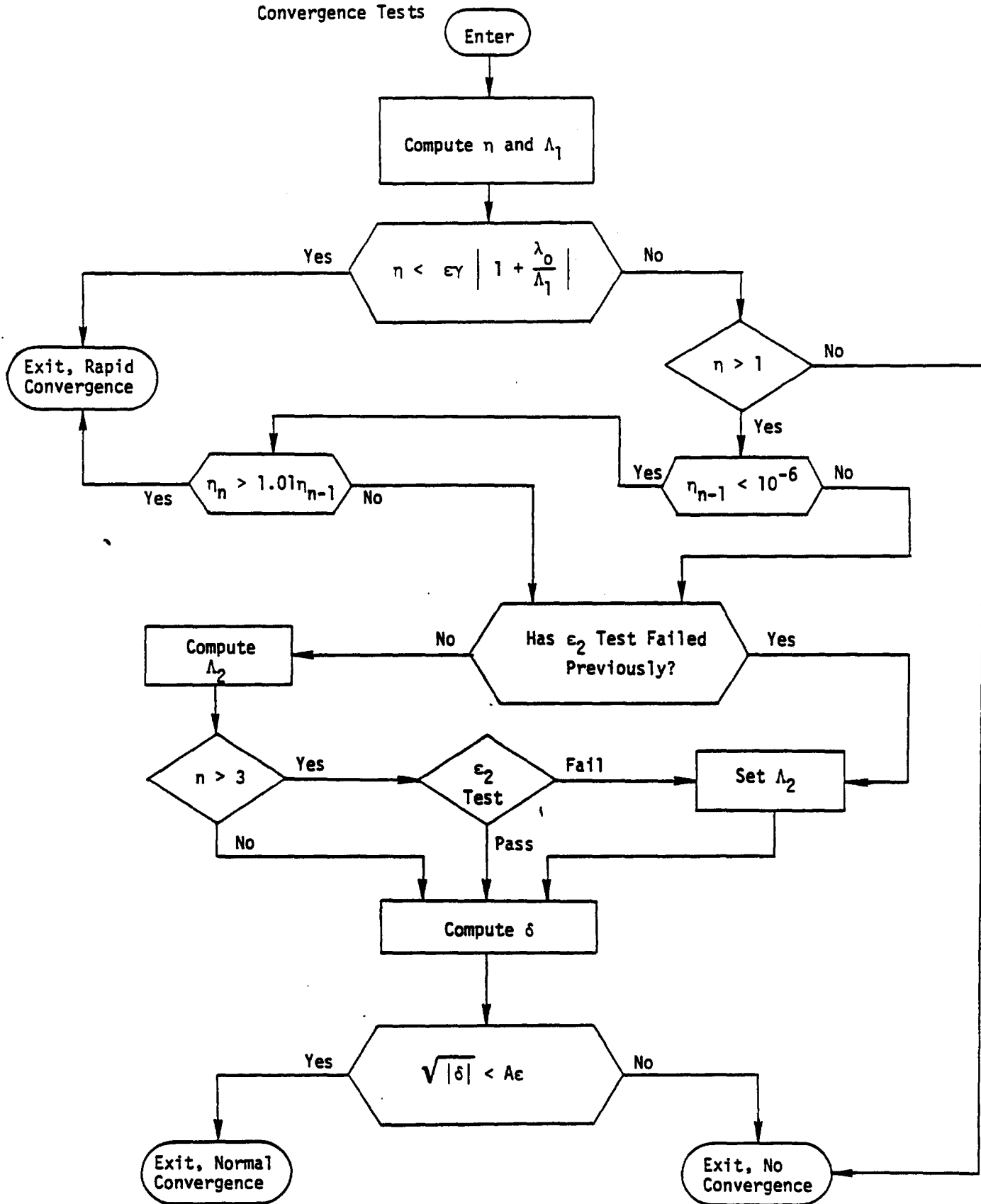


Figure 6. Flow diagram for block 4.3, convergence tests

EIGENVALUE EXTRACTION METHODS

9.4 THE TRIDIAGONAL REDUCTION (FEER) METHOD

9.4.1 Background for Real Eigenvalue Analysis

The Tridiagonal Reduction or FEER Method is an automatic matrix reduction scheme whereby the eigensolutions in the neighborhood of a specified point in the eigenspectrum can be accurately extracted from a tridiagonal eigenvalue problem whose order is much lower than that of the full problem. Specifically the order, m , of the reduced problem is never greater than

$$m = 2\bar{q} + 10, \quad (1)$$

where \bar{q} is the desired number of accurately computed eigenvalues.

As shown in Section 9.4.3, the Tridiagonal Reduction Method employs only a single initial shift of eigenvalues and hence usually requires only one matrix decomposition. It consequently tends to be much more efficient than the Inverse Power Method when more than one or two eigensolutions are required.

The restrictions on the use of the method for real eigenvalue analysis in NASTRAN are as follows:

1. For structural vibration mode applications, the method extracts a preselected number of eigenvalues which are closest to a specified shift value, λ_0 , rather than computing the eigenvalues in a prescribed range.
2. In buckling problems, a preselected number of eigenvalues of smallest magnitude are obtained, i.e., no shifting is performed. Physically, this implies that the buckling load parameters, whether positive or negative, are computed in order of increasing magnitude.

The real eigenproblem formulations associated with the Tridiagonal Reduction Method are presented in Section 9.4.2 and a summary of the computational procedures, including flow charts, is given in Section 9.4.3. Refer to the Theoretical Manual, Section 10.6, for details concerning the development of the relevant equations and numerical criteria.

EIGENVALUE EXTRACTION METHODS

9.4.2 Problem Formulation

The problem is to find a specified number of real eigenvalues and corresponding eigenvectors for

$$[K - \lambda_a M]\{\phi\} = 0. \quad (1)$$

It is further required that these eigensolutions constitute the set lying closest to a specified point, λ_0 , in the eigenspectrum.

The definitions of the eigenvalue, λ_a , the matrices $[K]$ and $[M]$, and their mathematical properties, depend on the type of problem being solved within the NASTRAN environment. For real analysis, only two separate problem types need be considered; structural vibration and buckling problems. The matrix definitions and mathematical distinctions for these two cases are summarized in the following table:

Table 1. Problem Formulations

Problem Type	Quantity	Definition	NASTRAN Notation	Most General Properties
Structural Vibration Modes	$[K]$	Stiffness Matrix - analysis set	$[K_{aa}]$	Symmetric, non-negative, semidefinite matrix
	$[M]$	Mass Matrix - analysis set	$[M_{aa}]$	Same
	λ_a	Square of a circular natural frequency	ω^2	Positive
Buckling	$[K]$	Stiffness Matrix - analysis set	$[K_{aa}]$	Symmetric, positive-definite matrix
	$[M]$	Differential Stiffness Matrix - analysis set	$[K_{aa}^d]$	Symmetric, indefinite matrix
	λ_a	Buckling Load Parameter	$-\lambda$	Positive or negative

The essential mathematical differences between the two types of problems center around the properties of the $[M]$ matrix, which is non-negative for vibration mode problems, but indefinite

THE TRIDIAGONAL REDUCTION (FEER) METHOD

for buckling problems, thereby permitting the existence of both positive and negative eigenvalues in the latter case. In addition, the stiffness matrix may be singular for vibration problems while it is always positive definite in buckling applications, which implies that the buckling analysis is performed on a kinematically stable structure.

In summary, the two problems under consideration are of the forms

$$[K_{aa} - \omega^2 M_{aa}]\{\phi\} = 0 \quad (\text{structural vibrations}) \quad (2)$$

and

$$[K_{aa} + \lambda K_{aa}^d]\{\phi\} = 0. \quad (\text{buckling}) \quad (3)$$

Further, if the user requests vibration modes in the neighborhood of a specified frequency, ω_0 , Equation 2 can be written as

$$[\bar{K}]\{\phi\} = \lambda' [M_{aa}]\{\phi\}, \quad (4)$$

where

$$[\bar{K}] = [K_{aa} - \omega_0^2 M_{aa}], \quad (5)$$

and

$$\lambda' = \omega^2 - \omega_0^2. \quad (6)$$

The resulting effective stiffness matrix, $[\bar{K}]$, is indefinite in this case, since it possesses both positive and negative eigenvalues. This requires that a non-square root decomposition scheme be used in subsequent operations. However, $\omega_0 = 0$ is taken as a default value, or it may be specified by the user. In this case, a specified number of natural frequencies starting with the lowest will be computed. In order to utilize a more efficient Cholesky decomposition of $[\bar{K}]$ under these conditions, a small negative shift $\lambda_0 = -\alpha^2$ is used, yielding

$$[\bar{K}] = [K_{aa} + \alpha^2 M_{aa}], \quad (7)$$

and

$$\lambda' = \omega^2 + \alpha^2. \quad (8)$$

It is easy to prove that the resulting effective stiffness matrix $[\bar{K}]$ is positive-definite provided that the system masses generate positive kinetic energy due to any kinematically admiss-

EIGENVALUE EXTRACTION METHODS

ible rigid body motions of the structure. This requirement is always satisfied by the mass matrix in a physically well posed problem, thereby allowing a Cholesky square-root decomposition to be performed when the roots are computed in the neighborhood of zero. Since no shifting is performed in buckling problems, the effective stiffness matrix is $[\bar{K}] = [K_{aa}]$, which is always positive-definite, again permitting the use of a Cholesky decomposition.

In any event, a decomposition or factoring of $[\bar{K}]$ is next performed:

$$[\bar{K}] = [L][\bar{d}][L]^T, \quad (\text{shifted vibration mode problems}) \quad (9)$$

or

$$[\bar{K}] = [C][C]^T, \quad (\text{buckling problems or vibration modes in the neighborhood of zero desired})$$

where $[L]$ and $[C]$ are lower triangular factors and $[\bar{d}]$ is a diagonal matrix.

To facilitate computation of eigenvalues closest to the point of interest within the eigen-spectrum, inverse forms of the eigenvalue problems are employed, as in the Inverse Power Method with Shifts.

The general form of the inverse problem may be written as

$$[B]\{X\} = \lambda[D]\{X\}, \quad (11)$$

where the above terms are defined as follows:

Table 2. Inverse Eigenproblem Definitions

Problem Type	$[B]$	$[D]$	$\{X\}$	λ
1. Shifted Vibration Modes	$[M_{aa}][L]^{-1})^T[\bar{d}]^{-1}[L]^{-1}[M_{aa}]$	$[M_{aa}]$	$\{\phi\}$	$\frac{1}{\omega^2 - \omega_0^2}$
2. Unshifted Vibration Modes (in the neighborhood of zero frequency)	$[C]^{-1}[M_{aa}][C]^{-1})^T$	$[I]$ (Identity Matrix)	$[C]^T\{\phi\}$	$\frac{1}{\omega^2 + \alpha^2}$
3. Buckling Modes	$[C]^{-1}[K_{aa}^d][C]^{-1})^T$	$[I]$	$[C]^T\{\phi\}$	$-\frac{1}{\lambda}$

THE TRIDIAGONAL REDUCTION (FEER) METHOD

9.4.3 Summary of Computational Procedures and Flow Charts - Real Eigenvalue Analysis

Flow diagrams illustrating the computational procedures are shown in Figures 1 and 2. The details of each block are summarized below.

1. Calculate Small Negative Shift Parameters, α^2

In the case of unshifted vibration mode problems a negative shift parameter for removing possible singularities is found from:

$$\alpha^2 = \max(\alpha_{\min}^2, \alpha_0^2), \quad (1)$$

where

$$\alpha_{\min}^2 = n(10^{2-t}) \left| \frac{k_{ii}}{m_{ii}} \right|_{\max}; \quad m_{ii} \neq 0, \quad (2)$$

and

$$\alpha_0^2 = 10^{-t/3} \left| \frac{k_{ii}}{m_{ii}} \right|_{\min}; \quad m_{ii} \neq 0. \quad (3)$$

k_{ii} and m_{ii} are the diagonal elements of $[K_{aa}]$ and $[M_{aa}]$ respectively, n is the number of $\{u_a\}$ degrees of freedom, and t is the number of floating decimal digits carried by the computer.

2. Zero-Out Excessively Small Elements of $[M]$ Matrix (see Table 1, Section 9.4.2)

- a. Compare the magnitudes of all off-diagonal elements of $[M]$ with the corresponding diagonal elements to determine whether

$$\left| \frac{m_{ij}}{m_{ii}} \right| \leq 10^{-2t/3}; \quad i \neq j, \quad m_{ii} \neq 0. \quad (4)$$

- b. Set $m_{ij} = 0.0$ for every off-diagonal element satisfying the above criterion.

3. Establish Tentative Reduced Problem Size

- a. Count the number, \bar{n} , of non-null columns or rows in the above modified $[M]$ matrix and set

$$\bar{r} = \bar{n} - f \quad (5)$$

where f is the number of previously computed eigensolutions.

EIGENVALUE EXTRACTION METHODS

- b. Calculate a tentative size, m , of the reduced eigenproblem from

$$m = \min[2\bar{q} + 10, \bar{r}], \quad (6)$$

where

$$\bar{q} = q - f, \quad (7)$$

and q is the total number of accurate eigenvalues requested by the user, including previously computed modes.

If $\bar{q} > \bar{r}$, the program will try to find all existing solutions.

4. Construct Factors of $[\bar{K}]$ Matrix (see Section 9.4.2)

- a. Set

$$(i) \quad [\bar{K}] = [K_{aa} - \omega_o^2 M_{aa}], \quad \text{(Shifted Vibration Mode Problems)} \quad (8)$$

or

$$(ii) \quad [\bar{K}] = [K_{aa} + \alpha^2 M_{aa}], \quad \text{(Unshifted Vibration Mode Problems)} \quad (9)$$

or

$$(iii) \quad [\bar{K}] = [K_{aa}]. \quad \text{(Buckling Problems)} \quad (10)$$

- b. Perform a non-square root decomposition:

$$[\bar{K}] = [L][\bar{d}][L]^T \quad (11)$$

for case (i), or a Cholesky symmetric decomposition:

$$[\bar{K}] = [C][C]^T \quad (12)$$

for cases (ii) and (iii), using real arithmetic without pivoting. Save the triangular and diagonal factors. If the decomposition for case (i) fails or the decompositions for cases (ii) and (iii) fail after two increases in α^2 by factors of one hundred, then the program is aborted because of unremovable stiffness matrix singularities.

THE TRIDIAGONAL REDUCTION (FEER) METHOD

5. Execute Tridiagonal Reduction Algorithm (see flow diagram for this block, Figure 2)

5.1 Initialize the Recurrence Algorithm

Initialize the vector index to $i = 0$ and set

$$\{v_0\} = \{0\}, \quad (13)$$

where $\{v_0\}$ is an $(n \times 1)$ null vector.

5.2 Generate a Starting or Restart Vector and Set $d_{i+1} = 0.0$

a. Construct an n -element vector $\{w\}$ using a pseudo-random number generator.

b. Solve for an un-normalized trial vector from the equation

$$\{\bar{v}_{i+1}\} = [\bar{B}]\{w\}, \quad (14)$$

where

$$[\bar{B}] = ([L]^{-1})^T [d]^{-1} [L]^{-1} [M_{aa}], \quad (\text{case i}) \quad (15)$$

or

$$[\bar{B}] = [C]^{-1} [M_{aa}] ([C]^{-1})^T, \quad (\text{case ii}) \quad (16)$$

or

$$[B] = [C]^{-1} [K_{aa}^d] ([C]^{-1})^T. \quad (\text{case iii}) \quad (17)$$

Forward and backward passes are used to perform the above inverse operations.

c. Normalize the above vector:

$$\{v_{i+1}^{(0)}\} = \left[\frac{1}{\{\bar{v}_{i+1}\}^T [D] \{\bar{v}_{i+1}\}} \right]^{1/2} \{\bar{v}_{i+1}\}, \quad (18)$$

where

$$[D] = [M_{aa}], \quad (\text{case i}) \quad (19)$$

$$[D] = [I]. \quad (\text{cases ii and iii}) \quad (20)$$

EIGENVALUE EXTRACTION METHODS

d. Set $d_{i+1} = 0.0$ and proceed to block 5.5.

5.3 Create One Approximate Trial Vector and One Diagonal Coefficient

The recurrence algorithm is:

$$a_{i,i} = \{v_i\}^T [B] \{v_i\}, \quad (21)$$

$$\{\bar{v}_{i+1}\} = [\bar{B}] \{v_i\} - a_{i,i} \{v_i\} - d_i \{v_{i-1}\}, \quad (22)$$

$$\bar{d}_{i+1} = [\{\bar{v}_{i+1}\}^T [D] \{\bar{v}_{i+1}\}]^{1/2}, \quad (23)$$

$$\{v_{i+1}^{(0)}\} = \frac{1}{\bar{d}_{i+1}} \{\bar{v}_{i+1}\}, \quad (24)$$

where,

$$[B] = [D][\bar{B}], \quad (25)$$

and $\{\bar{v}_{i+1}^{(0)}\}$ is an approximation to the new trial vector.

5.4 First Normalization Test

The test is

$$|\bar{d}_{i+1}| \geq 10^{2-t} |a_{i,i}|. \quad (26)$$

Pass: Proceed directly to block 5.5.

Fail: Return to block 5.2, generate a new restart vector for $\{v_{i+1}^{(0)}\}$, and then proceed to block 5.5.

5.5 Iterate to Obtain Orthogonalized Vector

Designate $\{\tilde{x}_j\}$, $j = 1, f$ as previously calculated and stored eigenvectors. Perform the iterations,

$$\begin{aligned} \{v_{i+1}^{(s+1)}\} &= \{v_{i+1}^{(s)}\} - \sum_{j=1}^i [\{v_j\}^T [D] \{v_{i+1}^{(s)}\}] \{v_j\} \\ &\quad - \sum_{j=1}^f [\{\tilde{x}_j\}^T [D] \{v_{i+1}^{(s)}\}] \{\tilde{x}_j\}; \quad s = 0, 1, 2, \dots, \end{aligned} \quad (27)$$

until

THE TRIDIAGONAL REDUCTION (FEER) METHOD

$$\max_{1 \leq j \leq i} |\{v_j\}^T [D] \{v_{i+1}^{(s)}\}| \leq 10^{2-t}, \quad (27)$$

and

$$\max_{1 \leq j \leq f} |\{\tilde{x}_j\}^T [D] \{v_{i+1}^{(s)}\}| \leq 10^{2-t}, \quad (28)$$

or

$$s = 14.$$

If the orthogonality criterion, Equation 28, is satisfied, proceed to block 5.6. Otherwise, set the problem size, m , equal to i and proceed to Exit.

5.6 Normalize the Orthogonalized Trial Vector

Compute

$$\{v_{i+1}\} = \frac{\{v_{i+1}^{(s+1)}\}}{[\{v_{i+1}^{(s+1)}\}^T [D] \{v_{i+1}^{(s+1)}\}]^{1/2}}. \quad (29)$$

This is the new orthogonalized and normalized trial vector.

5.7 Second Normalization Test and Creation of Off-Diagonal Coefficient

- a. Compute the next off-diagonal term in the reduced tridiagonal matrix from

$$d_{i+1} = \{v_{i+1}\}^T [B] \{v_i\}. \quad (30)$$

- b. Verify whether the following test is met:

$$|d_{i+1}| \geq 10^{2-t} |a_{i,i}|. \quad (31)$$

If it has, set $i = i+1$ and return to block 5.3 for continuation of the recurrence algorithm. If the test fails, set $m = i$ to reduce the problem size and proceed to Exit. Only i modes can be obtained since more than $r - f$ modes may have been requested.

6. Solve Reduced-System Eigenvalue Problem

- a. The coefficients $a_{11}, a_{22}, \dots, a_{mm}$ and d_2, d_3, \dots, d_m , computed in block 5 are interpreted as the following symmetric, tridiagonal array:

EIGENVALUE EXTRACTION METHODS

$$[A] = \begin{bmatrix} a_{11} & d_2 & & & \\ d_2 & a_{22} & d_3 & & \\ & d_3 & a_{33} & & \\ & & & \ddots & \\ & & & & d_m \\ & & & & d_m & a_{mm} \end{bmatrix} \quad (32)$$

- b. The m^{th} order eigenvalue problem

$$[A]\{y\} = \bar{\Lambda}\{y\}, \quad (33)$$

is solved for the eigenvalues, $\bar{\Lambda}_i$, and eigenvectors $\{y_i\}$ using a Q-R algorithm and eigenvector computational procedure similar to that described in Sections 10.2.3 and 10.2.4 of the NASTRAN Theoretical Manual.

- c. The reduced system eigenvectors are normalized so that

$$\{y_i\}^T \{y_i\} = 1; \quad i = 1, m. \quad (34)$$

7. Compute Maximum Eigenvalue Errors

- a. The maximum absolute relative errors in the computed physical eigenvalues are obtained from

$$\epsilon_i = \frac{|(\bar{d}_{m+1})(y_{mi})|}{|\bar{\Lambda}_i(1 + \lambda_0 \bar{\Lambda}_i)|}; \quad i = 1, m, \quad (35)$$

where \bar{d}_{m+1} is the last off-diagonal term computed in block 5.3 and y_{mi} is the last element in the vector $\{y_i\}$. If the physical eigenvalue, $\frac{1}{\bar{\Lambda}} + \lambda_0$, corresponds to a rigid body mode, the above computation is invalid and therefore bypassed. A rigid body mode is assumed to occur whenever

$$\left| \frac{1}{\bar{\Lambda}_i} + \lambda_0 \right| \leq 10^{-t/3}, \quad (36)$$

and is denoted by setting the relative error, ϵ_i , equal to a flat zero.

THE TRIDIAGONAL REDUCTION (FEER) METHOD

- b. The eigenvalues are processed in order of increasing distance from the center of range of interest, λ_0 , to determine whether their associated ϵ_i values meet an acceptable relative error tolerance, E , set by the user on the EIGR or EIGB bulk data card (the default value is $(.001/n) \%$ where n is the order of the stiffness matrix. The first eigenvalue not meeting the tolerance test, as well as all subsequent eigenvalues further removed from the center of interest, are considered to lack sufficient accuracy and are therefore rejected.
- c. Acceptable eigenvalues obtained in the above manner are reordered in terms of increasing physical value for subsequent processing by NASTRAN.

8. Compute Physical Eigenvalues and Eigenvectors

The mathematical eigenvalues, $\bar{\lambda}_i$, and eigenvectors, $\{y_i\}$, are converted to physical form as follows:

$$\bar{\lambda}_i = -\frac{1}{\bar{\lambda}_i}, \quad (\text{buckling problems}) \quad (37)$$

$$\bar{\omega}_i^2 = \frac{1}{\bar{\lambda}_i} - \alpha^2, \quad (\text{unshifted vibration mode problems}) \quad (38)$$

$$\bar{\omega}_i^2 = \frac{1}{\bar{\lambda}_i} + \omega_0^2, \quad (\text{shifted vibration mode problems}) \quad (39)$$

$$\{\bar{\phi}_i\} = ([C]^{-1})^T [V] \{y_i\}, \quad (\text{buckling or unshifted vibration mode problems}) \quad (40)$$

$$\{\bar{\phi}_i\} = [V] \{y_i\}, \quad (\text{shifted vibration mode problems}) \quad (41)$$

where

$$[V] = [\{v_1\}, \{v_2\}, \dots, \{v_m\}]. \quad (42)$$

EIGENVALUE EXTRACTION METHODS

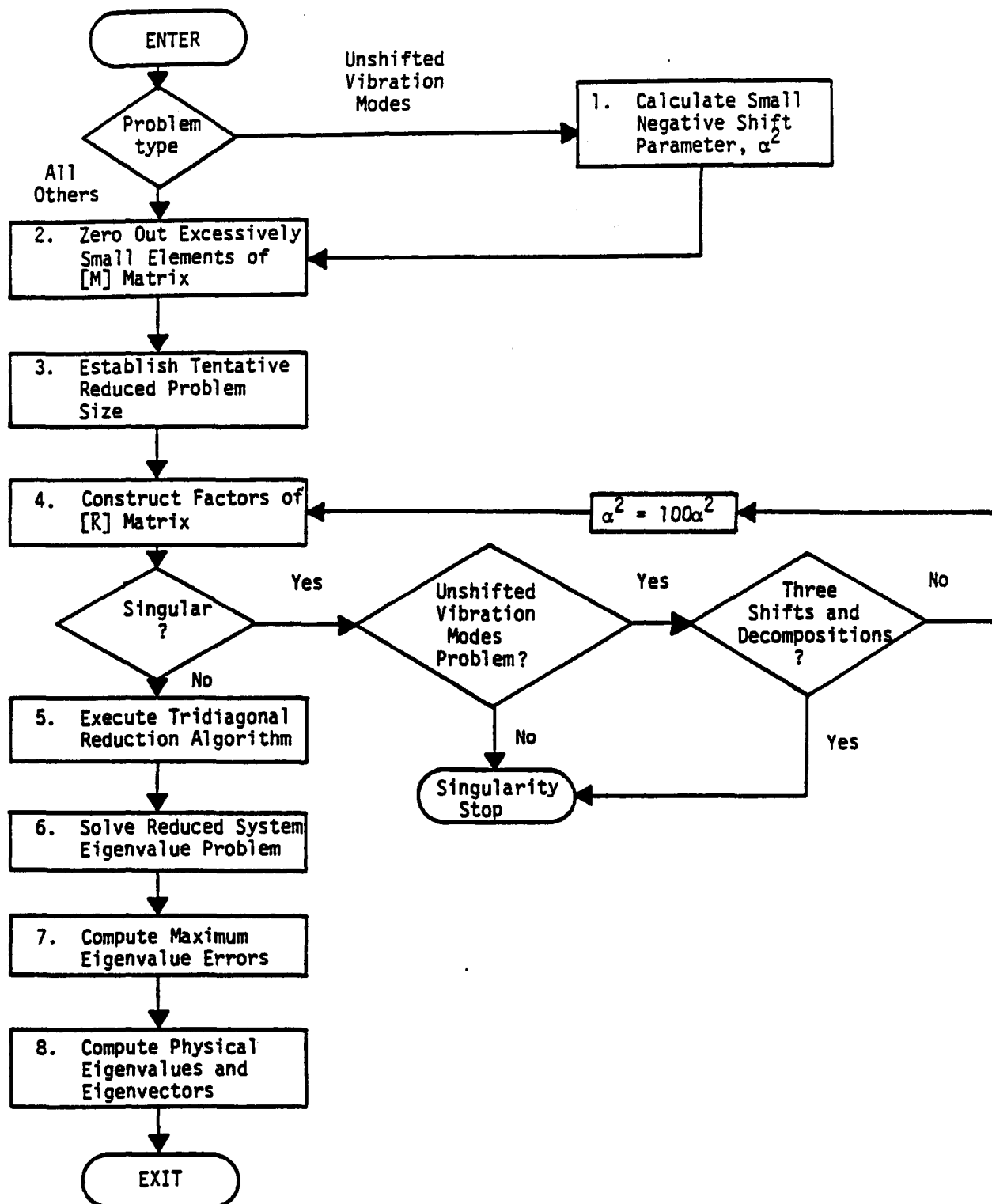


Figure 1. Overall Flow Diagram for Tridiagonal Reduction Method

THE TRIDIAGONAL REDUCTION (FEER) METHOD

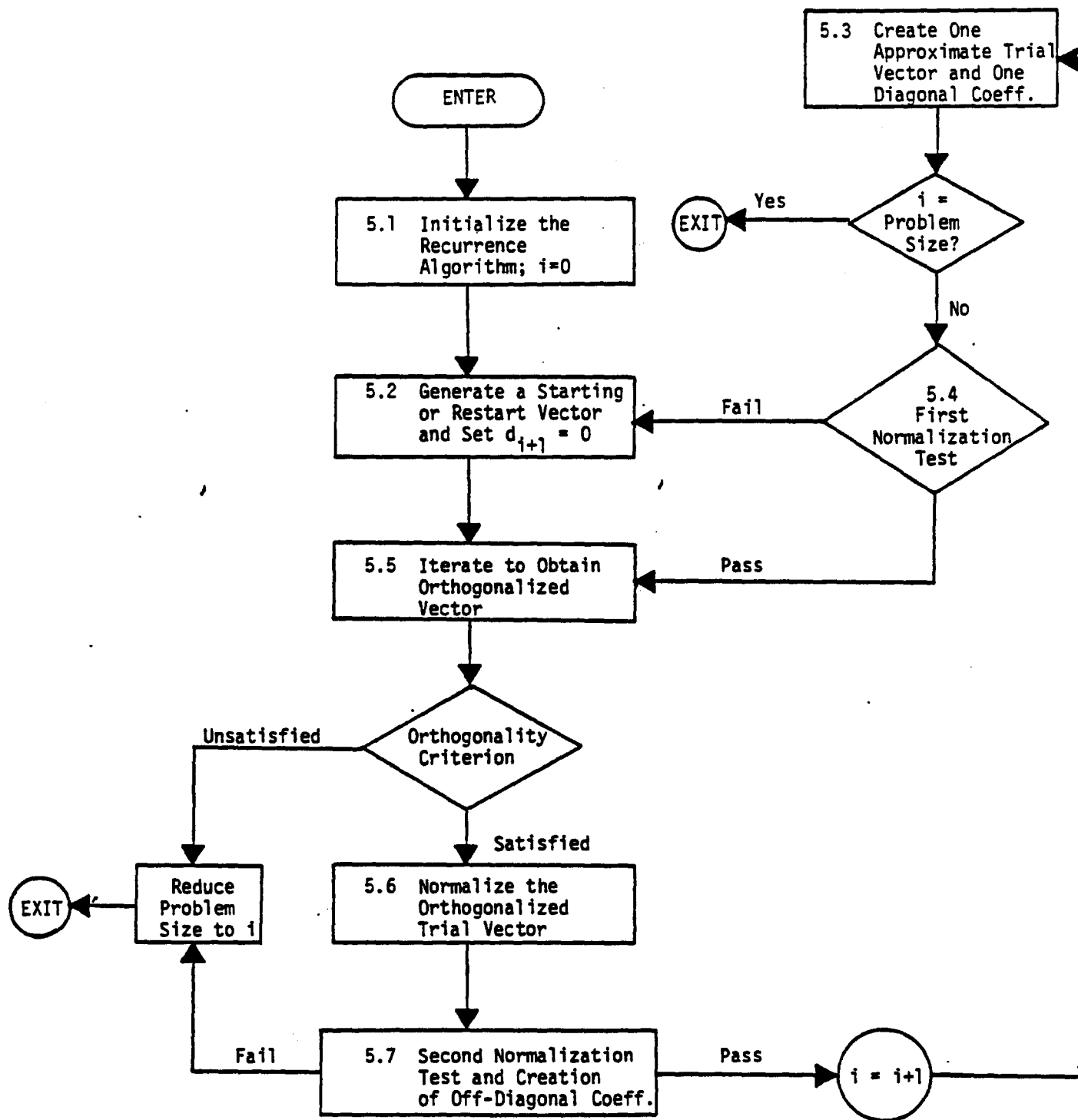


Figure 2. Flow Diagram for Block 5, Execute Tridiagonal Reduction Algorithm

9.4.4 Summary of Procedures for Complex Eigenvalue Analysis

The theory and computational procedures for complex eigenvalue analysis are described in Sections 10.6.4 and 10.6.5 of the Theoretical Manual. The major differences, as compared to real eigenvalue analysis, are:

1. Left and right bi-orthogonal vectors must be created in the process of constructing the reduced tridiagonal matrix.
2. The reduced tridiagonal matrix, while symmetric in form, is, in general, complex rather than real.
3. The calculated theoretical errors in the computed eigenvalues are estimates rather than upper bounds.
4. The general complex eigenvalue problem is of the form

$$[Mp^2 + Bp + K]\{u\} = 0, \quad (43)$$

where $[M]$, $[B]$ and $[K]$ may be real, complex, symmetric or unsymmetric, singular or non-singular. The problem size is doubled when $[B]$ is not null.

5. Eigensolutions, p_i , closest to one or more specified points (shift points) in the complex plane are found. All eigensolutions obtained for previous shift points are swept out of the problem to prevent their re-generation when dealing with the current shift point.