AD-755 370

MAGIC III: AN AUTOMATED GENERAL PURPOSE SYSTEM FOR STRUCTURAL ANALYSIS. VOLUME III. PROGRAMMER'S MANUAL

A. Michael Gallo

Bell Aerospace Company

Prepared for:

Air Force Flight Dynamics Laboratory

July 1972

DISTRIBUTED BY:

NTIS

National Technical Information Service
U. S. DEPARTMENT OF COMMERCE
5285 Port Royal Road, Springfield Va. 22151

AD755370

AFFDL-TR-72-42
VOLUME III

MAGIC III: AN AUTOMATED GENERAL PURPOSE SYSTEM FOR STRUCTURAL ANALYSIS

VOLUME III: PROGRAMMER'S MANUAL

A. MICHAEL GALLO

BELL AEROSPACE COMPANY

TECHNICAL REPORT AFFDL-TR-72-42, VOLUME III

JULY, 1972

DDC
RECEIVED
FEB 12 1973

Approved for public release; distribution unlimited.

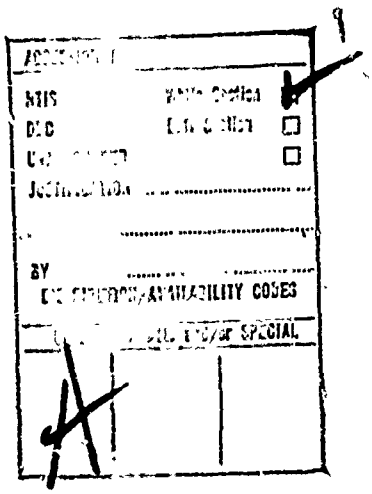
Reproduced by
NATIONAL TECHNICAL
INFORMATION SERVICE
U S Department of Commerce
Springfield VA 22151

AIR FORCE FLIGHT DYNAMICS LABORATORY
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO

722

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use or sell any patented invention that may in any way be related thereto.



Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

AIR FORCE/56780/18 December 1972 — 400

Unclassified
Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)
Bell Aerospace Company
A Division of Textron
Buffalo, New York 14240

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP
N/A

3. REPORT TITLE
MAGIC III - An Automated General Purpose System for Structural Analysis - Volume III - Programmer's Manual

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)
Final Report

5. AUTHOR(S) (First name, middle initial, last name)
A. Michael Gallo

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
August, 1972		

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
AF 33615-71-C-1390	AFFDL-TR-72-42 - Volume III
b. PROJECT NO. 1467	
c. Task No. 146702	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d.	

10. DISTRIBUTION STATEMENT
This document has been approved for public release and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
None	Air Force Flight Dynamics Laboratory, Structures Division for Wright-Patterson AFB, Ohio 45433

13. ABSTRACT

An automated general purpose system for analysis is presented. This system, identified by the acronym, "MAGIC III" for Matri· Analysis via Generative and Interpretive Computations, is an extension of the structural analysis capabilit, available in the initial MAGIC System. MAGIC III provides a powerful framework for implementation of the finite element analysis technology and provides diversified capability for displacement, stress, vibration, and stability analyses.

Additional elements have been added to the MAGIC element library in this phase of MAGIC development. These are the solid elements; rectangular prism, tetrahedron, triangular prism, symmetric triangular prism, and triangular ring (asymmetrical loading). Also included are the symmetric shear web element and a revised quadrilateral thin shell element. The finite elements listed include matrices for stiffness, mass, prestrain load, thermal load, distributed mechanical load, pressure and stress.

Documentation of the MAGIC III System is presented in three parts; namely, Volume I: Engineer's Manual, Volume II: User's Manual and Volume III: Programmer's Manual.

DD FORM 1 NOV 68 1473

Unclassified
Security Classification

IQ

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
1. Structural Analysis						
2. Matrix Methods						
3. Matrix Abstraction						
4. Digital Computer Methods						
5. Finite Element Techniques						

*U.S.Government Printing Office: 1972 — 759-438/252

Unclassified
Security Classification

Ib

MAGIC III: AN AUTOMATED GENERAL PURPOSE SYSTEM FOR STRUCTURAL ANALYSIS

VOLUME III: PROGRAMMER'S MANUAL

A. MICHAEL GALLO

Approved for public release; distribution unlimited.

IC

FOREWORD

This report was prepared by Textron's Bell Aerospace Company (BAC), Buffalo, New York, under USAF Contract No. F33615-71C-1390. The contract was initiated under Project No. 1467, "Structural Analysis Methods", Task No. 146702 "Thermal Elastic Analysis Methods." The program was administered by the Air Force Dynamics Laboratory (AFFDL), Air Force Systems Command, Wright-Patterson Air Force Base, Ohio 45433 under the cognizance of Mr. G.E. Maddux, AFFDL Program Manager. The Program was carried out by the Structural Systems Department, Bell Aerospace Company, during the period 15 March 1971 to 15 March, 1972, under the direction of Mr. Stephen Jordan, Program Manager.

This report, "MAGIC III: An Automated General Purpose System for Structural Analysis" is published in three volumes, "Volume I: Engineer's Manual", "Volume II; User's Manual", and "Volume III: Programmer's Manual". The manuscript for Volume III was released by the author in January 1972 for publication.

The author wishes to thank Miss Beverly Dale for her contribution to the development of the MAGIC System, and to acknowledge the assistance of the following personnel: M. Morgante, S. Skalski, W. Crill, W. Luberacki, S. Mah.

This technical report has been revised and is approved.

FRANCIS J. JANIK, JR.
Chief, Theoretical Mechanics Branch
Structures Division

ABSTRACT

An automated general purpose system for analysis is presented. This system, identified by the acronyn "MAGIC III" for "Matrix Analysis via Generative and Interpretive Computations," provides a flexible framework for implementation of the finite element analysis technology. Powerful capabilities for displacement, stress and stability analyses are included in the subject MAGIC III System for structural analysis.

The matrix displacement method of analysis based upon finite element idealization is employed throughout. Sixteen versatile finite elements are incorporated in the finite element library. These are: frame, shear panel, triangular cross-section ring, toroidal thin shell ring, quadrilateral thin shell, triangular thin shell, trapezodial ring, triangular plate, incremental frame, quadrilateral plate, tetrahedron, triangular prism, rectangular prism, symmetrical shear web, asymmetric triangular cross-section ring and high aspect-ratio quadrilateral thin shell elements. These finite element representations include matrices for stiffness, consistent mass, incremental stiffness, thermal stress, thermal load, distributed mechanical load, and stresses.

The MAGIC III System for structural analysis is presented as an integral part of the overall design cycle. Considerations in this regard include, among other things, preprinted input forms, automated data generation, data confirmation features, restart options, automated output data reduction and readable output displays.

Documentation of the MAGIC III System is presented in three parts; namely, Volum I: Engineer's Manual, Volume II: User's Manual and Volume III: Programmer's Manual. The subject Volume, Volume III, is designed to facilitate implementation, operation, modification, and extension of the MAGIC III System.

TABLE OF CONTENTS

Section		Page No.
I	INTRODUCTION	1
II	COORDINATION OF STRUCTURAL GENERATIVE SYSTEM WITH FORMAT II	3
	A. Detailed Analysis of .USER04. Instruction	3
	1. Input and Output Matrix Position Functions	3
	2. Suppression Option	5
	B. Use of FORMAT II Data Sets	6
	1. Master Input and Master Output Use for Material Library	6
	2. Instruction Input Data Sets	6
	3. Instruction Output Data Sets	7
	4. Scratch Data Sets	7
III	ORGANIZATION OF STRUCTURAL GENERATIVE SYSTEM .	9
	A. Basic Logic Flow	9
	B. Input Phase Logic Flow	9
	1. Report Form Input	9
	2. Interpreted Input	11
	C. Element Matrices Generation Phase Logic Flow	11
	D. Output Phase Logic Flow	12
	1. Organization of Output Matrices . . .	12
	2. Sequence of Output Matrices	17

TABLE OF CONTENTS, Continued

Section		Page No.
IV	OPERATIONAL CONSIDERATIONS	20
	A. Implementation	20
	1. Direct Machine Control	20
	2. SUBSYS Control	20

APPENDICES

I	Overlay Structure	1.1
II	Logical Flow Charts	2.1
III	List of Structural System Subroutine Functions.	3.1
IV	List of Subroutine Functions of Modules Added to the FORMAT II System	4.1
V	Revisions to FORMAT System Decks	5.1
VI	MAGIC Error Messages	6.1
VII	Examples of Static and Stability Instruction Sequences	7.1
VIII	Subroutine Documentation	8.1
IX	Direct Machine Control Implementation Document	9.1
X	SUBSYS Control Documentation	10.1
XI	Documentation for Element Insertion Into the MAGIC System	11.1

REFERENCES 12.1

SECTION I

INTRODUCTION

A Structural Generative System has been developed and inserted into FORMAT II (Reference 2) for the purpose of generating structural matrices for use by FORMAT II. The insertion of a Structural Generator into FORMAT II resulted in a computer program retaining ease of implementation and use, yet offering diversified capabilities.

Machine compatibility has been retained by the complete use of FORTRAN IV in the development of the structural Generative System. The absence of machine or assembler language from every portion of the program eliminates the problems of machine dependency and implementation difficulty.

Input to the Structural Generative System is accomplished by filling in preprinted structural engineering oriented input sheets. The combination of these sheets and the normal matrix abstraction instructions of FORMAT II allows minimal training for use of the program, thus decreasing the possibility of input errors.

The program is capable of restart at any point in the abstraction instruction sequence stipulated at the discretion of the User. Input data, intermediate results, final results or any matrix whatsoever may be automatically saved, by use of the proper instruction, and used as a starting point or new input to subsequent applications on continuing or independent projects.

The MAGIC System consists of a total of 477 subroutines of which 296 form the Structural Generative System. The 477 subroutines can be logically designed into an overlay structure which reflects the optimum use of available storage in relation to the longest link so that the program will maintain respectable execution efficiency. The Structural Generative System requires a minimum of 13,000 decimal words of work storage which is assigned to an unlabeled common block. A minimum of eight external storage units available to the FORMAT II System are required for use of the Structural Generative System, including at least one assigned to the Master Input FORMAT function, one assigned to the Master Output FORMAT function, and four assigned to the Utility FORMAT function. The other two units are necessary for intermediate matrix results and for an instruction data set. The MAGIC System needs 48,000 decimal words of internal storage to execute on an IBM 360/65 using a 91 link OVERLAY structure and a blank common area of 18,000 decimal words. (not considering internal core necessary for I/O buffers and OS system routines). Using the three level OVERLAY of CDC and a blank common area of 13,000 words, the MAGIC System can execute using 34,698 decimal (103,612 octal) words of internal storage on the CDC 6400, not considering internal storage for I/O buffers and SCOPE system routines necessary to execute the OVERLAY program.

The MAGIC System has been implemented on the IBM 360/65 under direct machine control, but some installations may not be able to execute MAGIC under direct machine control. This was the case when the MAGIC I System was implemented on the IBM 7090.

The number of subroutines contained in the FORMAT II program necessitated the use of SUBSYS, a software package developed by Westinghouse, which improved the loading capabilities of IBSYS on the IBM 7090/94. In addition to allowing the program to be loaded, SUBSYS allowed the program overlay tape to be saved, thereby improving execution time. Programs may be stacked on this overlay tape. Taking advantage of this fact, FORMAT II with the Structural Generative System insertion, was actually three programs executed automatically with no intervention by IBSYS. The first program consisted of the FORMAT II Preprocessor, the second consisted of the FORMAT II Execution Monitor and the third contained the Structural Generative System. Although the Structural Generative System was actually a separate program when operating under SUBSYS control on the IBM 7090/94, it is activated and controlled as a normal User Module under the FORMAT II System. Explicitly, the Structural Generative System is the fourth User Module (USER04) available under FORMAT II.

SECTION II

COORDINATION OF STRUCTURAL GENERATIVE SYSTEM WITH FORMAT II

A. DETAILED ANALYSIS OF USER04 INSTRUCTION

1. Input and Output Matrix Position Functions

The Structural Generative System may have as many as fifteen actual output matrices and require as many as four actual input matrices. The basic form of the USER04 instruction may be represented as follows:

```
OMP1, OMP2, OMP3, OMP4, OMP5, OMP6, OMP7, OMP8,
OMP9, OMP10,OMP11, OMP12, OMP13, OMP14, OMP15 =
IMP1, IMP2, IMP3, IMP4 .USER04.  ;
```

where OMP is read as output matrix position and IMP as input matrix position. All matrix positions, whether input or output, must be present. They may contain matrix names or be blank, but there must be nineteen matrix positions represented by the appropriate number of commas. Blank matrix positions are discussed in the next section. The output matrix positions, if nonblank, will contain the following matrices upon exit from the Structural Generative System:

OMP1	-	copy of input structure data deck
OMP2	-	revised material library
OMP3	-	interpreted input (structure input data as stored after being read and interpreted)
OMP4	-	external system grid point loads and load scalar matrix
OMP5	-	transformation matrix for application of boundary conditions
OMP6	-	transformation matrix for assembly of element matrices
OMP7	-	element stiffness matrices stored as one matrix
OMP8	-	element generated load matrices stored as one matrix
OMP9	-	element stress matrices stored as one matrix
OMP10	-	element thermal stress matrices stored as one matrix
OMP11	-	element incremental stiffness matrices stored as one matrix

OMP12 - element mass matrices stored as one matrix
OMP13 - structural system constants stored as one matrix
OMP14 - element matrices in compressed format stored as one matrix
OMP15 - prescribed displacement matrix

The input matrix positions, if nonblank must contain the following matrices:

IMP1 - structure data deck (this would be a previously generated matrix saved in OMP1)
IMP2 - interpreted input (this would be a previously generated matrix saved in OMP3 used for restart)
IMP3 - existing material library (this would be a previously generated matrix saved in OMP2)
IMP4 - displacement or stress matrix to be used for stability analyses (the stress matrix must have been generated by the structural abstraction instruction .STRESS.)

It should be noted that the following matrix positions are called matrices only in the sense that all input and output entities are considered matrices by FORMAT II - OMP1, IMP2, OMP3, OMP14, IMP1, IMP2 and IMP3.

It is important to note that OMP14 is mutually exclusive with OMP6, OMP7, OMP8, OMP9, OMP10, OMP11, and OMP12. In order to retain compatability with the MAGIC I system and eliminate redundant execution time, the following rules must be observed.

(a) If OMP14 is suppressed then OMP6, OMP7, OMP8, OMP9, OMP10, OMP11, and OMP12 will be generated according to their definition in Part A.1 of Section II. If this is the case then it is assumed the user is using MAGIC I abstraction instructions to solve his problem.

(b) If OMP14 is not suppressed then OMP7, OMP8, OMP9, OMP10, OMP11 and OMP12 will serve only as indicators to the .USER04. instruction for generation or non-generation of their respective element matrices. Since no matrices will be generated in OMP6 through OMP12 (if OMP14 is not suppressed) they should never be referenced in subsequent abstraction instructions.

2. Suppression Option

Incorporated into the Structural Generative System is an option to suppress the generation and output of any of the output matrices and also to indicate the absence of any of the input matrices. This option is indicated to the Structural Generative System by the absence of a matrix name in the desired position in the .USER04. instruction. A matrix name is considered to be absent if the matrix position contains all blanks or the character length of the name is zero. For example, an instruction of the form: ,,INTINP, LOADS, TR, TA, KEL, FEL, SEL, SZALEL,,,,, = ,,MATLB1,.USER04.; would cause suppression of the copy of the data deck, the revised material library, the element incremental stiffness matrices, the element mass matrices, the structural system constant matrix, the compressed element matrix and the prescribed displacement matrix. The instruction also indicates that there is no input data deck on tape, (directing the Structural Generative System to read data from cards), no interpreted data on tape and no input displacements or stresses. It should be noted that certain sections of the data deck are necessary for the generation of each of the output matrices and that error checking is done to determine if the required sections are present. A table of the required data sections for generation of each matrix appears in the User's Manual. Accordingly, error checking is invoked for the input matrix positions to determine if ambiguous or conflicting input indications have been made.

Internally, the logic flow of the suppression option is controlled by inserting key characters for suppressed matrices. Upon detection of a suppressed matrix by Subroutine INST, a matrix name of the form ////XX is inserted into that matrix position. The four slashes are inserted for recognition by the Structural Generative System of a suppressed matrix and the last two positions may each contain the digits 0-9 assigned sequentially starting from 00 for each suppressed matrix encountered. The last two positions in the inserted name for suppressed matrices ensure that each suppressed matrix name will be unique, thereby eliminating inconsistencies in the FORMAT II Preprocessor.

Suppressed input matrices, i.e. those occurring to the right of the equal sign in the input .USER04. abstraction instruction, are recorded on NDATA, the data set reserved for card input matrices, as null matrices to satisfy FORMAT II Preprocessor input matrix existence requirements. This operation is accomplished by subroutine MATSUP.

B. USE OF FORMAT II DATA SETS

1. Master Input and Master Output Use for Material Library

References to the Material Library are indicated by output matrix position two and input matrix position three in the .USER04. abstraction instruction. Retention of a newly generated or revised Material Library is governed solely by use of the SAVE abstraction instruction at the discretion of the User. If retention is desired, the matrix name in output matrix position two must appear in a SAVE abstraction instruction, in which case it will be placed on a Master Output tape. If a non-blank matrix name appears in input matrix position three, the Master Input Tape will be searched for that name.

Usage and generation of the Material Library is controlled by the three legal combinations of suppression of output matrix position two and input matrix position three. If the matrix name in output matrix position two is non-blank, but input matrix position three is suppressed, a new Material Library will be generated and used. If both involved matrix positions are non-blank, the old Material Library will be located on the Master Input tape, will be revised, stored as the matrix named in the specified output position, and then this revised Material Library will be used. If output matrix position two is suppressed and input matrix position three is non-blank, then the named input Material Library will be used: Suppression of both involved matrix positions results in an error condition.

Since the material library is stored under a matrix name on Master Output tapes, and also, therefore Master Input tapes, any other matrices may also be saved on the same tape, including other Material Libraries.

2. Instruction Input Data Sets

An instruction input data set is an external storage unit that contains at least one of the non-blank matrices named in input matrix positions one, two, three or four in

the .USER04. abstraction instruction. The Structural Generative System conforms to all the rules of FORMAT II with regard to use of instruction input data sets. All searching, reading, and rewinding is accomplished by use of the FORMAT II data set handling subroutines EUTL1-EUTL9. No attempt is ever made to write on an instruction input data set.

3. Instruction Output Data Sets

An instruction output data set is an external storage unit which has been designated by the FORMAT II Preprocessor to contain at least one of the non-blank matrices in output matrix positions one to fifteen in the .USER04. abstraction instruction. The Structural Generative System conforms to all rules of FORMAT II with regard to instruction output data sets by using the FORMAT II data set handling subroutines EUTL1-EUTL9 to write all matrix headers, matrix trailers, data set trailers and end of files on instruction output data sets. All matrices are stored by column in the required record format. No attempt is ever made by the Structural Generative System to rewind an instruction output data set.

4. Scratch Data Sets

Scratch data sets are external storage units that have been assigned by the FORMAT II System to the Structural Generative System to be used as temporary storage areas. There are no reading, writing or rewinding rules imposed on scratch data sets by the FORMAT II System. The required four scratch data sets are assigned to the following functions by the Structural Generative System:

SCRATCH DATA SET 1 - 1st use - external storage areas for report form input preprocessor

2nd use - contain structure control information including system orders, boundary conditions and system print operations

SCRATCH DATA SET 2 - 1st use - contain temporary copy of direct input structure data deck

2nd use - contain generated element matrices in compact form

SCRATCH DATA SET 3 - 1st use - contain temporary copy of actual input deck

2nd use - contain element input data after reading and interpretation

SCRATCH DATA SET 4 - 1st use - external storage area for report form input preprocessor

2nd use - contain input loads matrix

3rd use - contain input displacements or input stresses, if any

SECTION III

ORGANIZATION OF STRUCTURAL GENERATIVE SYSTEM

A. BASIC LOGIC FLOW

The Structural Generative System has three basic phases of operational flow; the input phase, the element matrices generation phase, and the output phase. The input phase consists of reading, interpreting and storing the information contained in the structure data deck. From the stored input, the element matrices selected are generated in the second phase. Phase three outputs all non-suppressed matrices indicated by the .USER04. abstraction instruction in output matrix position six through twelve, if output matrix position fourteen has been suppressed, or outputs only output matrix position fourteen if it was non-suppressed. Output matrix positions one through five and thirteen and fifteen are generated directly from the input structure data deck and for this reason are actually output during the first or input phase. Subroutine US04 controls the three logical phases by directly controlling subroutine US04A which controls the input phase and US04B which controls the generation and output phases. Normally, the basic logical flow of the Structural Generative System would be sequentially through the three phases, however, by use of the suppression option, it is possible to completely skip a given phase. The actual logic flow of the system is created by subroutine LOGFLO as determined by the .USER04. abstraction instruction. For example, if the .USER04. instruction was written such that only the boundary conditions had changed and the remainder of the necessary matrices were saved from a previous application as indicated by the suppression option, subroutine LOGFLO would eliminate the second and third phases.

B. INPUT PHASE LOGIC FLOW

The logic flow of the input phase is determined by the type of input encountered. The two types of input are report form input and interpreted input.

1. Report Form Input

The location of the input data deck is determined by examining IMP1 of the input .USER04. abstraction instruction. If this input position was blank, then the data deck is assumed to be on NFIT, the system input unit. If IMP1 contained a

non-blank matrix name, then the input data deck exists as a matrix and the original card form deck is reconstructed by subroutine INDECK.

Report Form Input is a highly flexible, engineering oriented type of input for the Structural Generative System. From a programming viewpoint, report form input allows ease of use by the Analyst and by translation allows logical readability by the program.

Encountering a report form input deck causes the input phase to pass control to the Report Form Input Preprocessor. Basically, the report form input preprocessor translates the flexible report form input deck into a sophisticated direct input deck. Translation is accomplished by two steps controlled by subroutine REFORM.

The first step is to read and store the report form input deck. This step is accomplished by subroutine PHASE1 with support by subroutines LATCH and FORMIN. PHASE1 controls all storage, both internal core storage and external storage on scratch data sets one and four. LATCH performs label matching tests to determine the various sections of input and FORMIN reads all table form input, sections; non-table form input sections are read directly in PHASE1.

The second step in processing a report form input deck is to merge the data stored by the first step into a direct data deck. These two operations are performed by subroutine PHASE2 supported by subroutine OPEN. The information stored by the first step is merged into a compact direct data deck by PHASE2 and output on scratch data set two. The OPEN subroutine aids PHASE2 by locating, in any order designated by PHASE2 the input sections stored on scratch data sets one and/or four. At this point, a complete direct data input deck is resident on scratch data set two and control returns to US04A. Once a direct data deck is resident on scratch data set two, reading, interpreting and storage is controlled by subroutine INPUT with each input section handled as indicated by the following table:

INPUT SECTION	SUBROUTINE	INTERPRETED STORAGE
Title (TITLE)	INPUT	None
System Control (SYSTEM)	INPUT	Scratch data set 1
Grid Points (COORD)	INPUT	Scratch data set 3
Boundary Conditions (BOUND)	BOUNT	Scratch data set 3
Element Definitions (ELEM)	ELEM	Scratch data set 3
Grid Point Loads (LOADS)	FGRLDS	Scratch data set 4
Grid Point Axes (GRAXES)	FRED	Scratch data set 3
Material Library Requests (MATER)	FMAT	Master Output data set
Grid Point Temperatures (TEMP)	INPUT	Scratch data set 3
Grid Point Pressures (PRESS)	INPUT	Scratch data set 3
Prescribed Displacements (PDISP)	BOUND	Scratch data set 3

If output matrix position one was non-blank, then a copy of the actual input data deck is also written on the instruction output data set specified by the FORMAT II System by subroutine COPYDK.

2. Interpreted Input

After the data deck has been read and interpreted under control of subroutine INPUT, all pertinent data exists on scratch data sets one and three. If output matrix position three in the .USER04. abstraction instruction is non-blank, then the contents of scratch data sets one and three are output under that matrix name onto the instruction output data set specified by the FORMAT II System by subroutine OUTINT. If this "matrix" is saved and input at input matrix position two in the .USER04. instruction, the Structural Generative System is capable of restart at the second or element generation phase, thereby eliminating a repeat of the input phase. This feature is recommended for usage on large applications where the procedure would be to run the data deck, stop after interpreting and storing the data, check for input errors, and if no errors are present restart at the element generation phase.

Before exiting from the input phase, subroutine CHEK is called to perform input error cross-checking. While determining the logical flow at the Structural Generative System, subroutine LOGFLO also recorded the input sections required to generate the requested output matrices. If any of the required input sections have not been processed, then execution will be terminated after the input phase.

C. ELEMENT MATRICES GENERATION PHASE LOGIC FLOW

The second phase of operation of the Structural Generative System consists of generation of the element matrices.

If input matrix position two of the input .USER04. abstraction instruction is non-blank, then subroutine ININT is called to reconstruct the data on scratch data sets one and three from the input matrix.

If input matrix position four of the input .USER04. abstraction instruction is non-blank, then subroutine DEFLEX is called to store the input displacements or stresses (which ever was input) on scratch data set four.

At this point all the necessary data is located on scratch data sets one and three, placed there by either phase one or restart using input matrix position two of the .USER04. abstraction instruction. Basic control of the second phase is accomplished by subroutine FELEM under subroutine US04B. FELEM reads scratch data set one to obtain system control information and sets suppression controls to eliminate generation of undesired element matrices by calling subroutine SQUISH. Scratch data set three contains the necessary input for each element, one set of element input per record. For each element, subroutine ELPLUG reads an element input record, selects the proper element to calculate the matrices and then writes the generated matrices on scratch data set two in compact form.

Prior to being written upon scratch data set two, the element matrices are temporarily stored in the blank common work area. Also, all work areas that are needed by the specific element are allocated from the blank common work area. For these reasons, the Structural Generative System requires a blank common work area of at least 13,000 words of internal core storage.

Imbedded into the Element Matrices Generation Phase, at strategic locations, are utility packages accessible by the specific elements which require their capabilities. Integration packages and small scale matrix operation packages are examples of utility sections commonly accessible to the necessary elements. The exact locations of these packages are indicated by the Structural System Overlay Chart (Appendix I). Overlay to each element has been avoided wherever possible to reduce execution process time. However, an area of approximately 1000 locations between the longest link and the origin of the common area has been kept clear to allow for future substantial alterations to be made without redesigning the complete overlay structure.

D. OUTPUT PHASE LOGIC FLOW

1. Organization of Output Matrices

All output entities from the Structural Generative System are written following the rules of the FORMAT II System. Each output entity is written as a matrix, consisting of a matrix header, matrix column records and a matrix trailer. The following list exhibits the contents, interpretation of matrix header information (number of rows, number of columns) and interpretation of matrix column records for each output position in the .USER04. abstraction instruction.

a. Output Matrix Position One (OMP1)

Contents	- Copy of card input data deck
Number of rows	- Set to eighty (80)
Number of columns	- Number of cards in data deck
Column records	- One data card per column record, one card column per row

b. Output Matrix Position Two (OMP2)

Contents	- Material library
Number of rows	- 306 (maximum number of words possible for one material entry)
Number of columns	- Number of material tables in library plus one
Column records	- One material table per column record

c. Output Matrix Position Three (OMP3)

Contents	- Interpreted input
Number of rows	- Set to number of words in maximum record created
Number of columns	- Number of elements plus four
Column records	- One element input block per record

d. Output Matrix Position Four (OMP4)

Contents	- External system grid point loads
Number of rows	- Number of degrees of freedom in total system plus 1
Number of columns	- Number of load conditions
Column records	- The first word is the external load scalar followed by one load condition per column record (use .DEJOIN. to obtain the load scalar).

e. Output Matrix Position Five (OMP5)

Contents	- Transformation matrix for application of boundary conditions
Number of rows	- Number of degrees of freedom in total system
Number of columns	- Number of degrees of freedom in total system

Column records - (1) for desired degrees of freedom - contain a one in the assigned reduced degree of freedom row

(2) for undesired degrees of freedom - column record is omitted (null column)

f. Output Matrix Position Six (OMP6)

Contents - Transformation matrix for assembly of element matrices
Number of rows - Number of degrees of freedom in total system
Number of columns - Summation of element degrees of freedom
Column records - Contain a one in the assigned degree of freedom row for that summed element degree of freedom

g. Output Matrix Position Seven (OMP7)

Contents - Element stiffness matrices
Number of rows - Summation of element degrees of freedom
Number of columns - Summation of element degrees of freedom
Column records - Each record contains a column of an element stiffness matrix

h. Output Matrix Position Eight (OMP8)

Contents - Element applied load matrices
Number of rows - Summation of element degrees of freedom
Number of columns - One
Column record - Contains all element applied load matrices

i. Output Matrix Position Nine (OMP9)

Contents - Element stress matrices
Number of rows - Summation of element stress point and component orders
Number of columns - Summation of element degrees of freedom
Column records - Each record contains a column of an element stress matrix

j. Output Matrix Position Ten (OMP10)

Contents	- Element thermal stress matrices
Number of rows	- Summation of element stress point and component orders
Number of columns	- One
Column record	- Contains all element thermal stress matrices

k. Output Matrix Position Eleven (OMP11)

Contents	- Element incremental stiffness matrix
Number of rows	- Summation of element degrees of freedom
Number of columns	- Summation of element degrees of freedom
Column records	- Each record contains a column of an element incremental stiffness matrix

l. Output Matrix Position Twelve (OMP12)

Contents	- Element mass matrices
Number of rows	- Summation of element degrees of freedom
Number of columns	- Summation of element degrees of freedom
Column records	- Each record contains a column of an element mass matrix

m. Output Matrix Position Thirteen (OMP13)

Contents	- System constants
Number of rows	- Twenty-seven
Number of columns	- One
Column record	- Ninteen structural system constants (for use outside of the .USER04. module)

The following is a description of the variables in this matrix:

Word 1	- Number of directions allowed
Word 2	- Number of types of movement allowed
Word 3	- Number of reference points (highest reference node in element connections)
Word 4	- Order of the reduced system (number of 1's plus 2's)
Word 5	- Number of bounded degrees of freedom (number of 0's)

Word 6 - Number of unknown degrees of freedom (number of 1's)
Word 7 - Number of known degrees of freedom (number of 2's)
Word 8 - Number of 0's plus 1's
Word 9 - Element type code, equal to zero if word 2=3, equal to one otherwise
Word 10 - Order of the total system
Word 11 - Number of elements
Word 12 - Number of load conditions
Word 13 - Word 20 - Reserved for future expansion
Word 21 - Number of eigenvalues requested
Word 22 - Eigenvalue/vector convergence criteria
Word 23 - Maximum number of iterations
Word 24 - Control for iteration debug print
Word 25 - First normalizing element for print
Word 26 - Second normalizing element for print
Word 27 - Control for guess vector iteration start

n. Output Matrix Position Fourteen (OMP14)

Contents - Element matrices in compressed form
Number of rows - Varies depending on problem
Number of columns - One column for each element
Column records - Each record contains all element matrices generated by .USER04. instruction in compressed form (to be used by structural modules outside of .USER04.)

o. Output Matrix Position Fifteen (OMP15)

Contents - Prescribed displacements
Number of rows - Number of degrees of freedom in system
Number of columns - Number of load conditions
Column records - One prescribed displacement condition per column record

It should be noted that OMP1, OMP2, or OMP3 and OMP14 are not actually matrices and, therefore, should never be referenced as input to an algebraic matrix operation. OMP7, OMP9, OMP11 and OMP12 are formed by placing the element matrices into the output matrix such that the main diagonal of the element matrix coincides with the next available main diagonal positions in the output matrix. For example, if the first two element stiffness matrices represented 48 element degrees of freedom each (such as 8 element defining points with 6 degrees of freedom each) then the first would be located in rows one to 48 and column one to 48 in the output matrix and the second would be placed into rows 49 to 96 and columns 49 to 96. Output matrices in these positions are almost always written in FORMAT II compressed column format due to the inherent sparseness of non-zero matrix elements.

OMP8 and OMP10 are formed by placing each element matrix, which is a column matrix, into the succeeding available row positions in the output matrix.

2. Sequence of Output Matrices

Output matrix positions one to five, thirteen and fifteen are output sequentially in numerical order by the Structural Generative System. Since these seven matrices are generated directly from data contained in the input deck, they are output, if non-blank, as part of phase one or input phase operations. Specifically, these seven output matrices are placed into the FORMAT II system by the following subroutines in phase one:

OMP1	-	Subroutine COPYDK
OMP2	-	Subroutine FMAT
OMP3	-	Subroutine OUTINT
OMP4	-	Subroutine FLOADS
OMP5	-	Subroutine FTR
OMP13	-	Subroutine TSYS
OMP15	-	Subroutine PDISP

Either output matrix positions six through twelve or output matrix position fourteen is released into the FORMAT II System during phase three of the Structural Generative System. Output of matrices six through twelve is controlled by subroutine OUTMAT using utility subroutines US461, US462 and US463. In contrast to output of the first seven matrices, which is achieved consecutively, output of matrices six through twelve will

usually occur concurrently. Output matrix position fourteen is released to the FORMAT System by subroutine ELMAT. Since output matrix fourteen is mutually exclusive with output matrice six through twelve only one of the above subroutines OUTMAT or ELMAT is activated.

Operational flow in the output phase of matrices six through twelve, if output matrix fourteen is suppressed, consists of extracting the compacted element matrices from scratch data set two and releasing them to the FORMAT II System in the required form. Due to the fact that more than one output matrix may have been assigned tc the same instruction output data set by the FORMAT II System, direct output at matrix generation time (phase two) is impossible, thus necessitating the use of scratch data set two. However, at output time, the optimum procedure is determined by subroutine OUTMAT to achieve multiple matrix output per pass of scratch data set two. The procedure involves determining which matrices may be output during the same pass of scratch data set two by (a) comparing the assigned instruction output data set number, and (b) type of matrix being output. Output matrix positions eight and ten, if non-blank, are always output on the first pass. Output matrix positions six, seven, nine, eleven, and twelve may require from one to five passes of scratch data set two, recognizing the best and worst possible cases. In general, OUTMAT may only output one matrix per pass on a given instruction output data set with the exception of output matrix positions eight and ten which are always output on the first pass regardless of their instruction output data set numbers.

For example, given the following instruction output data set assignments by the FORMAT II System (all output matrix positions referenced are non-blank):

Output Matrix Position	Format Assigned Instruction Output Data Set
6	4
7	8
8	3
9	3
10	8
11	4
12	3

OUTMAT would release all the requested matrices (6-12) to the FORMAT II System in two passes of scratch data set two as indicated below.

PASS 1 - 6, 7, 8, 9, 10
PASS 2 - 11, 12

Output Matrix Positions 6, 7 and 9 may be output concurrently on pass one since they are to be located on different data sets. Positions eight and ten will always be output on pass one. Since positions 11 and 12 are to be located on different data sets, they may be output on the same pass.

If a matrix is less than 50% dense, the compressed column record format is invoked.

SECTION IV

OPERATIONAL CONDITIONS

A. IMPLEMENTATION

1. Direct Machine Control

Under direct machine control the only changes required for implementation on any system are contained in one deck, subroutine MRES. The implementation operations involved are explained in detail in Appendix IX. In general, the information which must be supplied consists of defining system parameters; such as system input unit, system output unit, size of blank common work area, and limiting size of matrix capability; and assigning MAGIC III System functions to the available external storage units.

Under direct machine control the Structural Generative System has been inserted as a normal user module with the same origin and accessibility as any other user module.

Operation of the Structural Generative System requires the common area to be at least 13000_{10} storages and the number of external storage units to be at least eight. Both of these facts must be inserted into MRES at implementation time.

2. SUBSYS Control

Implementation upon an IBM 7090/94 requires an improvement of the loading capabilities of IBSYS. The software package selected is SUBSYS, developed by Westinghous Corporation. A software package was selected in deference to multiple passes at IBJOB due to the inflexibilities of IBLDR under IBJOB. For example, IBLDR requires the use of at least three tape drives to load each portion, thereby removing units from use by FORMAT II. Also, data would be inserted in the middle of program deck and printed output would be interspersed with IBJOB Processor Output. The most decisive advantage, however, was the saving of load time under SUBSYS. Normal load time under IBLDR for the complete program is approximately eight minutes on a 7090, whereas under SUBSYS control the program is placed into core and execution started with a load time of fifteen to twenty seconds. A more detailed discussion of SUBSYS is given in Appendix X.

APPENDIX I

OVERLAY STRUCTURE

The Overlay structure is divided into two sections. The first section is the revised FORMAT II Overlay Structure (Reference 2) and the second section is the Structural System Overlay structure.

SECTION I

REVISED FORMAT II OVERLAY

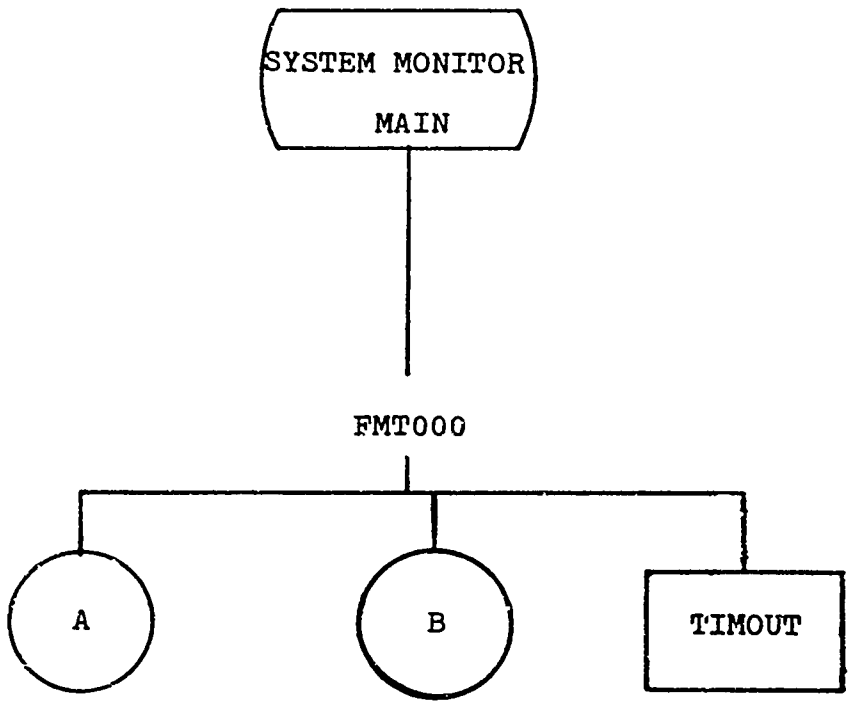


FIGURE I.1 SYSTEM MONITOR

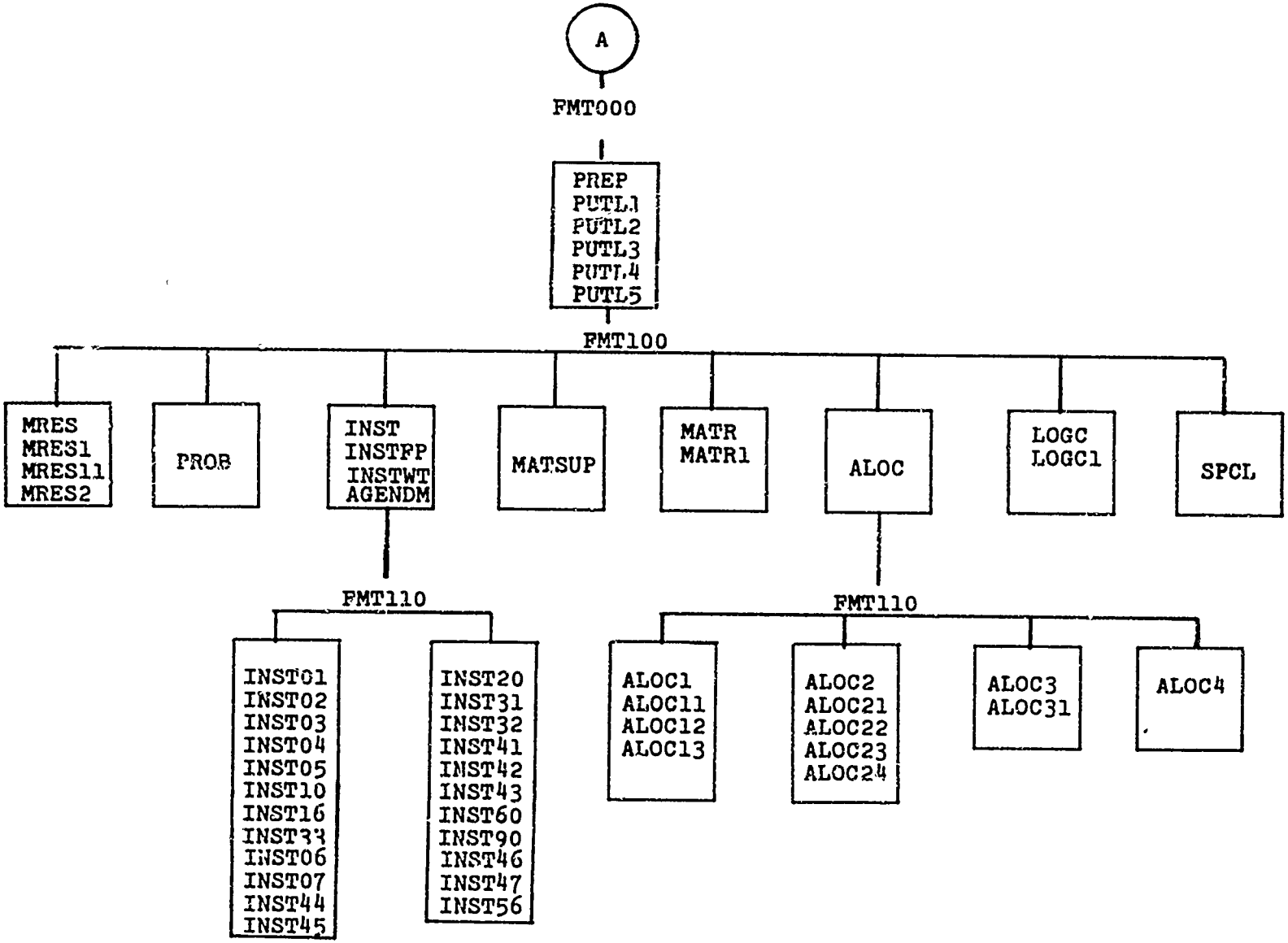


FIGURE I.2 PREPROCESSOR MONITOR

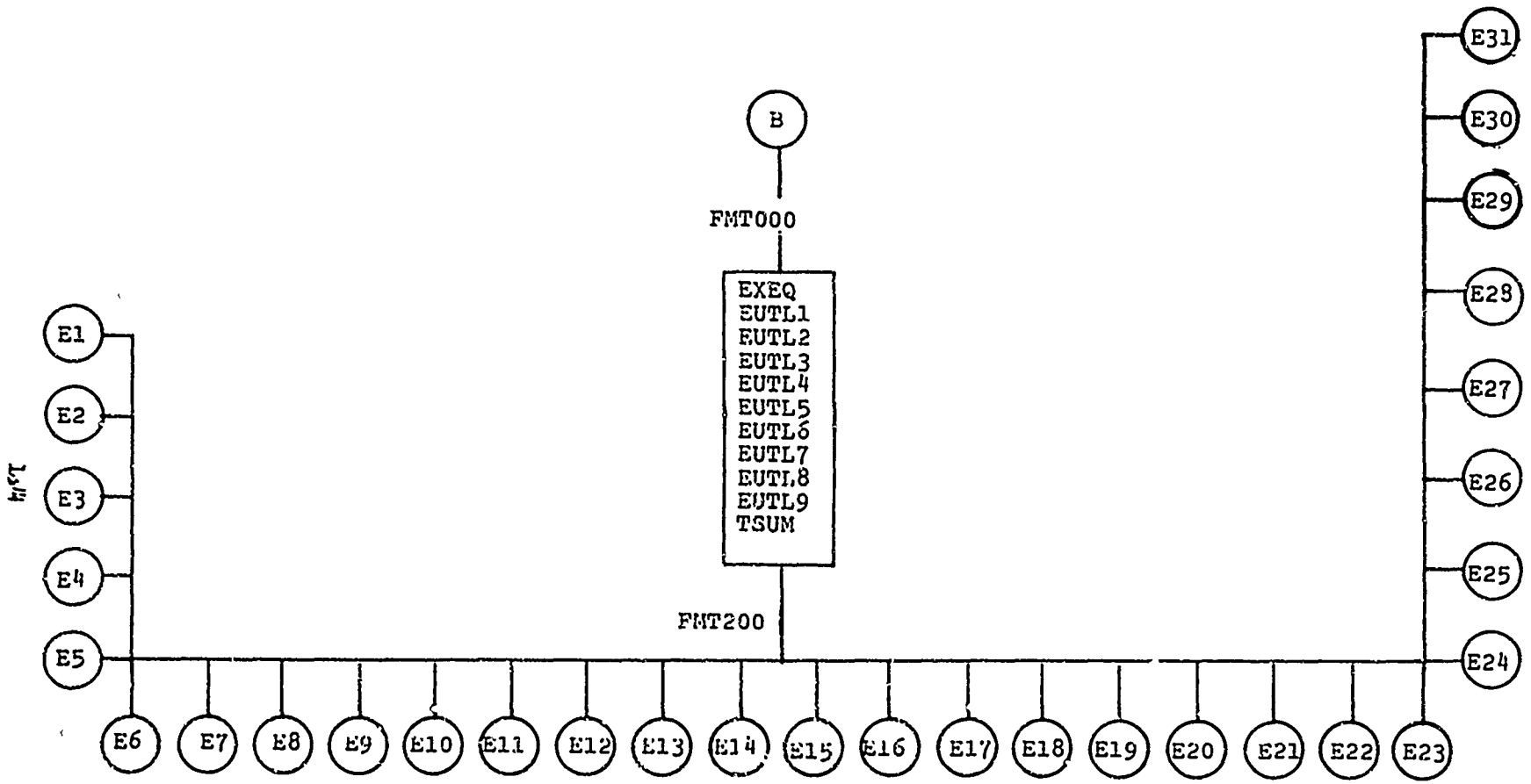


FIGURE I.3 EXECUTION PHASE

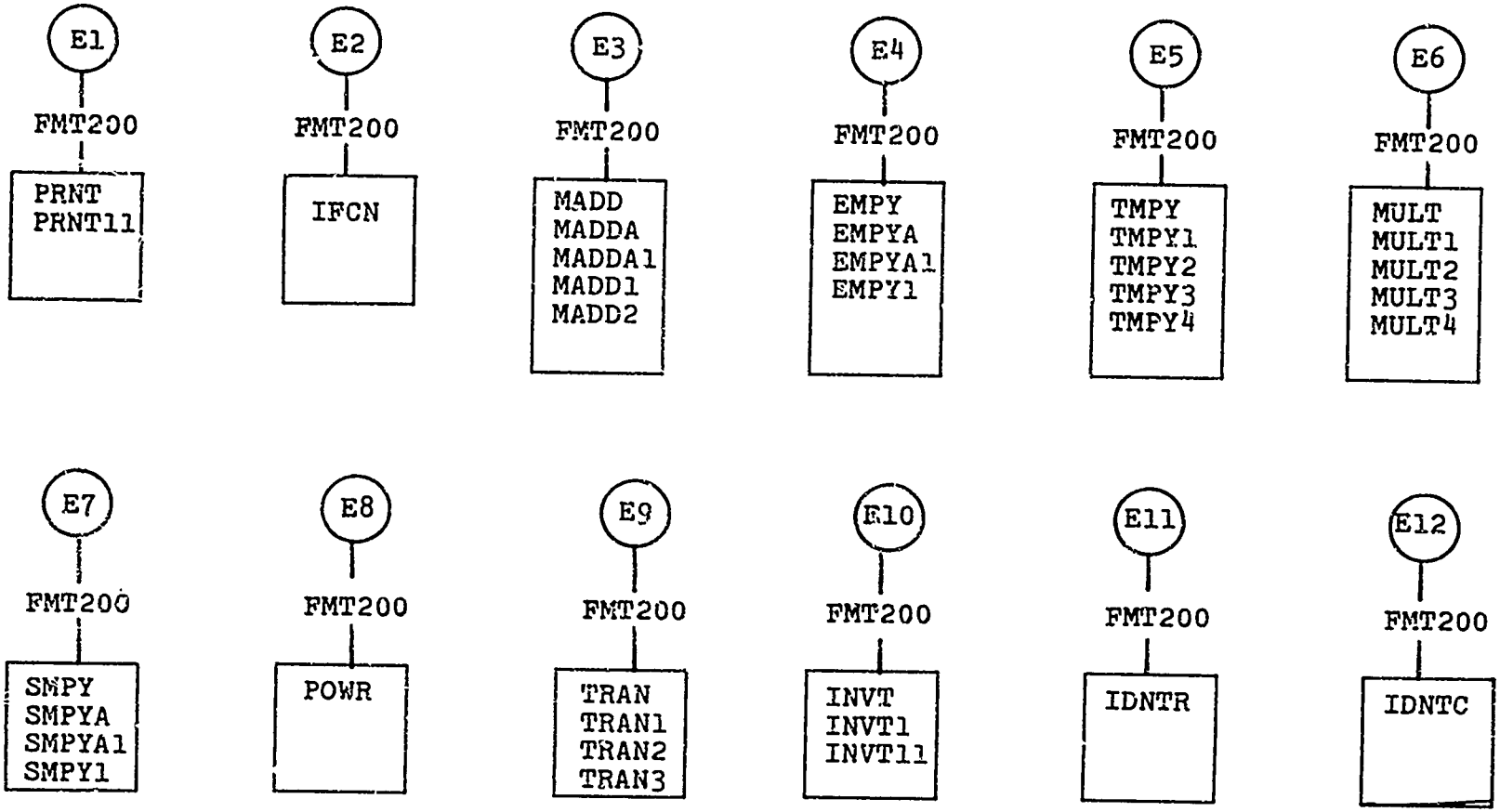


FIGURE I.4 EXECUTION PHASE, Continued

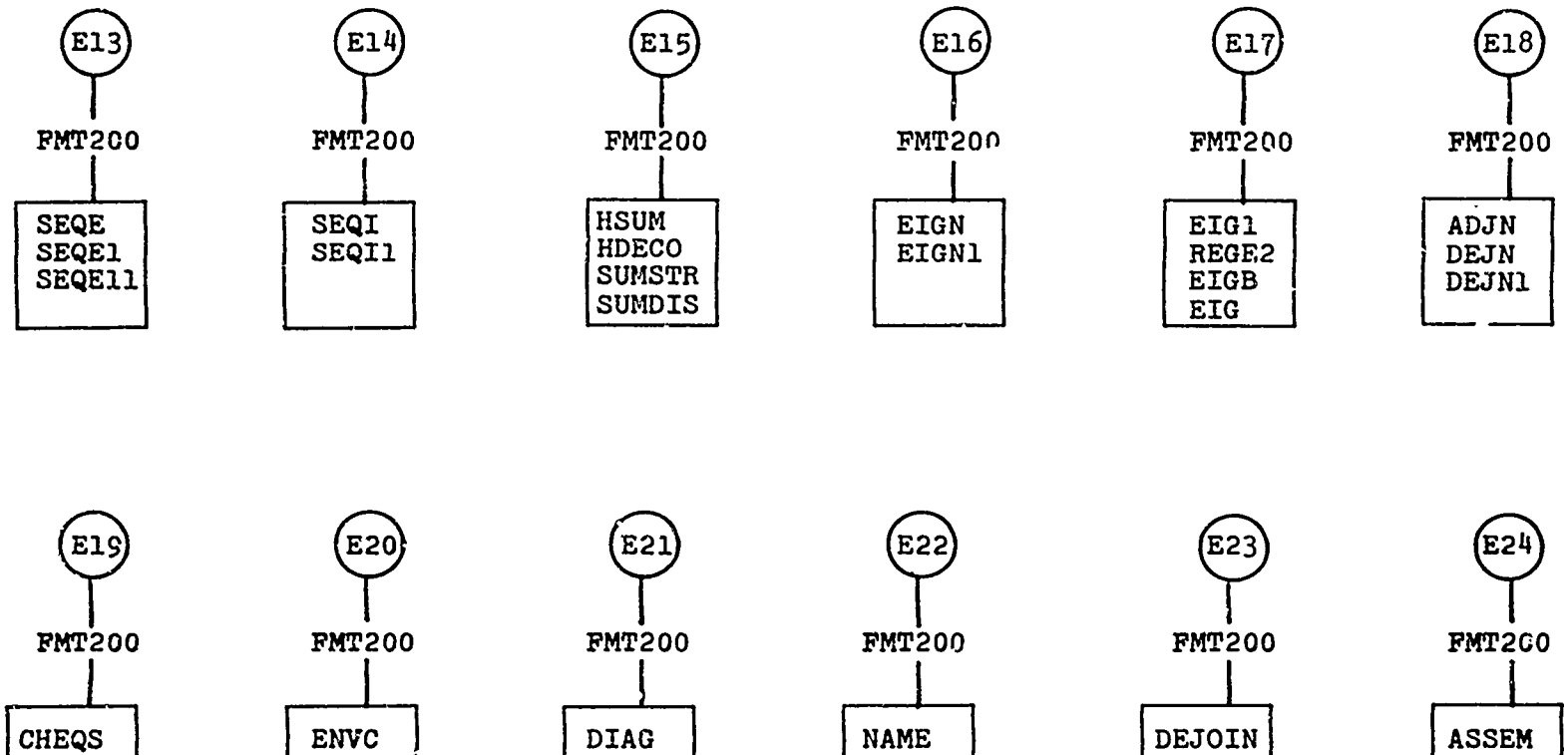


FIGURE I.5 EXECUTION PHASE, Continued

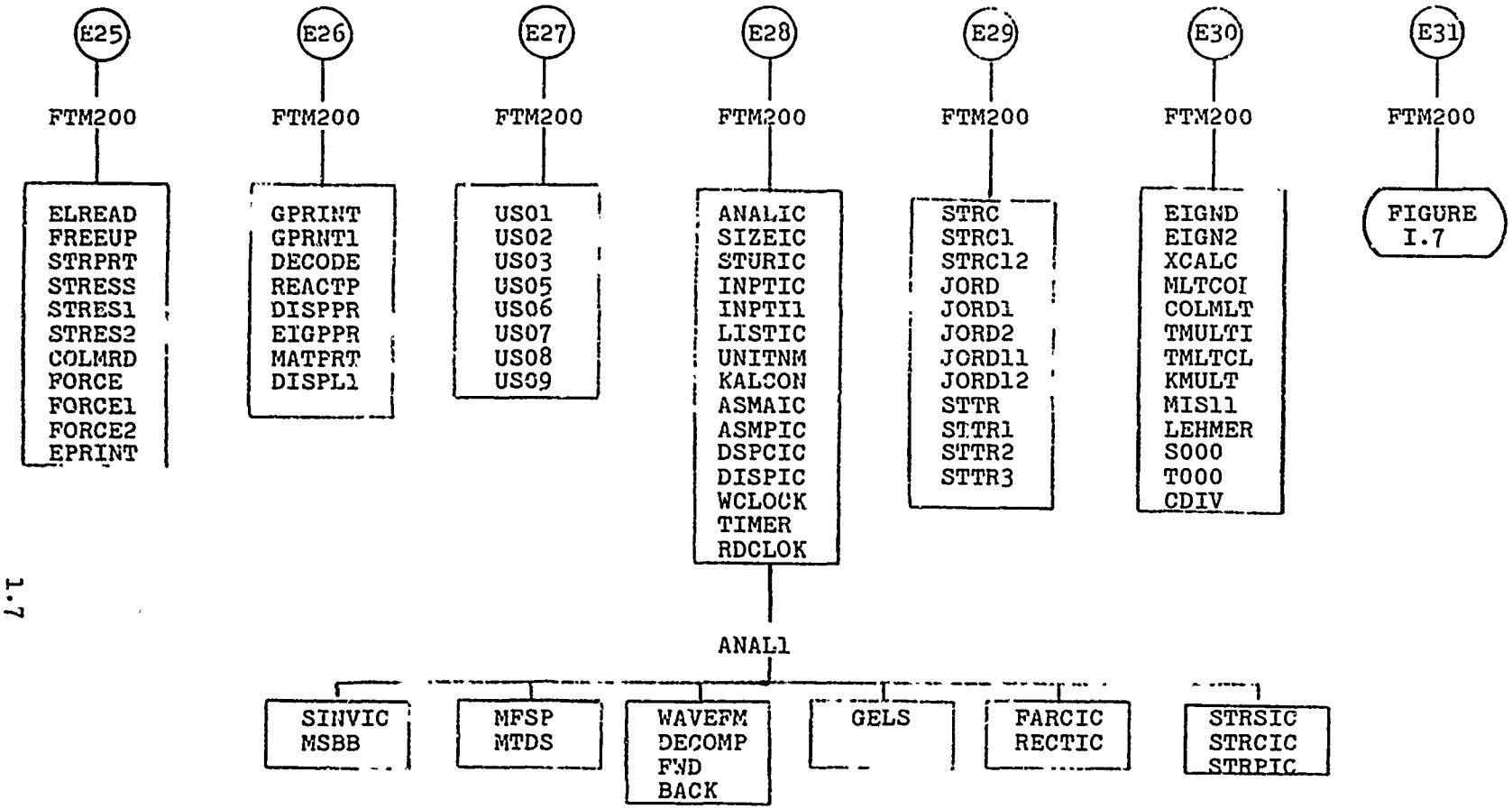


FIGURE I.6 EXECUTION PHASE, continued

SECTION II

STRUCTURAL SYSTEM OVERLAY CHART

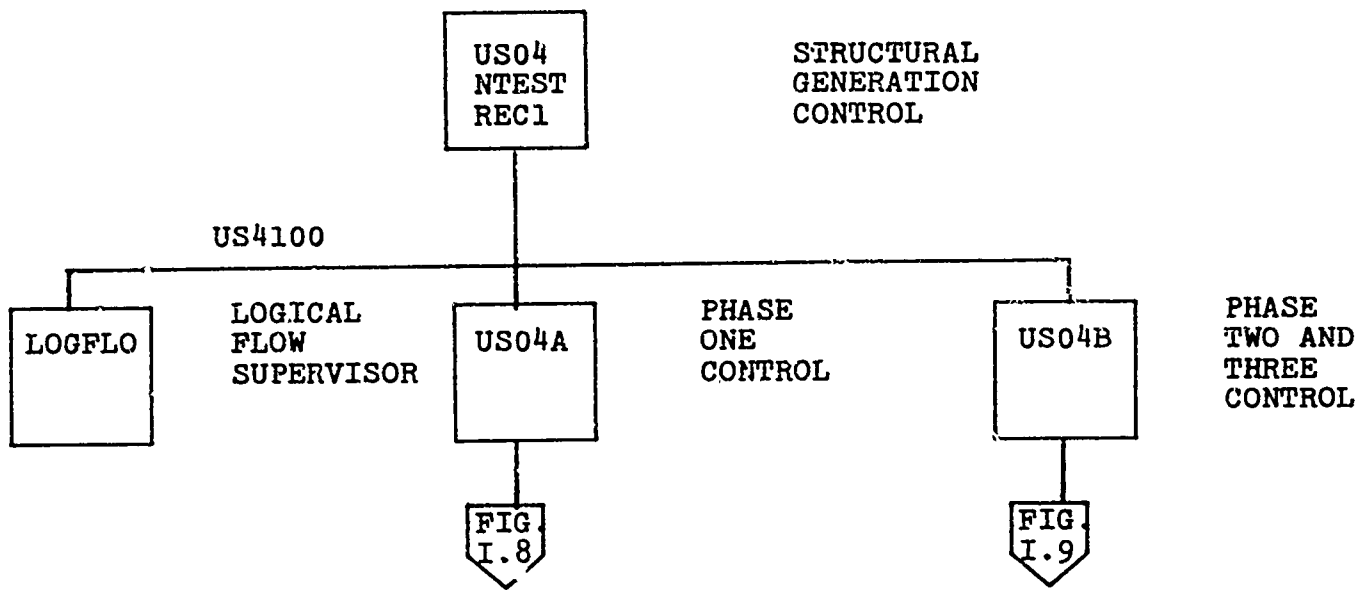


FIGURE I.7 CONTROL SECTION

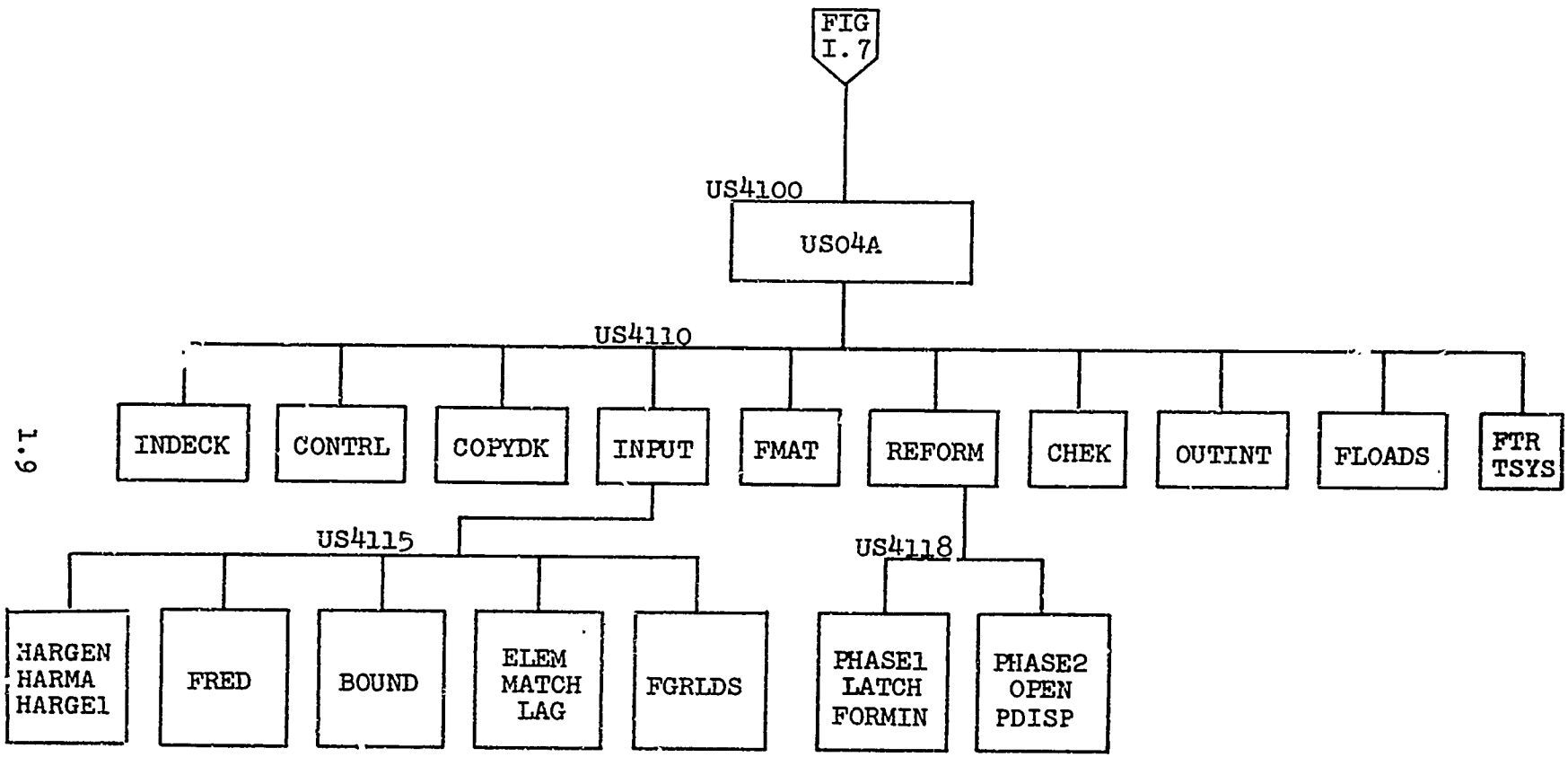


FIGURE I.3 PHASE ONE SECTION

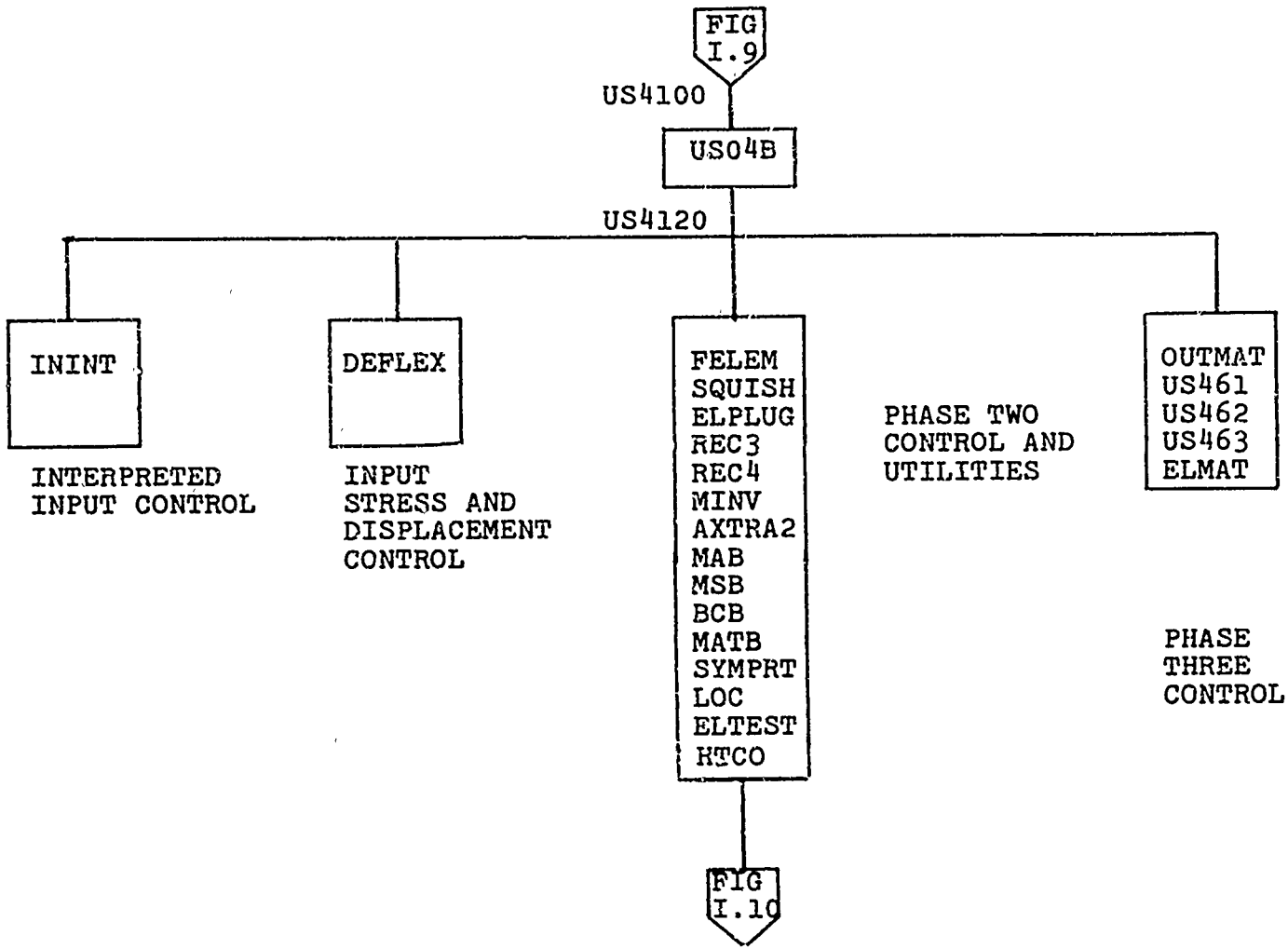


FIGURE I.9 PHASE TWO (ELEMENT GENERATION) AND PHASE THREE (OUTPUT) CONTROL

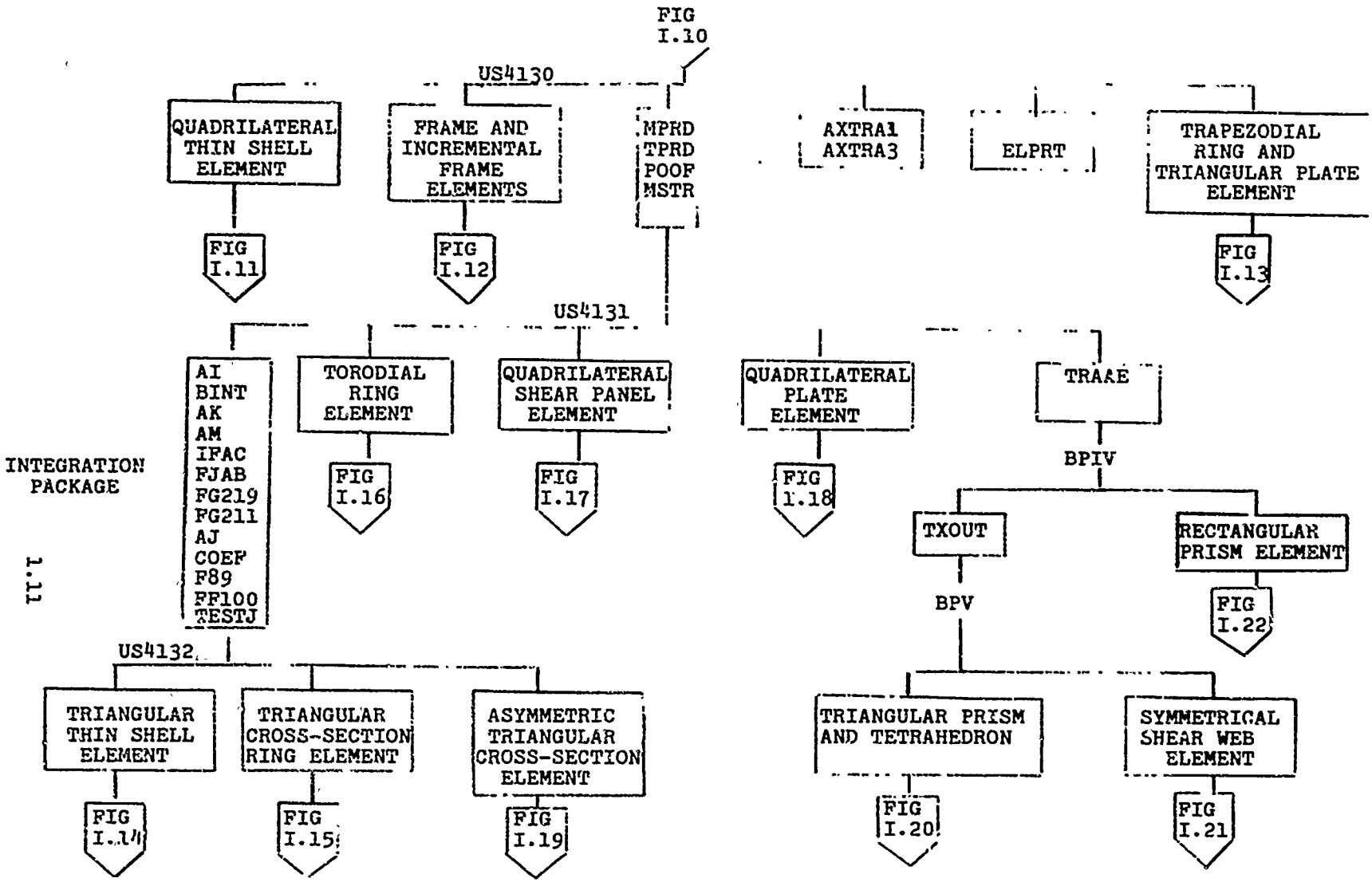


FIGURE I.10 PHASE TWO CONTROL AND UTILITY PACKAGES

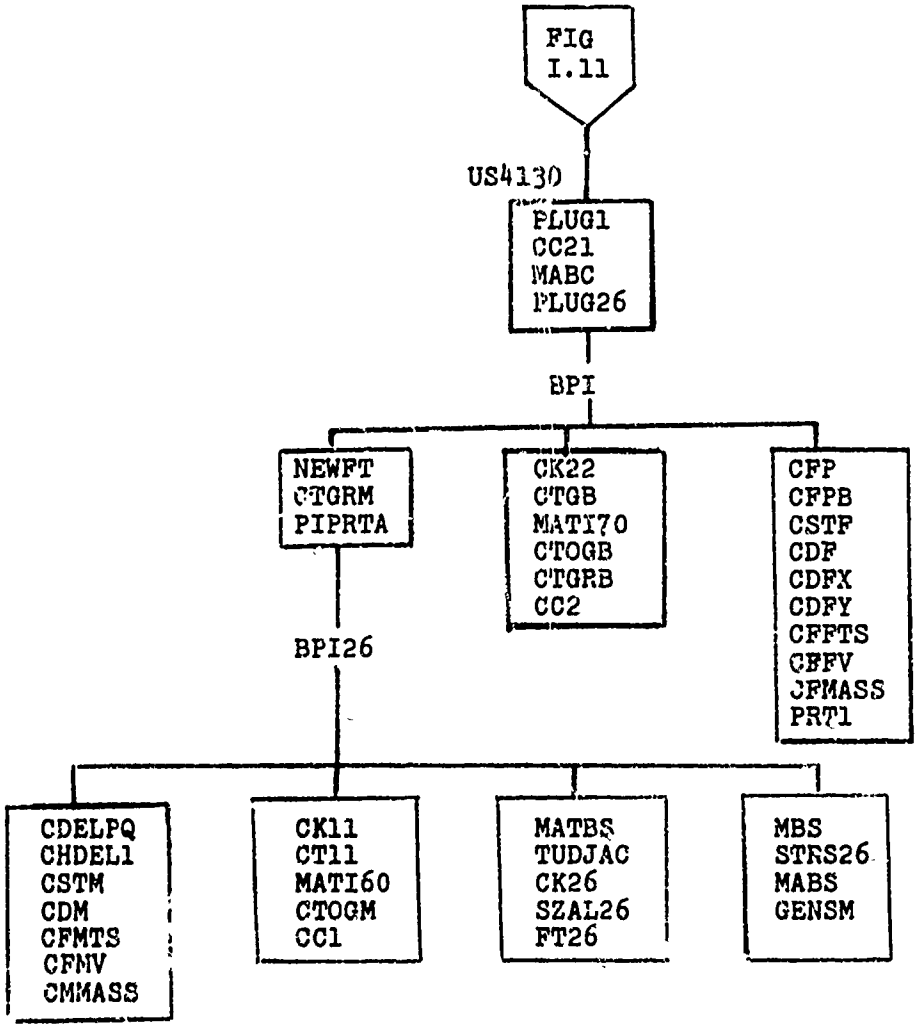


Figure I.11 Quardrilateral Thin Sehll Element and High-Aspect Ratio Quadrilateral Thin Shell Element

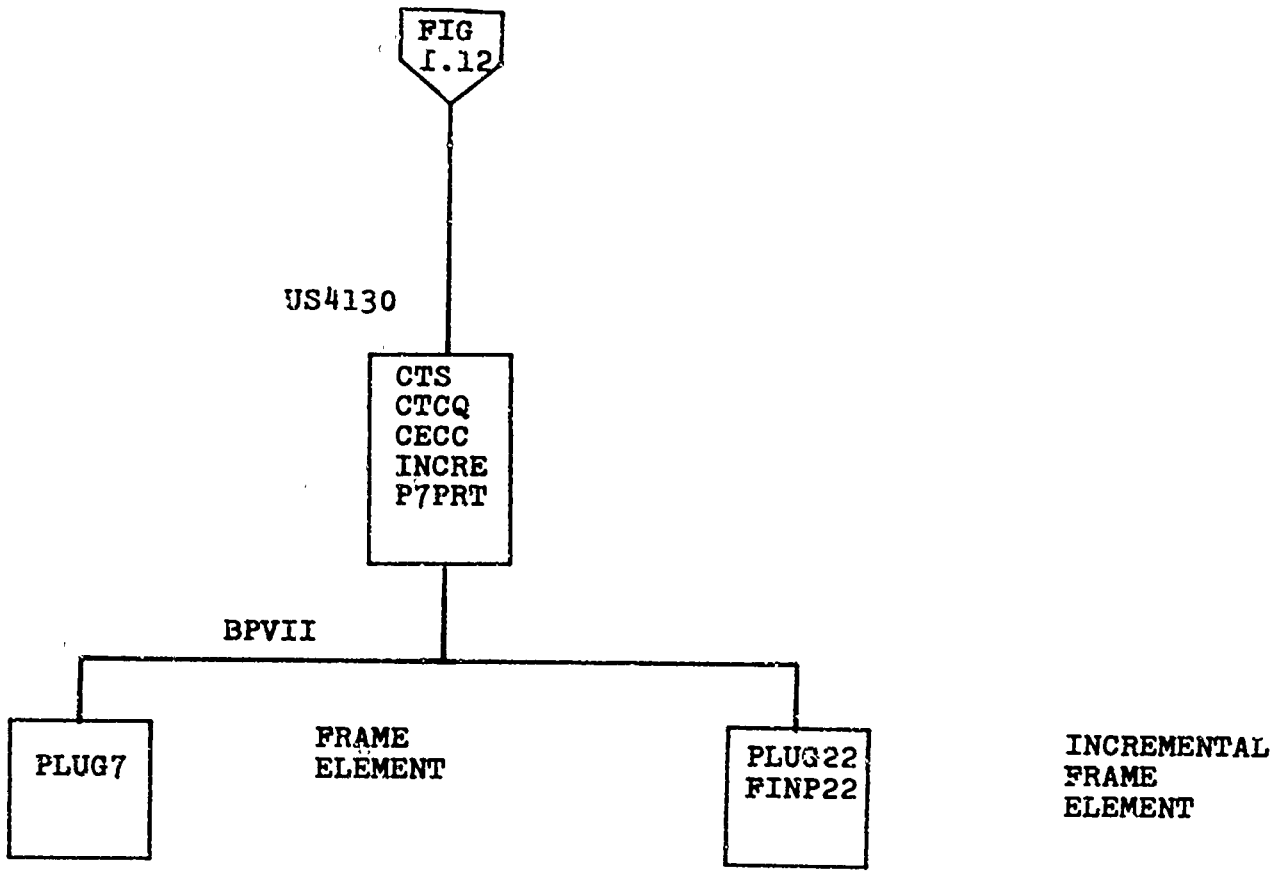


FIGURE I.12 FRAME AND INCREMENTAL FRAME ELEMENT

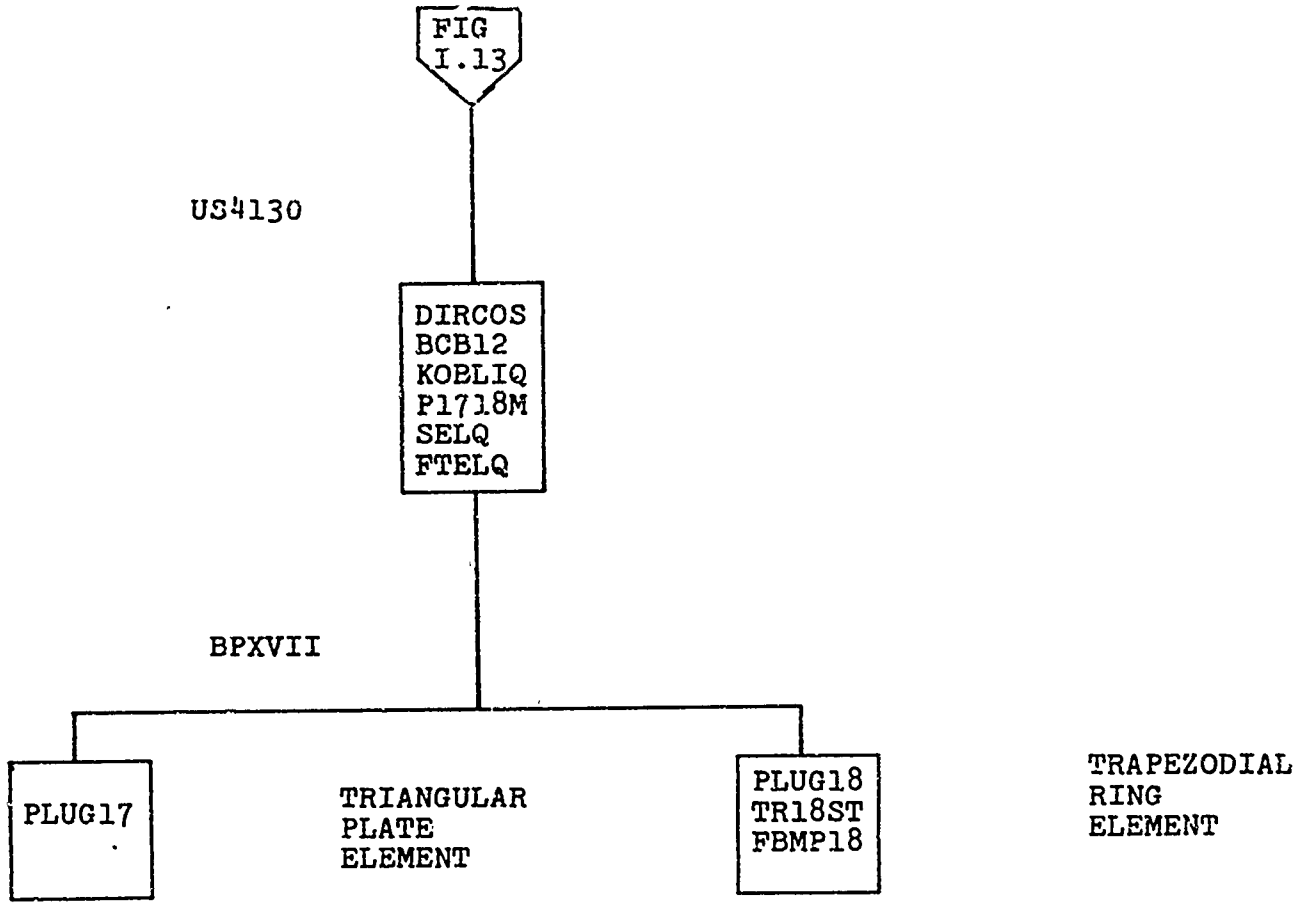


FIGURE I.13 TRAPEZODIAL RING AND TRIANGULAR PLATE ELEMENTS

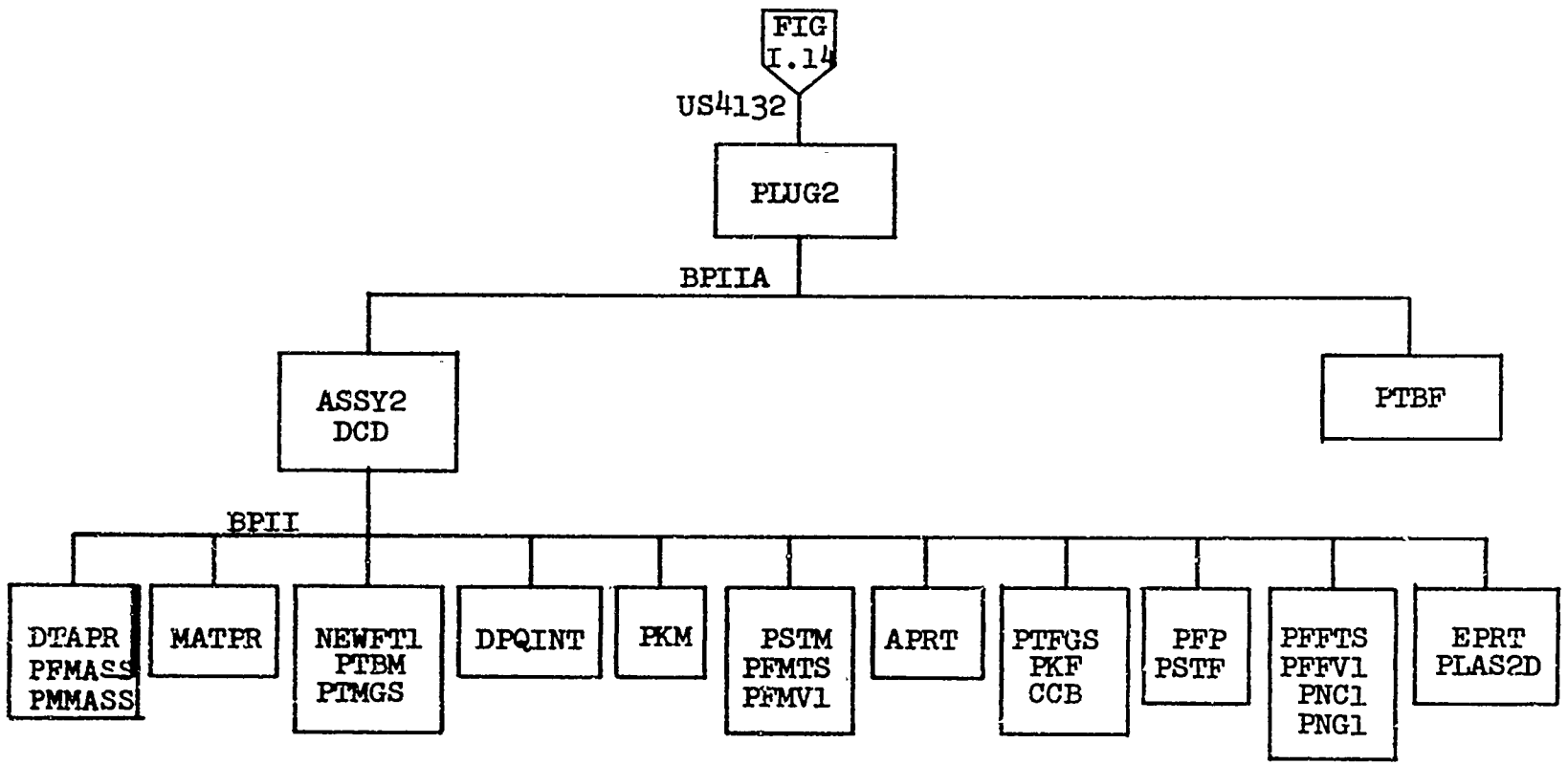


FIGURE I.14 TRIANGULAR THIN SHELL ELEMENT

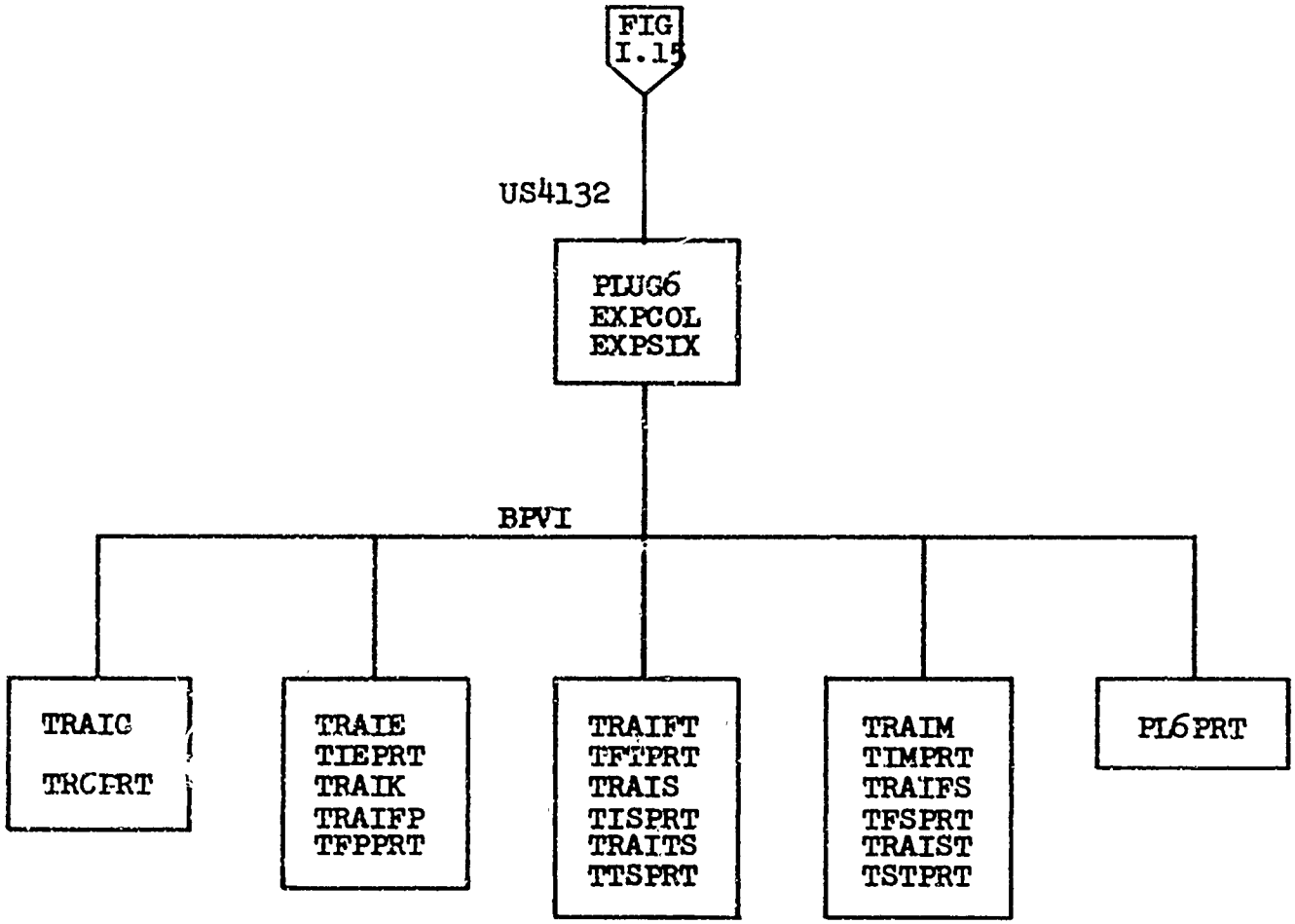


FIGURE I.15 TRIANGULAR CROSS SECTION RING ELEMENT

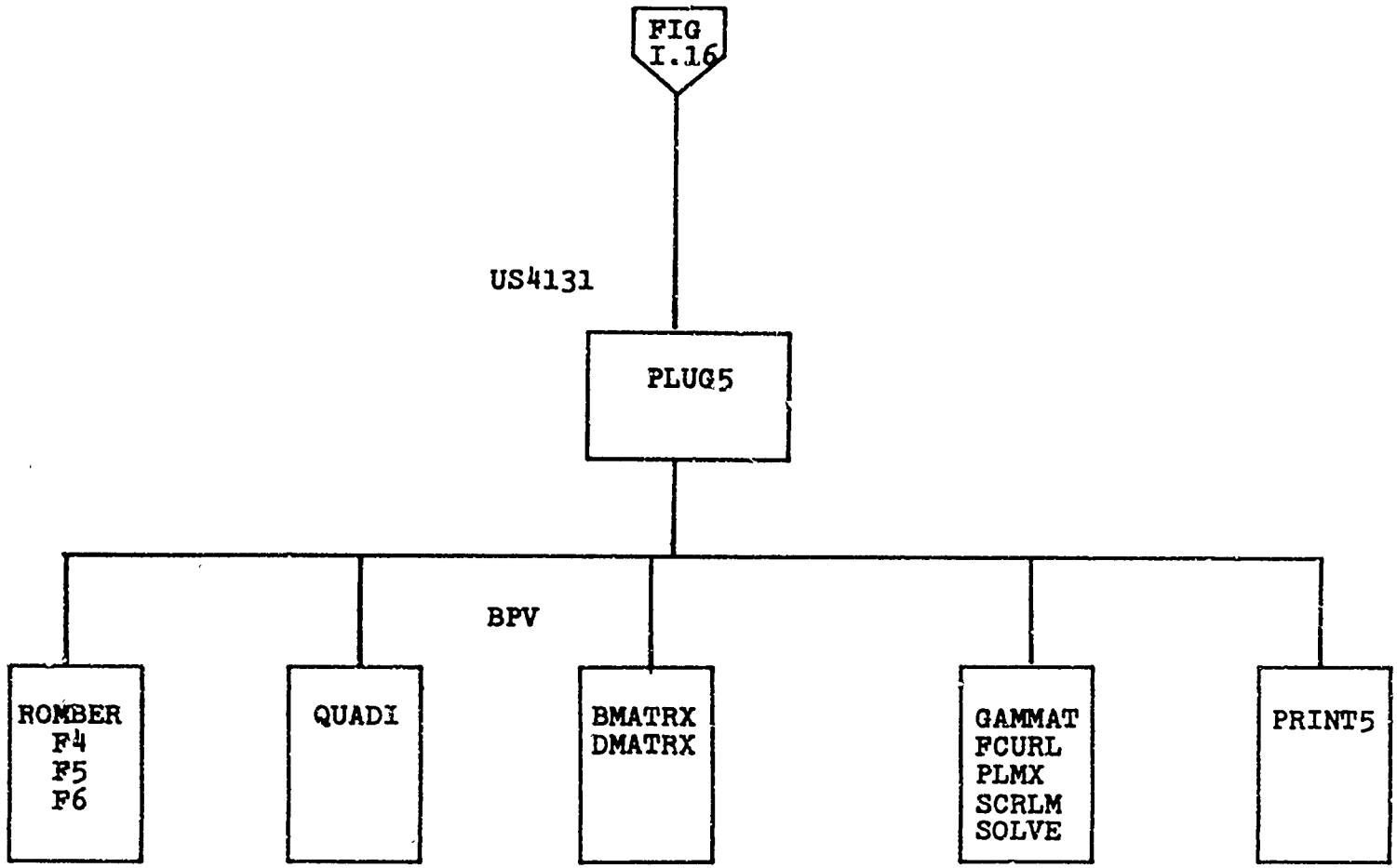


FIGURE I.16 TOROIDAL RING ELEMENT

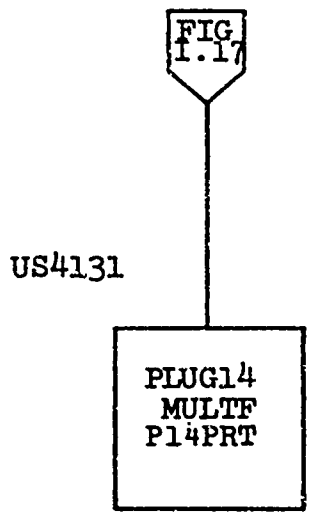


FIGURE I.17 QUADRILATER SHEAR PANEL ELEMENT

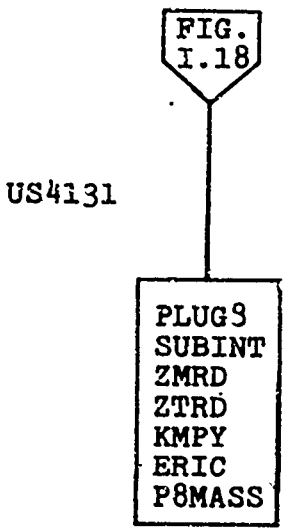


FIGURE I.18 QUADRILATERAL PLATE ELEMENT

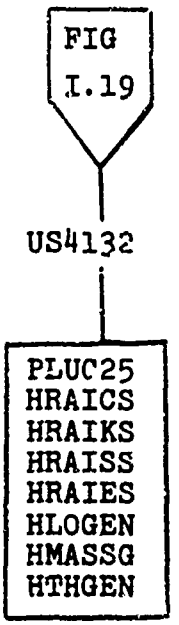


Figure I.19 Asymmetric Triangular Cross-Section Ring Element

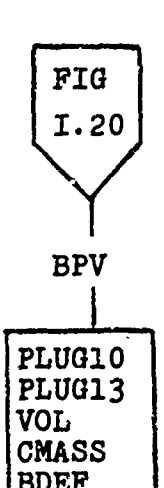


Figure I.20 Triangular Prism Element and Tetrahedron Element

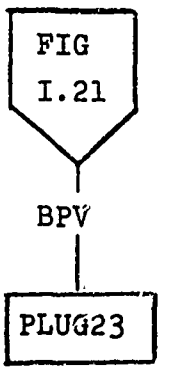


Figure I.21 Symmetrical Shear Web Element

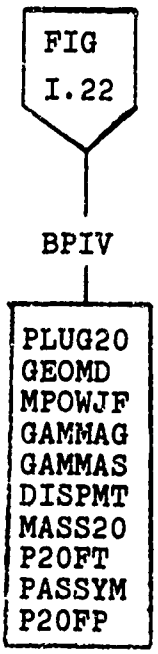
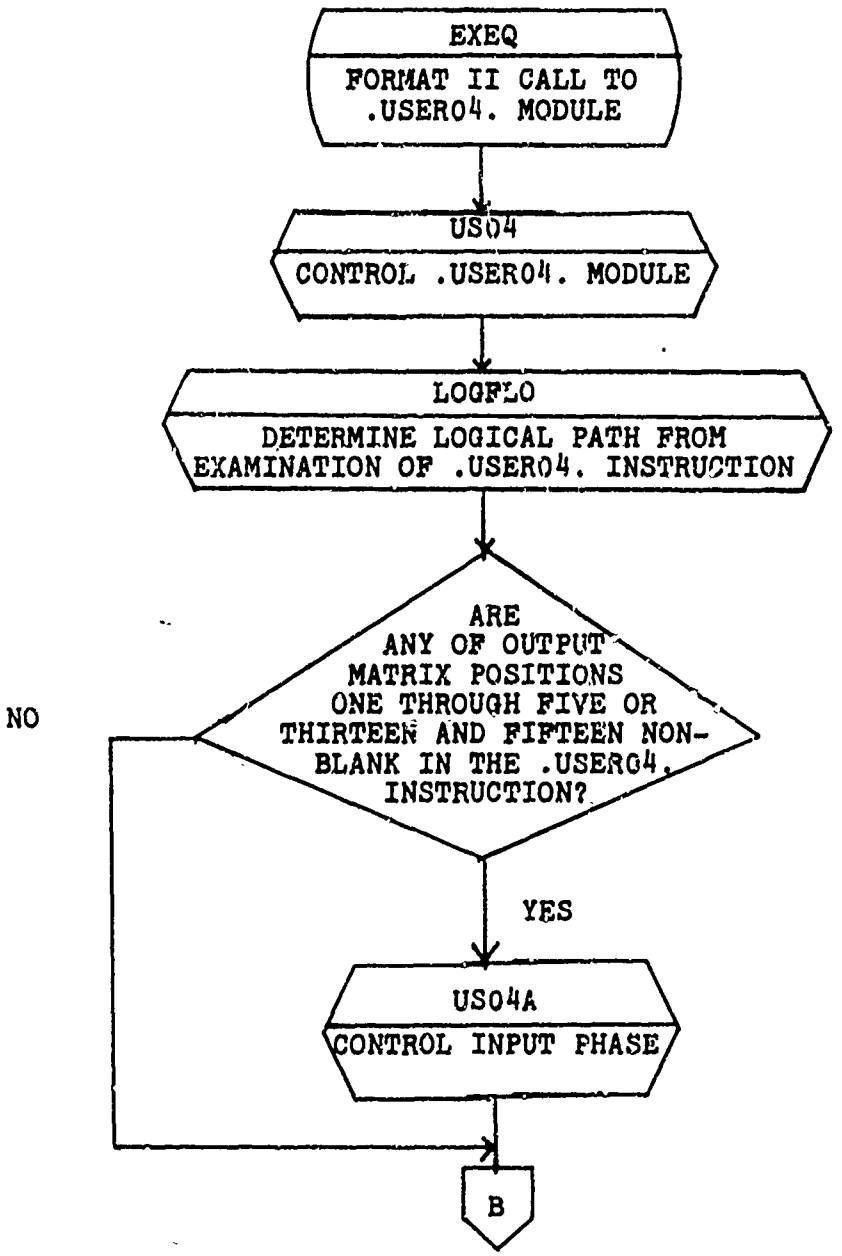


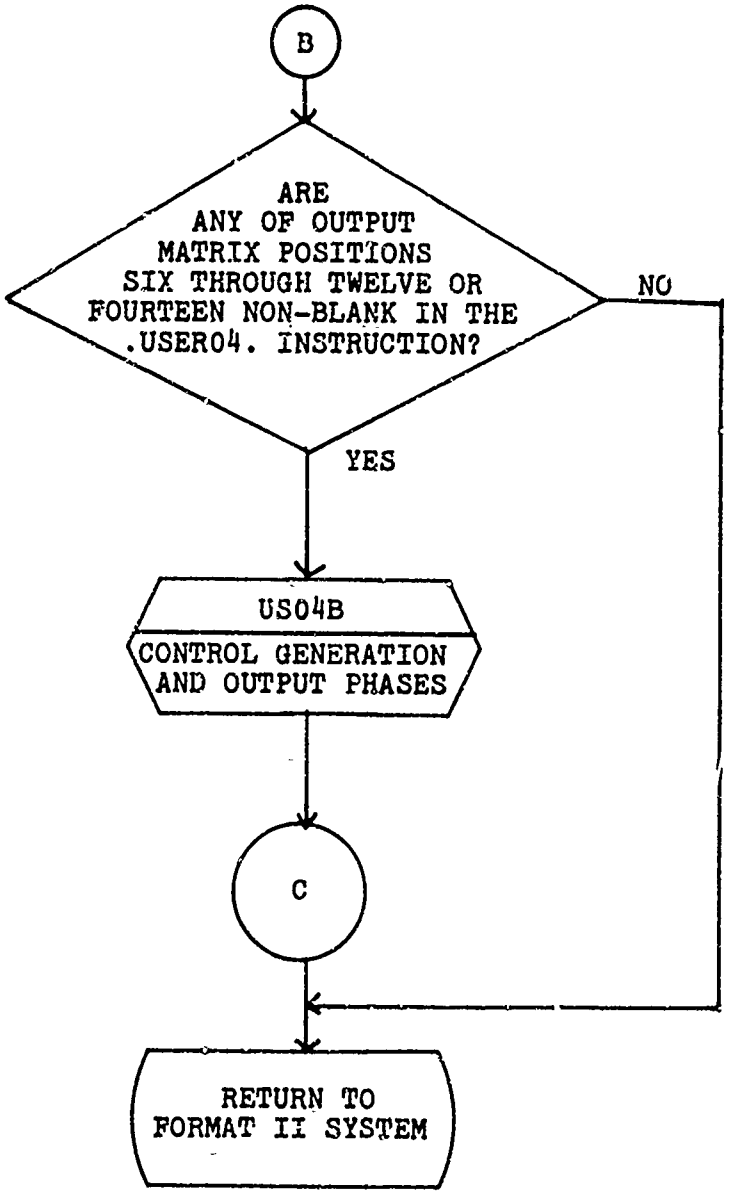
Figure I.22 Rectangular Prism Element

APPENDIX II

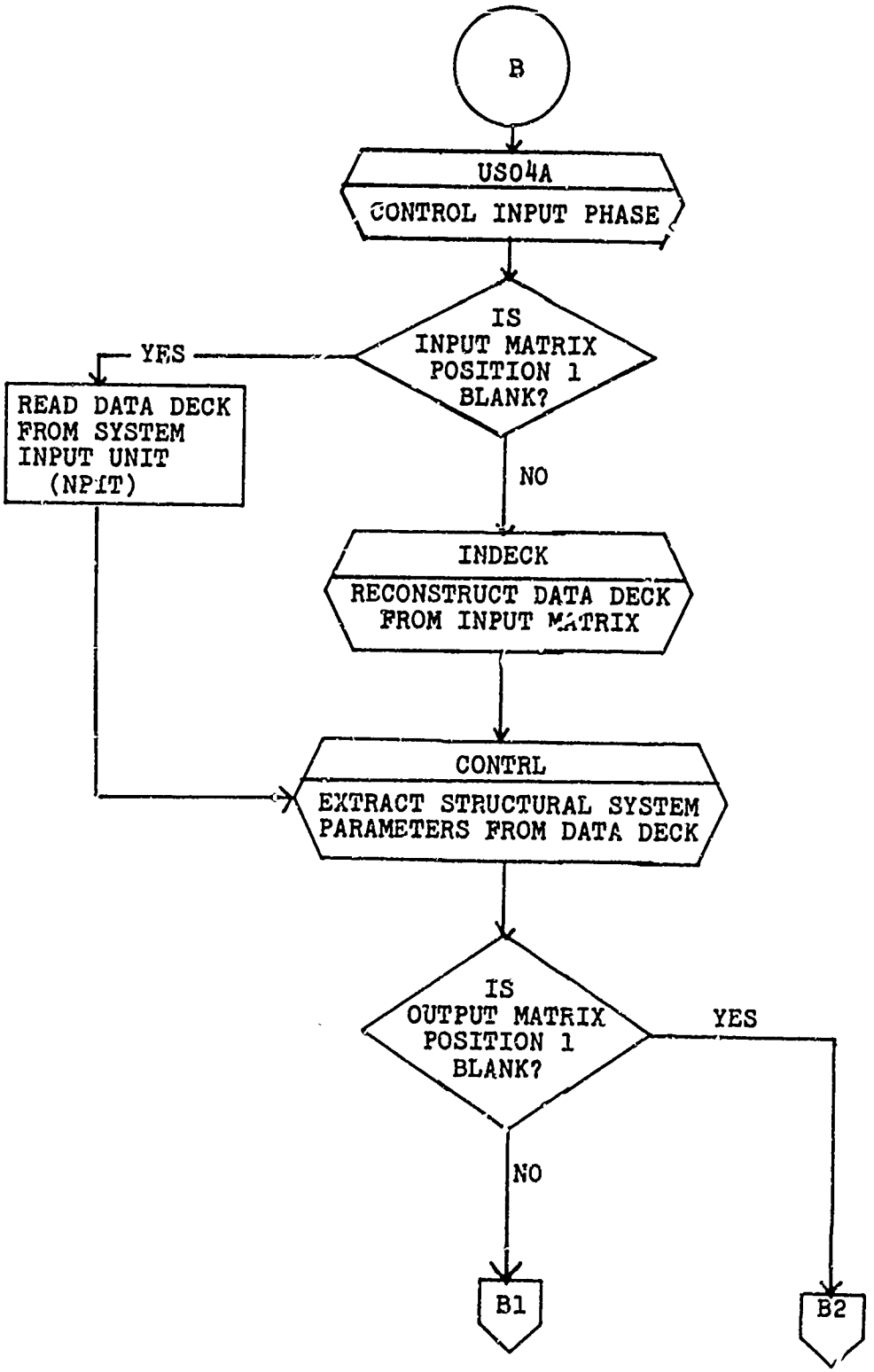
LOGICAL FLOWCHARTS

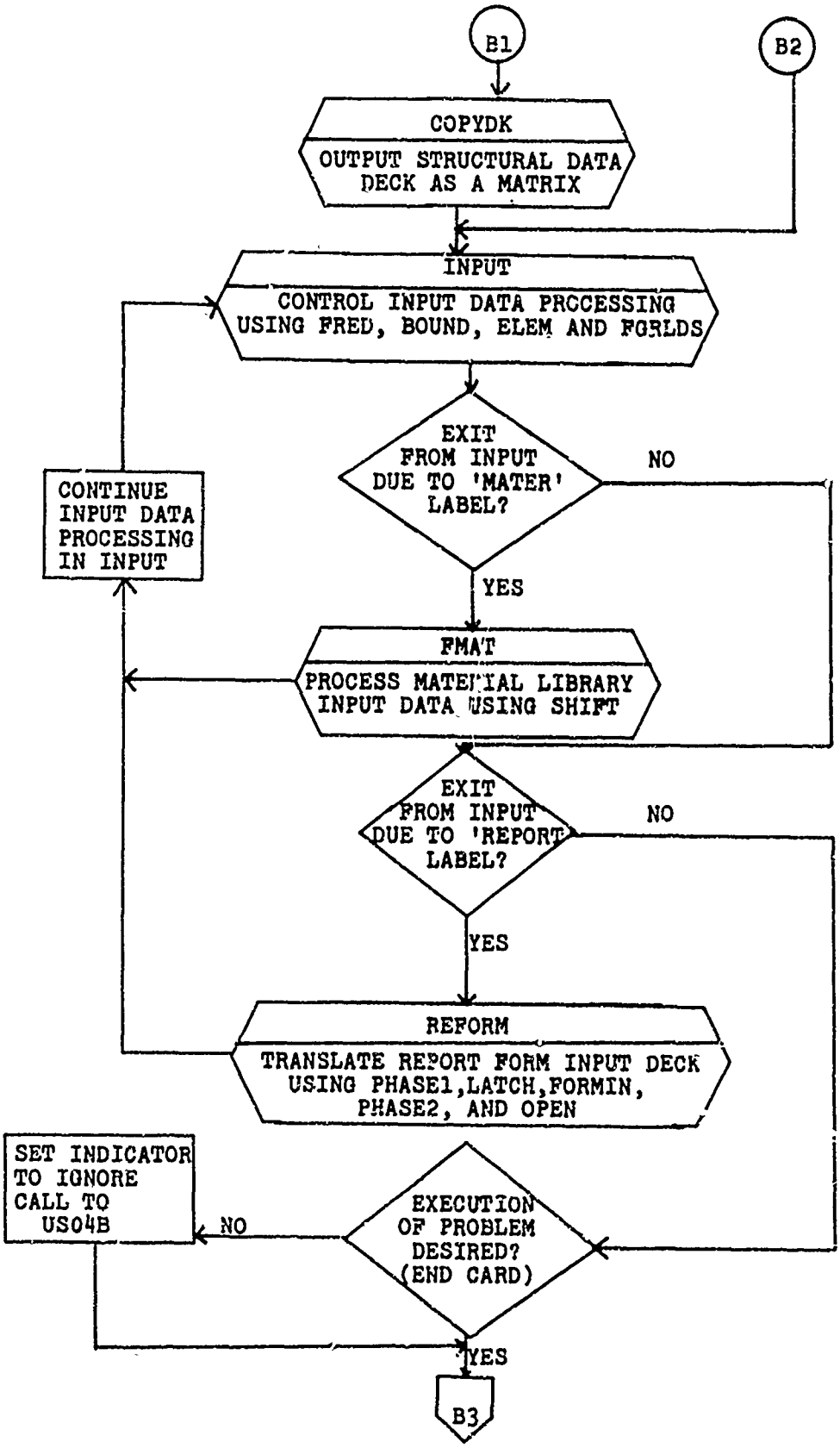
A. STRUCTURAL GENERATIVE SYSTEM LOGIC FLOW

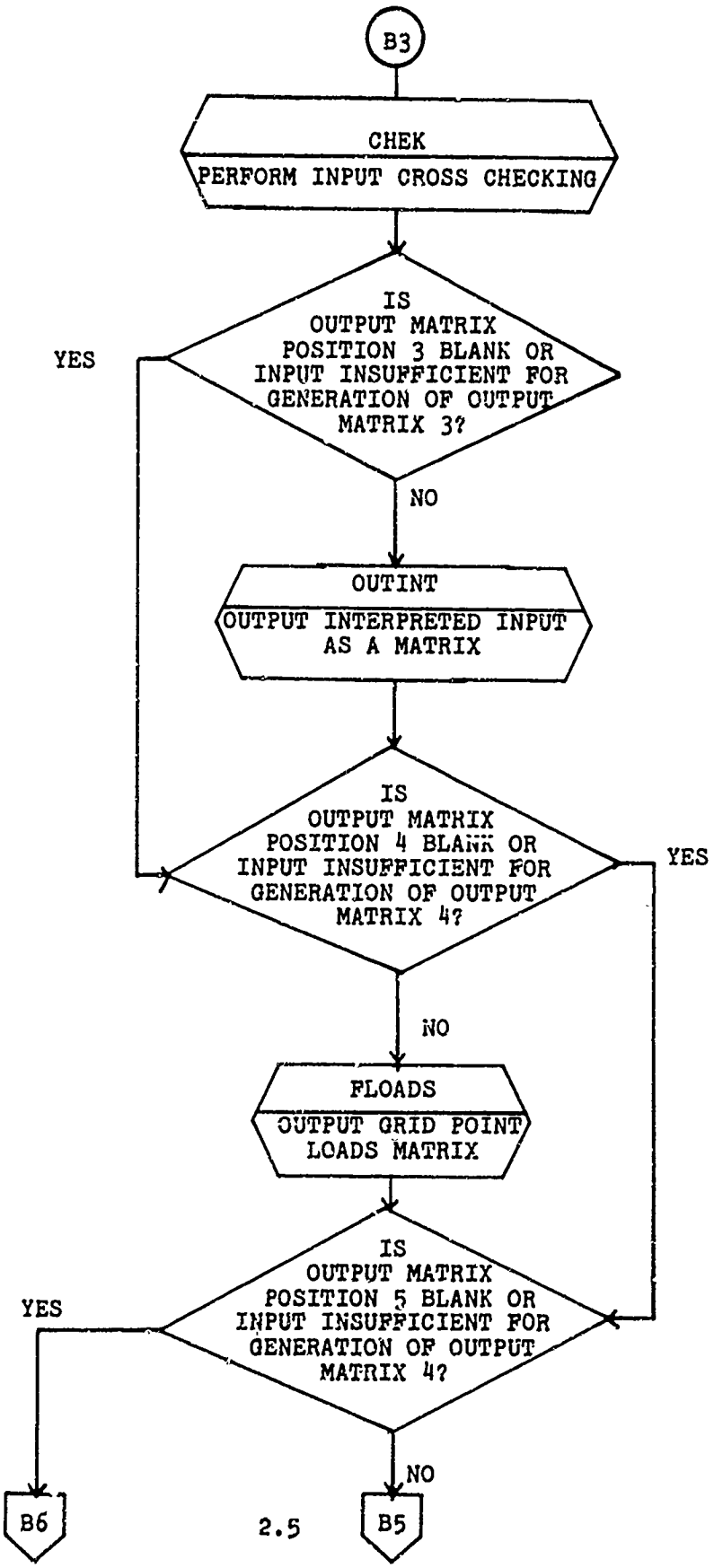


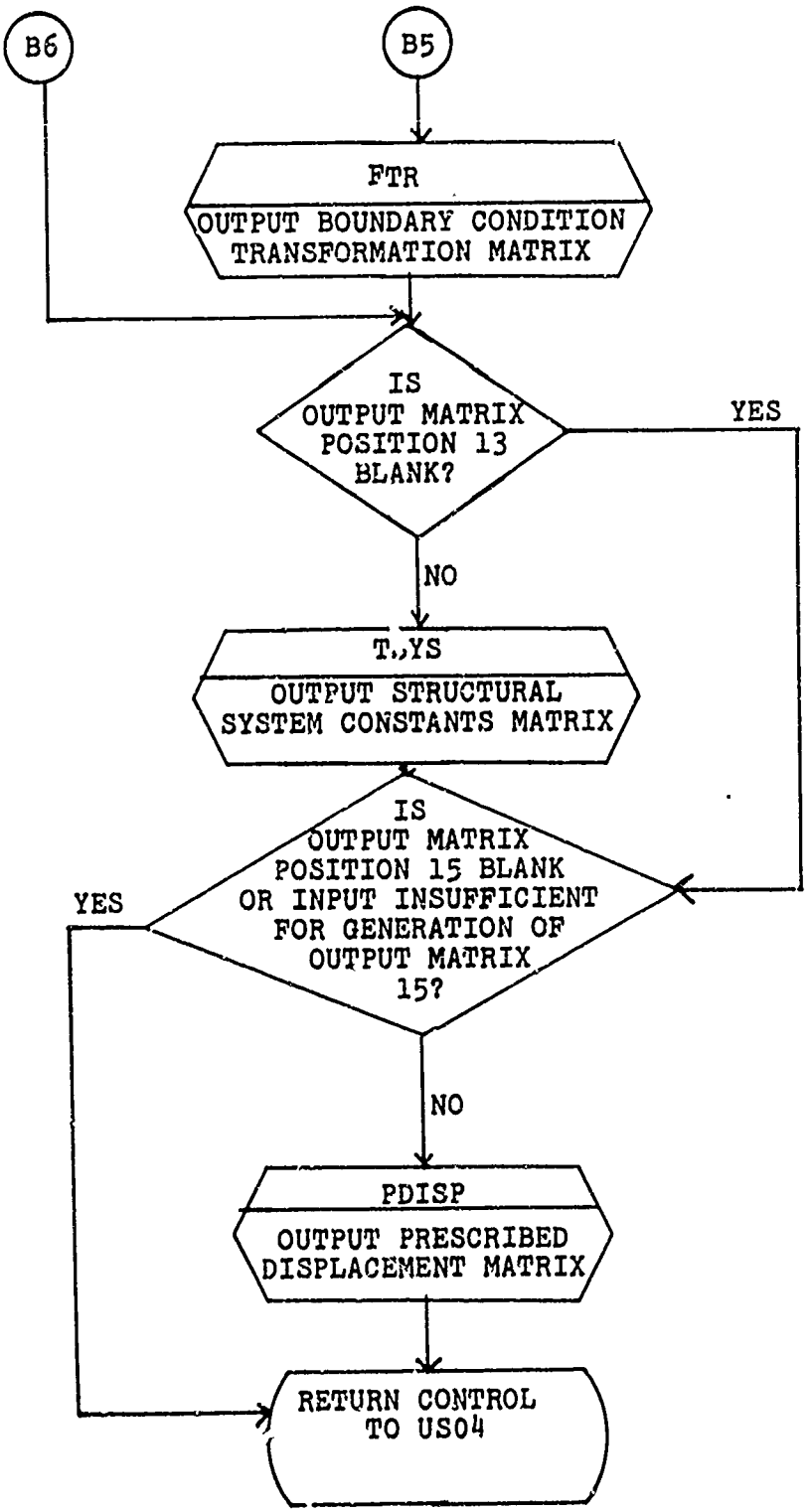

B
ARE ANY OF OUTPUT MATRIX POSITIONS SIX THROUGH TWELVE OR FOURTEEN NON-BLANK IN THE .USER04. INSTRUCTION?
NO
YES
US04B
CONTROL GENERATION AND OUTPUT PHASES
C
RETURN TO FORMAT II SYSTEM

B. INPUT PHASE LOGIC FLOW

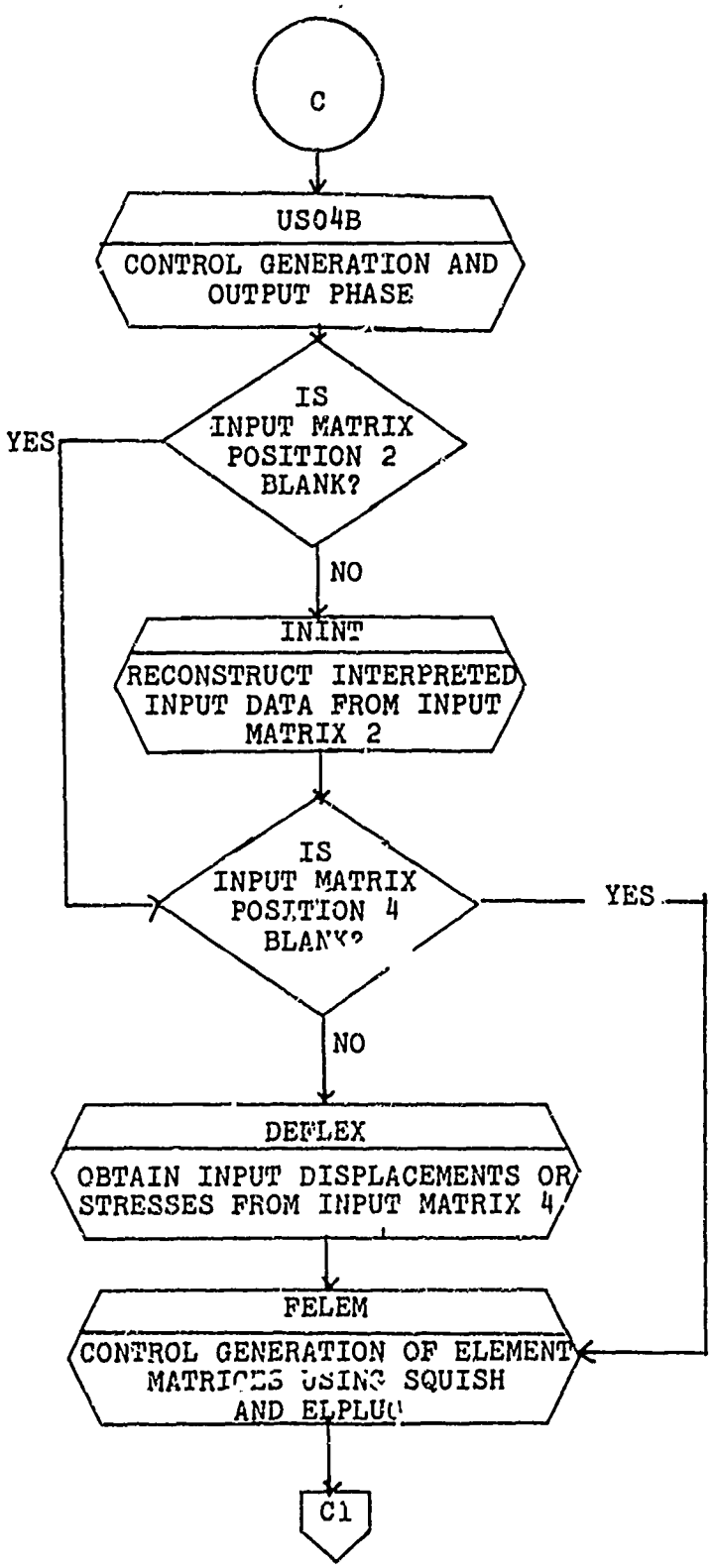


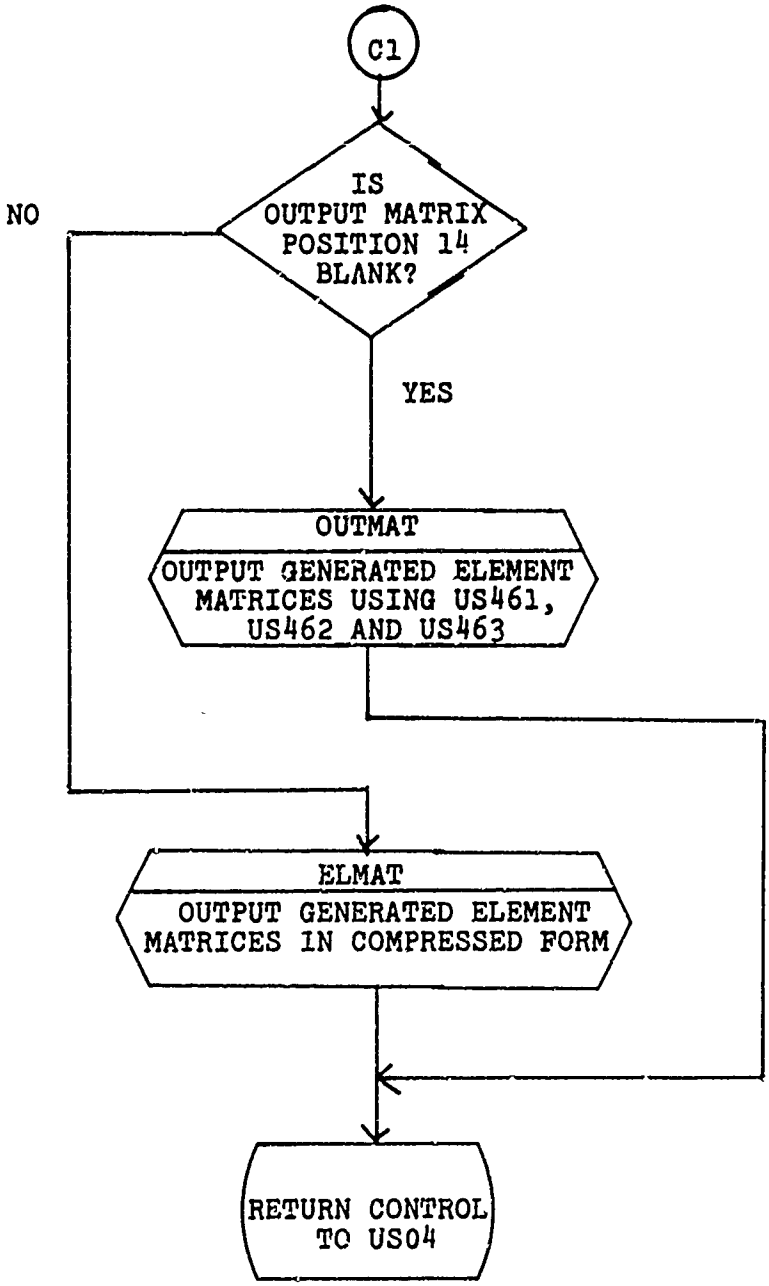

B1
B2
COPYDK
OUTPUT STRUCTURAL DATA DECK AS A MATRIX
INPUT
CONTROL INPUT DATA PROCESSING USING FRED, BOUND, ELEM AND FGRLDS
EXIT FROM INPUT DUE TO 'MATER' LABEL?
NO
YES
CONTINUE INPUT DATA PROCESSING IN INPUT
FMAT
PROCESS MATERIAL LIBRARY INPUT DATA USING SHIFT
EXIT FROM INPUT DUE TO 'REPORT' LABEL?
NO
YES
REFORM
TRANSLATE REPORT FORM INPUT DECK USING PHASE1,LATCH,FORMIN, PHASE2, AND OPEN
SET INDICATOR TO IGNORE CALL TO US04B
NO
EXECUTION OF PROBLEM DESIRED? (END CARD)
YES
B3


B3
CHEK
PERFORM INPUT CROSS CHECKING
IS OUTPUT MATRIX POSITION 3 BLANK OR INPUT INSUFFICIENT FOR GENERATION OF OUTPUT MATRIX 3?
YES
NO
OUTINT
OUTPUT INTERPRETED INPUT AS A MATRIX
IS OUTPUT MATRIX POSITION 4 BLANK OR INPUT INSUFFICIENT FOR GENERATION OF OUTPUT MATRIX 4?
YES
NO
FLOADS
OUTPUT GRID POINT LOADS MATRIX
IS OUTPUT MATRIX POSITION 5 BLANK OR INPUT INSUFFICIENT FOR GENERATION OF OUTPUT MATRIX 4?
YES
NO
B6
B5


B6
B5
FTR
OUTPUT BOUNDARY CONDITION TRANSFORMATION MATRIX
IS OUTPUT MATRIX POSITION 13 BLANK?
YES
NO
T.,YS
OUTPUT STRUCTURAL SYSTEM CONSTANTS MATRIX
IS OUTPUT MATRIX POSITION 15 BLANK OR INPUT INSUFFICIENT FOR GENERATION OF OUTPUT MATRIX 15?
YES
NO
PDISP
OUTPUT PRESCRIBED DISPLACEMENT MATRIX
RETURN CONTROL TO US04

C. GENERATION AND OUTPUT PHASE LOGIC FLOW




C1
IS OUTPUT MATRIX POSITION 14 BLANK?
NO
YES
OUTMAT
OUTPUT GENERATED ELEMENT MATRICES USING US461, US462 AND US463
ELMAT
OUTPUT GENERATED ELEMENT MATRICES IN COMPRESSED FORM
RETURN CONTROL TO US04

APPENDIX III

LIST OF STRUCTURAL SYSTEMS SUBROUTINE FUNCTIONS

Section		Page No.
A	Control and Utility Subroutines	3.2
B	Quadrilateral Thin Shell Element Subroutines	3.6
C	Frame and Incremental Frame Element Subroutines	3.8
D	Triangular Plate and Quadrilateral Plate Element Subroutines	3.9
E	Triangular Thin Shell Element Subroutines	3.10
F	Triangular Cross Section Ring Element Subroutines. . .	3.12
G	Toroidal Ring Element Subroutines	3.14
H	Quadrilateral Shear Panel Element Subroutines	3.15
I	Trapezoidal Ring Element Subroutines.	3.16
J	Rectangular Prism Element Subroutines	3.17
K	Tetrahedran Element Subroutines	3.18
L	Triangular Prism Element Subroutines	3.19
M	Symmetrical Shear Web Element Subroutines	3.20
N	Asymmetric Triangular Cross Section Ring . . . . . Element Subroutines	3.21
O	High Aspect Ration Quadrilateral Thin Shell Element Subroutines	3.22

APPENDIX III

LIST OF STRUCTURAL SYSTEM SUBROUTINE FUNCTIONS

A. CONTROL AND UTILITY SUBROUTINES

US04	Control three phases of operation of .USER04. module
NTEST	Examine matrix name for suppression code
REC1	Perform writing and reading of tape records for interpreted element input
LOGFLO	Determine logical path for .USER04. module
US04A	Control first phase (input phase) of operation of .USER04. module
INDECK	Create data deck from input deck matrix
CONTRL	Select scratch tape unit for copying structural data deck, extracting structural system information in the process
COPYDK	Create input deck matrix from data deck
INPUT	Master control subroutine for reading and storing of structural input data
FRED	Generate grid point axes transformation matrices
BOUND	Read and store boundary constraints
ELEM	Read and store element input data
MATCH	Compare a material name to an entry name in the material library
LAG	Interpolate material properties with respect to temperature
FGRLDS	Read and store grid point load conditions and load scalars
FMAT	Generate, revise and/or display material library information

SHIFT	Manipulate material library internal storage area
REFORM	Control report form input preprocessing
PHASE1	Read and store report form input data deck
LATCH	Compare an input label to list of legal input labels
FORMIN	Read and store report form table input
PHASE2	Merge data stored by PHASE1 into logical sequence for INPUT
OPEN	Control scratch tape manipulations for report form input
PDISP	Output prescribed displacements as a FORMAT matrix
CHEK	Perform input cross checking
OUTINT	Output interpreted input as a matrix
FLOADS	Output grid point load conditions and load scalars as Format matrix
FTR	Output boundary constraints as a Format matrix
TSYS	Output structural system constants as a Format matrix
USO4B	Control second and third phases (element matrix generation and output) of operation of .USER04. module
ININT	Create interpreted input from a matrix
DEFLEX	Sort and store input displacements
FELEM	Control generation of element matrices
SQUISH	Set non-generation indicators for suppressed matrices
ELPLUG	Allocate work storage for elements, read interpreted element input, select proper element and store element matrices on scratch tape in compact form

REC3	Perform writing of tape records for element control data
REC4	Perform compact writing of tape records for generated element matrices
MINV	Perform in-core matrix inversion
AXTRA2	Apply grid point axes transformation
MAB	Perform in-core matrix multiplication
MSB	Perform in-core matrix multiplication where first matrix is symmetric
BCB	Perform in-core matrix triple product of the form T^TKT where K is symmetric
MATB	Perform in-core matrix multiplication of the form A^TB
SYMPRT	Print symmetrically stored matrix
LOC	Compute single subscript index given double subscript indices
ELTEST	Compare input element control information to required element control information
MPRD	Perform generalized in-core matrix multiplication
TPRD	Perform generalized in-core matrix transpose multiplication
POOF	Expands element matrices to displacement degrees of freedom
MSTR	Change storage arrangement of a matrix
AXTRA1	Apply grid point axes transformations
AXTRA3	Apply grid point axes transformations
ELPRT	Print generated element matrices
OUTMAT	Output generated element matrices as Format matrices

US461	Write a matrix column record in compressed format
US462	Generate each elements contribution to the assembly transformation matrix
US463	Generate full column from symmetrically stored matrix
ELMAT	Output compressed element matrices as a format matrix
AI	Controls calculation procedures of triangular integration package
BINT	Perform integration by expansion of binomial theorem
AK	Calculate slope of line between two points of a triangle
AM	Calculate intercept of line between two points of a triangle
IFAC	Calculate n factorial for a given n
FJAB	Perform defined integration
F6219	Perform defined integration
F6211	Perform defined integration
AJ	Perform defined integration
COEF	Calculate binomial coefficients
F89	Perform defined integration
FF100	Perform defined integration

B. QUADRILATERAL THIN SHELL ELEMENT SUBROUTINES

PLUG1	Master control
CC21	Form intermediate stiffness matrix by summation
MABC	Perform in-core matrix triple product multiplication
NEWFT	Calculate revised thermal load formulation
CDELPQ	Calculate coordinate integrals
CHDEL1	Arrange coordinate integrals in storage
P1PRTA	Print results of coordinate and material properties calculations
CK11	Control generation of membrane stiffness matrix
CT11	Generate membrance stiffness transformation sub-matrix
MATI60	Invert 8 x 8 matrix in-core
CTOGM	Generate membrane transformation matrix for transformation from oblique to geometric coordinates
CTGRM	Generate membrane transformation matrix for transformation from geometric to reference system coordinates
CC1	Generate membrane stiffness sub-matrices
CMMASS	Generate membrance contribution to element mass matrix
CSTM	Generate membrane contribution to element stress matrix
CDM	Generate membrance displacement derivative matrix for element stress matrix control
CFMTS	Control generation of membrance contribution to element thermal stress and element thermal load matrices

CFMV	Generate membrane thermal load matrix
PRT1	Print membrane and flexure transformation matrices and contribution to element stiffness, stress, thermal stress, thermal load and pressure
CK22	Control generation of flexure stiffness matrix
CTGB	Generate flexure transformation sub-matrix
MATI70	Invert 16 x 16 matrix in-core
CTOGB	Generate flexure transformation matrix for transformation from oblique to reference system coordinates
CTGRB	Generate flexure transformation matrix for transformation from geometric to reference system coordinates
CC2	Generate flexure stiffness sub-matrices
CFP	Control generation of element pressure load matrix
CFPB	Generate intermediate element pressure load matrix
CSTF	Generate flexure contribution to element stress matrix
CDF	Generate flexure displacement derivative matrix for element stress matrix
CDFX	Generate flexure displacement partial with respect to X derivative matrix for element stress matrix
CDFY	Generate flexure displacement partial with respect to Y derivative matrix for element stress matrix
CFFTS	Control generation of flexure contribution to element thermal stress and thermal load matrices
CFFV	Generate flexure contribution to element thermal load matrix
CFMASS	Generate flexure contribution to element mass matrix

C. FRAME AND INCREMENTAL FRAME ELEMENT SUBROUTINES

CTS	Generate transformation matrix for transformation from geometric to reference
CTCQ	Generate transformation matrix for transformation from material to geometric axes
CECC	Evaluate effect of eccentricities
INCRE	Generate element incremental stiffness matrix
P7PRT	Print transformation matrices and intermediate calculations
PLUG7	Master control, generation of frame element matrices
PLUG22	Master control, generation of incremental frame matrices
FINP22	Generate element incremental matrix for the incremental frame element

D. TRIANGULAR PLATE AND QUADRILATERAL PLATE ELEMENT SUBROUTINES

DIRCOS	To evaluate the direction cosines given any three points that define a plane
BCB12	To evaluate a triple product matrix where all matrices are square
KOBLIQ	To perform a transformation on the element stiffness matrix AKEL ($TRAN^T$*AKEL*TRAN)
P1718M	Initialize element properties from the material table for membrane properties with flexural data only for PLUG17 and PLUG18
SELQ	To transform the stress matrix generated by PLUG17 and PLUG18 to the stress system required (generally local)
FTELQ	To transform the element thermal load matrix into global or oblique system
PLUG17	Master control for the generation of triangular plate element matrices
PLUG18	Master control for the generation of quadrilateral plate element matrices
TR18ST	From transformation matrices for the stress and thermal stress matrices
FBMP18	To evaluate the B matrix for the quadrilateral plate elements, out of plane

E. TRIANGULAR THIN SHELL ELEMENT SUBROUTINES

PLUG2	Master Control
ASSY2	Assemble membrane and flexure stiffness sub-matrices
DCD	Perform in-core matrix multiplication of the form TST where T is a diagonal matrix and S is a symmetric matrix
DTAPR	Process coordinate data
PFMASS	Calculate the flexural contribution to the mass matrix
PMMASS	Calculate the membrane contribution to the mass matrix
MATPR	Generate material properties matrices
NEWFT1	Calculate revised thermal matrices
PTBM	Generate membrane transformation matrix for transformation from oblique to geometric coordinate systems
PTMGS	Generate membrane transformation matrix for transformation from geometric to reference system coordinates
DPQINT	Calculate coordinate integrals
PKM	Generate membrane contribution to element stiffness matrix
PSTM	Generate membrane contribution to element stress matrix
PFMTS	Generate membrane contribution to element thermal load and thermal stress matrices
PFMV1	Generate intermediate membrane thermal load matrix
APRT	Print membrane and flexure transformation matrices and contributions to element stiffness, stress, thermal stress, thermal load and pressure load matrices

PTFGS	Generate flexure transformation matrix for transformation from geometric to reference system coordinates
PKF	Generate flexure contribution to element stiffness matrices
CCB	Perform in-core matrix triple product of the form $T^{T}KT$ where K is symmetric and accuracy criteria is imposed
PFP	Generate element pressure load matrix
PFFTS	Generate flexure contribution to element thermal stress and thermal load matrices
PFFV1	Generate intermediate flexure thermal load matrix
PSTF	Generate flexure contribution to element stress matrix
PTBF	Generate flexure transformation matrix for transformation from oblique to geometric coordinate systems
EPRT	Print final element matrices
PLAS2D	Non-functional
PNC1NE	Non-functional
PNG1NE	Non-functional

F. TRIANGULAR CROSS SECTION RING ELEMENT SUBROUTINES

PLUG6	Master control
EXPCOL	Expand column matrix to six degrees of freedom per point
EXPSIX	Expand symmetric matrix to six degrees of freedom per point
TRAIC	Generate coordinate transformation matrices and integrals
TESTJ	Impose accuracy criteria upon integrals
TRCPRT	Print coordinate transformation matrices and integrals
TRAIE	Generate material properties matrices
TIEPRT	Print material properties matrices
TRAIK	Generate element stiffness matrix
TIKPRT	Print element stiffness matrix
TRAIFP	Generate element pressure load matrix
TFPPRT	Print element pressure load matrix
TRAIFT	Generate element thermal load matrix
TFTPRT	Print element thermal load matrix
TRAIS	Generate element stress matrix
TISPRT	Print element stress matrix
TRAITS	Generate element thermal stress matrix
TTSPRT	Print element thermal stress matrix
TRAIM	Generate element mass matrix
TIMPRT	Print element mass matrix
TRAIFS	Generate element pre-strain load matrix
TFSPRT	Print element pre-strain load matrix

TRAIST	Generate element pre-stress load matrix
TSTPRT	Print element pre-stress load matrix
PL6PRT	Print all element matrices generated

G. TOROIDAL RING ELEMENT SUBROUTINES

PLUG5	Master control, generate element stiffness, thermal load, pressure load, stress and thermal stress matrices
ROMBER	Perform integration by Romberg Method
F4	Evaluate a defined function for ROMBER
F5	Evaluate a defined function for ROMBER
F6	Evaluate a defined function for ROMBER
AMATRX	Generate coordinate transformation matrix
DMATRX	Generate material properties matrix
GAMMAT	Generate material transformation matrix
FCURL	Generate intermediate thermal load matrix
PLMX	Generate intermediate pressure load matrix
SCRLM	Generate intermediate stress matrix
SOLVE	Solve for element stress coefficients
QUADI	Performs integration using numerical quadrature methods
PRINT5	Print generated element matrices

H. QUADRILATERAL SHEAR PANEL ELEMENT SUBROUTINES

PLUG14	Master control, generate element stiffness, stress and mass matrices
MULTF	Performs in-core specialized matrix multiplication
P14PRT	Prints intermediate calculations and generated element matrices

I. TRAPEZODIAL RING ELEMENT SUBROUTINES

PLUG8	Master control for the generation of trapezodial ring element matrices
SUBINT	Solves the integral $H = \iint \frac{z^Q}{R} drdz$ for values of Q = 0, 1 and 2 for a trapezoid
ZMRD	Perform double precision multiplication of two matrices (C = A * B)
ZTRD	Perform double precision multiplication of two matrices ($C = A^T * B$)
KMPY	Multiply, in double precision, each element of a matrix by a scalar to form a resultant matrix
ERIC	Compute the pressure load vector for the trapezodial ring
P8MASS	Generate element mass matrix for the trapezodial ring

J. Rectangular Prism Element Subroutines

PLUG20	-	Master Control
GEOMD	-	Generate geometry data
TRAAE	-	Generate material properties matrix
MPOWJF	-	Print element matrices
GAMMAG	-	Generate reorder transformation matrix
GAMMAS	-	Generate system transformation matrix
DISPMT	-	Generate strain-displacement matrix
MASS20	-	Generate consistent mass matrix
P20FT	-	Generate thermal load
P20FP	-	Generate pressure load
PASSYM	-	Assembles local consistent mass matrix into system mass matrix

K. Tetrahedran Element Subroutines

PLUG10	-	Master Control
P10TM	-	Generates transformation matrix
CMASS	-	Generates the consistent mass matrix
BDEF	-	Generates area terms of tetrahedran
P10FP	-	Generates pressure load
VOL	-	Generates volume of tetrahedran

L. Triangular Prism Element Subroutines

PLUG13	-	Master Control
PLUG10	-	Generates element matrices for the 3 tetrahedrans of the prism

M. Symmetrical Shear Web Element Subroutines

PLUG23 - Master Control

N. Asymmetric Triangular Cross Section Ring Element subroutines

PLUG25	-	Master Control
HRAIKS	-	Generate element stiffness matrix
HRAIES	-	Generate elastic constants
HLOGEN	-	Generate pressure load
HMASSG	-	Generate mass matrix
HTHGEN	-	Generate thermal load
HRAICS	-	Generate transformation matrix and area integrals
HRAISS	-	Generate stress matrix

O. High Aspect Ratio Quadrilateral Thin Shell Element Subroutines

PLUG26	-	Master Control
NEWFT	-	Calculate thermal load
CDELPQ	-	Calculate coordinate integrals
CK26	-	Generate membrane stiffness and transformation matrix
CMMASS	-	Generate membrane mass matrix
PIPRTA	-	Print results of coordinate and material properties matrix
GENSM	-	Calculate mode shapes and derivative functions
PRT1	-	Print transformation, stiffness, stress, thermal stress, thermal load, and pressure matrices
CK22	-	Generates flexural stiffness matrix
CFP	-	Generates pressure load
CSTF	-	Generates flexural stress matrix
CFFTS	-	Generates flexural thermal load and thermal stress
CFMASS	-	Generates flexural mass matrix
STRS26	-	Generates membrane stress matrix
FT26	-	Generates membrane thermal load matrix
SZAL26	-	Generates membrane thermal stress matrix

APPENDIX IV

LIST OF SUBROUTINE FUNCTIONS OF MODULES ADDED TO THE FORMAT II SYSTEM

ANALIC	Control routine for statics analysis in core when using the .ANALIC. abstraction instructions
ASSEM	Control routine for assembling element matrices using the .ASSEM. abstraction
ASSEMC	Assembly thermal load element matrices
ASSEMS	Assembly element stiffness, element mass and element incremental matrices
CHEQS	Control routine for solving simultaneous equations and triangularizing matrix using the .CHOL., .CHTRIA., .TRIA. abstraction instructions
COLMRD	Utility subroutine to uncompress a column of a matrix in dynamic storage
COLREP	Generate a rectangular matrix by repeating the input column the specified number of times using the .COLREP. abstraction instruction
DECODE	Generate a copy of a Format matrix on a scratch tpae in full format
DEJNC	Perform column partitioning of a matrix
DEJNR	Perform row partitioning of a matrix
DEJOIN	Control routine for matrix partitioning using the .DEJOIN. abstraction instruction
DISPL1	Printing routine used by GPRINT
DISPPR	Controls printing of displacements from GPRINT
EIG	Main iteration routine of .EIGEN1. module
EIGB	Controls iteration routine EIG
EIG1	Controls routine for calculating eigenvalues and eigenvectors using the .EIGEN1. abstraction instruction
EIGPPR	Controls printing of eigenvalues and vectors

ELREAD	Routine to decode the compressed element matrix output by the .USER04. module
EPRINT	Controls printing of element stresses and forces when using the .EPRINT. abstraction instruction
FORCE	Control routine to calculate element force when using the .FORCE. abstraction instruction
FORCE1	Routine to set up dynamic sotrage and control calculation of element forces for each element
FORCE2	Calculates element force
FREEUP	Routine to return work sotrage to available use
GPRINT	Control routine to print reactions, displacement, eigenvalues and eigenvectors when using the .GPRINT. abstraction instruction
GPRNT1	Controls storage and correct print transfers for .GPRINT.
HDECO	Control routine to calculate element matrices dependent upon the Nth harmonic when using the .HDEC. abstraction instruction
HSUM	Control routine for assembling the contributions of the harmonics when using the .HSUM. abstraction instruction
IDENTC	Generates an identity matrix when using the .IDENTC. abstraction instruction
IDENT	Generates an identity matrix when using the .IDENTR. abstraction instruction
INST04	Instruction analyzer for the .GRPINT. instruction
INST05	Instruction analyzer for the .EPRINT. instruction
INST43	Instruction analyzer for the .DEJOIN. instruction
INST60	Instruction analyzer for the .STRESS. and .FORCE. instructions
MATPRT	Controls printing of a user matrix when using .GPRINT.

MATSUP	Insert suppressed input matrix names into the Format system
NULL	Generates a null matrix using the .NULL. abstraction instruction
REACTP	Controls printing of reactions when using .GPRINT.
REGE2	Utility routine used by EIG1
REPLAS	Control routine to replace one matrix with another matrix when using the .REPLAC. abstraction instruction
STRESS	Control routine to calculate element stresses when using the .STRESS. abstraction instruction
STRES1	Routine to set up dynamic storage and control calculation of element stresses for each element
STRES2	Calculates element stresses
STRPRT	Prints element stresses and forces
TSUM	Generates a tape summary of matrices on a specified logical unit

APPENDIX V

REVISIONS TO FORMAT SYSTEM DECKS

Subroutine Name: PREP

Purpose of Revision: Provide the cabability for suppressing input matrices in an abstraction instruction

Method: Fortran statement number 200 was changed to initialize the variable NUMSUP to zero. NUMSUP was added to the calling sequence to subroutine INST and upon return will contain the number of input suppressed matrices located during compilation of the input abstraction instructions. If NUMSUP is non-zero upon return from INST, then subroutine MATSUP is called to introduce the input suppressed matrices into the Format system.

Subroutine Name: EUTL4

Purpose of Revision: To retain the second word in the matrix header when copying a matrix. Thus the KODE word in the matrix header will not be changed to zero when copying a matrix.

Method: After EUTL3 finds the matrix to be copied, a backspace is issued to read the KODE word of the matrix header. This KODE is transferred to the matrix header of the new matrix.

Subroutine Name: EUTL5

Purpose of Revision: To insure that the second word in the matrix header is given the value assigned by the user in the calling argument of EUTL5 to the variable KODE.

Method: When writing the matrix header write the variable KODE from the argument list as the second word of the header.

Subroutine Name: INST

Purpose of Revision: Provide distinct names for suppressed matrices and record the number of input suppressed matrices encountered while compiling the abstraction instructions.

Method: The variable NUMSUP was added to the calling sequence of INST and inserted into the calling sequence for INST90 to record the number of input suppressed matrices located. The variable KOUNT was initialized in INST as zero and inserted in the calling sequence to INST90 to be used as a counter to ensure the generation of unique suppressed matrix names.

Subroutine Name: INST90

Purpose of Revision: Introduce unqiue matrix names into the Format system for both output and input suppressed matrices for the .USERXX. form input abstraction instruction.

Method: The variables KOUNT and NUMSUP were added to the calling sequence for subroutine INST90, KOUNT to indicate the next unique suppressed matrix name and NUMSUP to record the number of input suppressed matrices encountered. Whether input or output, a suppressed matrix is located and a name assigned to it by the same procedure. All blanks have been removed from the input instruction by subroutine PUTL1. The instruction is scanned, first the output side, then the input side. Whenever a matrix position has length zero, i.e. the matrix name was blank, the suppressed name is created by inserting four slashes for the first four characters and adding one to KOUNT and inserting that value as the last two characters. The sign of the matrix is set to plus. If the suppressed matrix was an input matrix, i.e. was encountered on the right sign of the equal sign, then NUMSUP is incremented by one.

Subroutine Name: MATR

Purpose of Revision: Provide the capability of placing card input matrices on the same data set as input suppressed matrices, if necessary.

Method: If card input matrices are present then subroutine MATR is called to place these matrices on NDATA, the data set selected by the Format pre-processor for that purpose. However, if input suppressed matrices were present then they already exist on NDATA at the time that MATR is called. Therefore MATR had to be revised to check NUMD, the variable indicating the number of matrices already on NDATA, before recording card input matrices on NDATA. If NUMD is zero then NDATA is rewound and a data set header written and the card input matrices recorded. If NUMD is non-zero, then NDATA is searched until the data set trailer is located, then backspaced over the data set trailer and then the card input matrices are recorded.

Subroutine Name: ALOC

Purpose of Revision: Pass the value of IPRINT, the Format system print control, to subroutine ALOC4 for transmittal when operating under SUBSYS control.

Method: The variable IPRINT was added to the calling sequence for ALOC and inserted into the call statement to ALOC4.

Subroutine Name: ALOC31

Purpose of Revision: Indicate to the Format system the number of scratch data sets required to execute the .USER04. instruction.

Method: The variable MINSCR(94) was set equal to four.

Subroutine Name: ALOC4

Purpose of Revision: Store on the instruction data set, NINST, the necessary data for re-initialization of program constants for operation under Subsys control.

Method: When proceeding from program to program under Subsys control, the necessary system parameters must be reset at the start of each program. The values of the parameters are obtained as follows: NPIT, the system input unit, NPOT, the system output unit, KONST, the maximum matrix size capability and NWORK, the number of available work storages are obtained via the COMMON statement in ALOC4. The value of IPRINT is received through the calling sequence of ALOC4. These five system parameters, NPIT, NPOT, KONST, NWORK and IPRINT, are added as extra words to the return instruction recorded on NINST.

APPENDIX VI

MAGIC ERROR MESSAGES

The following is a list of all MAGIC error messages. The list is divided into three sections. The first section contains all Format error messages (Reference 2) and is divided into two parts, the preprocessor error message, and the execution error message. The second section contains error messages from all arithmetic and non-arithmetic modules developed to be used in conjunction with the structural generative module. The third section contains error messages generated by the structural generative system itself, which is the .USER04. module. In each section the error messages are in alphabetic order. The error message codes are significant in that the first six characters identify the subroutine from which the error message emanates. The occurrence of **** in the error message indicates that additional descriptive information will be supplied.

SECTION 1. FORMAT ERROR MESSAGES

ALOC01 INSUFFICIENT STORAGE FOR ALLOCATION

The number of words of working storage available to the allocator is less than the minimum required for complete allocation of this job. This condition can be remedied by reducing the number of abstraction instructions.

ALOC02 INVALID NO. OF MASTER INPUT/OUTPUT DATA SETS SPECIFIED

The number of master input data sets and/or master output data sets specified on "INPUT TAPE" or "OUTPUT TAPE" cards is greater than the number of master input and/or master output data sets defined in the machine resources area as being available to FORMAT II. This condition can be remedied by reducing the number of "INPUT TAPE" and/or "OUTPUT TAPE" cards.

ALOC03 INSUFFICIENT UTILITY DATA SETS FOR ALLOCATION

The number of data sets with the FORMAT II system function IOUTIL is less than the minimum number required by the FORMAT II Preprocessor during the preprocessing phase. This condition can be remedied by reducing the number of "INPUT TAPE" or "OUTPUT TAPE" cards used in this job or by modifying the machine resources area. (i.e., define additional data sets with the FORMAT II system function IOUTIL.

ALOC04 MASTER OUTPUT DATA SET ****** SPECIFIED IN SAVE INSTRUCTION NOT DEFINED

A "SAVE" instruction in the abstraction instruction sequence refers to a master output data set name which has not been defined on an "OUTPUT TAPE" card. This condition can be remedied by including the appropriate "OUTPUT TAPE" card in the job.

ALOC05 MASTER INPUT DATA SET ****** HAS NOT BEEN MOUNTED

The FORMAT II allocator has not been able to locate a master input data set which has been specified on an "INPUT TAPE" card. This condition is usually caused by mounting the correct master input data set on the wrong unit or by misspelling the name of a properly mounted data set on the "INPUT TAPE" card.

ALOC06 MATRIX ****** IS NON-EXISTENT

A matrix, which appears in the abstraction instruction sequence and which has not been created in the abstraction instruction sequence prior to its use, has not been card input and does not appear on any master input data set. This condition can be remedied by inputting the required matrix.

ALOC07 DUPLICATE MATRICES ****** IN MATRIX DATA

Two or more matrices with the same name have been card input. This condition can be remedied by ensuring that all card input matrices have unique names.

ALOC08 CREATED MATRIX ****** IS CARD INPUT

A matrix which is created in the abstraction instruction sequence has the same name as a matrix which is card input. This condition can be remedied by removing the matrix in question from the card input matrix data.

ALOC09 SUBSCRIPTS OF ****** EXCEED DIMENSIONS OF MATRIX

The indices of a scalar element to be extracted from a matrix are larger than the dimensions of that matrix. This condition can be remedied by changing the indices of the scalar element specified in the abstraction instruction sequence.

ALOC10 DUPLICATE MATRICES CREATED -- NAME ******

A matrix in the abstraction instruction sequence appears more than once on the left side of an equal sign. This condition can be remedied by ensuring that all matrix names, which appear on the left side of an equal sign in the abstraction instruction sequence, have unique names.

ALOC11 MATRIX ****** IS USED MORE THAN ONCE IN INSTRUCTION ***

The matrix names appearing in the indicated instruction in the abstraction instruction sequence do not have unique names. This condition can be remedied by ensuring that all matrix names appearing in a given abstraction instruction have unique names.

ALOC12 CREATED MATRIX ****** HAS BEEN INPUT

A matrix which appears on the left side of an equal sign in the abstraction instruction sequence has the same name as a required input matrix. This condition can be remedied by either changing the name of the required input matrix or by changing the name of the matrix which appears on the left side of the equal sign.

ALOC13 MATRICES CREATED IN INSTRUCTION *** NEVER REFERENCED

The indicated abstraction instruction in the abstraction instruction sequence creates matrices, none of which are referenced in subsequent abstraction instructions. This condition can be remedied by removing the indicated abstraction instructions from the abstraction instruction sequence.

ALOC14 DUPLICATE STATEMENT NUMBERS ******

Duplicate statement numbers occur in the abstraction instruction sequence. This condition can be remedied by ensuring that each statement number occuring in the abstraction instruction sequence is unique.

ALOC15 GO TO DESTINATION ****** IS MISSING OR OCCURS BEFORE IF TEST

An abstraction instruction "IF" in the abstraction instruction sequence conditionally transfers to a non-existent statement number or transfers to a statement number on an abstraction instruction which is sequentially earlier than the "IF" abstraction instruction in question. This condition can be remedied by ensuring that all "IF" abstraction instructions conditionally transfer to a statement number which occurs sequentially after the "IF" abstraction instruction.

ALOC16 NON CONFORMABLE MATRICES IN INSTRUCTION ***

Two matrices occur in the indicated abstraction instruction in the abstraction instruction whose dimensions are such that the matrix operation in the indicated abstraction instruction is not defined.

EXEQ01 THE FORMAT SYSTEM IS UNABLE TO LOCATE MATRIX ******

This message signifies a malfunction of the user-coded subroutine which creates the specified matrix.

EXEQ02 CONFORMABILITY ERROR IN INSTRUCTION CREATING MATRIX ******

The matrices involved on the right side of the equals sign in the instruction creating the specified matrix are unconformable.

EXEQ03 MATRIX ****** IS SINGULAR

The matrix is singular in a "Solution of Equations" routine, i.e., in "STRCUT," "SEQEL" or "INVERS."

EXEQ04 AN ERROR HAS OCCURRED IN THE USER ** MODULE

An error recognized by the indicated user-coded subroutine has occurred. This will usually be associated with incorrect definition of the special data for use by thesubroutine.

EXEQ05 AN IMPROPER UPDATE HAS BEEN MADE TO THE FORMAT SYSTEM - EXECUTION TERMINATED

A new permanent module has not been properly incorporated. The FORMAT II systems analyst should be contacted if this error message occurs.

EXEQ05 AN ERROR HAS OCCURRED IN A USER-CODED MODULE, ERROR HAS BEEN WRITTEN BY MODULE

An error has occurred in a non-Format module. The specific error has been written by the subroutine in which the error was found.

EUTL3 THE SYSTEM IS UNABLE TO LOCATE A MATRIX. A TAPE SUMMARY OF LOGICAL UNIT **** WILL FOLLOW

The Format system is unable to locate a matrix. A tape summary of the data set on which the matrix should have been is printed out. The name of the matrix will appear in the next error message.

INST01 ILLEGAL OPTION SPECIFIED ON $INSTRUCTION CARD

An option other than "SOURCE" or "NOSOURCE" has been specified on the "$INSTRUCTION" card or a valid option starts before card column 16 in the "$INSTRUCTION" card.

INST02 INVALID STATEMENT NUMBER SPECIFIED

The statement number which is specified in card columns 1-5 of the abstraction instruction preceding this error message is composed of characters which are not all numeric.

INST03 INVALID CHARACTER IN COLUMN 6

Card column 6 of the abstraction instruction preceding this error message contains a character other than a blank or zero.

INST04 UNRECOGNIZABLE OPERATION CODE

The operation specified in the abstraction instruction preceding this error message is not contained in the FORMAT II library of valid operations.

INST04 SYNTAX ERROR IN - GPRINT - INSTRUCTION

INST04 ILLEGAL NEGATIVE INPUT VALUE FOR SUPPRESSION OF MATRIX ELEMENTS, ABSOLUTE VALUE TAKEN

The effective zero value for suppression of element print in the GPRINT instruction must be positive.

INST04 INVALID SPECIFICATION OF INPUT MATRICES

An incorrect number of input matrices has been specified in the GPRINT instruction.

INST04 ILLEGAL SPECIFICATION OF COLUMN HEADERS

Incorrect syntax in GPRINT when written column headers.

INST05 SYNTAX ERROR IN - IF - INSTRUCTION

The abstraction instruction "IF" which precedes this error message contains an unrecognizable field.

INST05 SYNTAX ERROR IN - EPRINT - INSTRUCTION

INST05 INVALID PRINT CONTROL

The print control in the EPRINT instruction was incorrectly specified.

INST05 ILLEGAL NEGATIVE INPUT VALUE FOR SUPPRESSION OF MATRIX ELEMENTS, ABSOLUTE VALUE TAKEN

The effective zero value for suppression of element print in the EPRINT INSTRUCTION must be position.

INST05 ILLEGAL SUPPRESSION OF PARAMETER

The code indicating either stress or force matrices to be printed has been omitted.

INST06 SYNTAX ERROR IN - PRINT - INSTRUCTION

The abstraction instruction "PRINT" which precedes this error message contains an unrecognizable field.

INST07 SYNTAX ERROR IN - SAVE - INSTRUCTION

The abstraction instruction "SAVE" which precedes this error message contains an unrecognizable field.

INST08 OPERATION CODE NOT INCLOSED BY PERIODS

The operation code in the abstraction instruction preceding this error message is not inclosed by periods.

INST09 SYNTAX ERROR IN ARITHMETIC INSTRUCTION

The arithmetic abstraction instruction preceding this error message contains an unrecognizable field.

INST10 THIS INSTRUCTION IS NOT AVAILABLE

An incomplete modification to the instruction card processor area has been made. The FORMAT II systems analyst should be notified immediately.

INST43 INVALID SPECIFICATION OF PARAMETERS

A syntax error has occurred in the DEJOIN instruction.

INST43 - INVALID INDEX SPECIFIED

Parameter specifying row or column dejoin is illegal.

INST43 INVALID MATRIX NAME

The DEJOIN instruction contains one invalid matrix name.

MATR01 UNRECOGNIZABLE OPTIONS ON $MATRIX CARD STANDARD OPTIONS USED WARNING ONLY

An option other than "LIST", "NOLIST", "PRINT" or "NOPRINT" has been specified on the "$MATRIX" card or a valid option starts before column 16 on the "$MATRIX" card.

MATR02 CARD FOLLOWING $MATRIX CONTROL CARD IS NOT A HEADER CARD OR HAS - H - MISSING IN COLUMN 1

The first card following the "$MATRIX" card must be the header card of the first card input matrix. All data up to the first header card will be ignored.

MATR03 NAME ON DATA CARD IS DIFFERENT FROM NAME ON HEADER CARD. THIS MATRIX WILL BE IGNORED

The matrix header card and all associated matrix data must have the same name in card columns 67-72.

MATR04 ROW AND/OR COLUMN VALUE EXCEED MATRIX SIZE, IS NEGATIVE OR IS ZERO AND VALUE IS NONZERO. THIS MATRIX WILL BE IGNORED.

An element specified in the matrix card input data is outside the dimensions of the matrix, of which it is supposed to be an element.

MATR05 MATRIX EXCEEDS ALLOTTED STORAGE. THIS MATRIX WILL BE IGNORED.

The number of words of working storage available to the matrix card reader module is less than the number of words necessary to contain all the nonzero elements in one of the card input matrices. The number of words of working storage required for a given matrix is approximately three (3) times the number of nonzero elements in the matrix. This condition can be remedied by decreasing the number of nonzero elements in the card input matrix.

MATR06 DUPLICATE I-J VALUES ENCOUNTERED. THIS MATRIX WILL BE IGNORED. I = **** J = ****

Two or more values have been specified for the same matrix element in the matrix card input data. This condition can be remedied by ensuring that each matrix element has a unique set of I - J values.

MATR07 I VALUE ON HEADER CARD EXCEEDS ALLOTTED SIZE OR IS LESS THAN OR EQUAL TO ZERO. THIS MATRIX WILL BE IGNORED.

The number of rows specified in the header card of a card input matrix is greater than the maximum number of rows permitted in a matrix which is processed by the FORMAT II system, or is less than or equal to zero. This condition can be remedied by reducing the dimensions of the card input matrix.

MATR08 J VALUE ON HEADER CARD EXCEEDS ALLOTTED SIZE OR IS LESS THAN OR EQUAL TO ZERO. THIS MATRIX WILL BE IGNORED.

The number of cclumns specified in the header card of a card input matrix is greater than the maximum number of columns permitted in a matrix which is processed by the FORMAT II system, or is less than or equal to zero. This condition can be remedied by reducing the dimensions of the matrix.

MATR09 FIRST CHARACTER OF MATRIX NAME ON HEADER MUST BE ALPHABETIC. THIS MATRIX WILL BE IGNORED.

The matrix name which is to be given to a set of matrix card input data and which is punched in card column 67-72 of the header card and all associated data cards must follow the rules for valid matrix names as defined for the FORMAT II system. The rule which applies in this case is that the first character of a matrix name must be alphabetic.

MATR10 ILLEGAL CARD ENCOUNTERED. FOLLOWING CARDS IGNORED UNTIL ANOTHER - $ - CONTROL CARD IS FOUND.

A card has been encountered in the matrix card input data which has an illegal character punched in card column 1. The only valid characters which may appear in card column 1 are "H", "E", and blank.

MATR11 CARD FOLLOWING E CARD IS NOT A $ CONTROL CARD - WARNING ONLY.

In a valid FORMAT II deck setup the only cards which may follow the "E" card which is the last card in the matrix card input data, are the "$SPECIAL" card and the "$END" card.

MRES01 FIRST CARD IS NOT A - $ - CONTROL CARD

The first card of all FORMAT II jobs must be a "$MAGIC" or a "$FORMAT" card.

MRES02 FIRST - $ - CONTROL CARD IS NOT A $MAGIC CARD. ALLOCATION SUPPRESSED

The first card of all FORMAT II jobs must be a "$MAGIC" or a "$FORMAT" card.

MRES03 UNRECOGNIZABLE OPTION ON - $MAGIC CARD STANDARD OPTION ASSUMED

An option other than "NEW", "STANDARD" (or blank) or "CHANGE" has been specified on the "$MAGIC" card or a valid option starts before column 16 on the "$MAGIC" card.

MRES04 ILLEGAL CARD FOR - CHANGE - OPTION - ALLOCATION SUPPRESSED

The "DELETE" card and the "UPDATE" card are the only valid machine resources data cards which are valid when the "CHANGE" option has been specified on the "$FORMAT" card. The "SETUP" card is the only valid machine resources data card which is valid when the "NEW" option has been specified on the "$FORMAT" card.

MRES05 THE SYSTEM INPUT DATA SET OR OUTPUT DATA SET HAS BEEN SPECIFIED AS A FORMAT II SYSTEM FUNCTION

Two Fortran logical data sets which must not be specified on "UPDATE", "DELETE", or "SETUP" cards are the system input data set and the system output data set.

MRES06 DUPLICATE DATA SETS SPECIFIED - ALLOCATION SUPPRESSED

A Fortran logical data set has been specified more than once on "SETUP" or "UPDATE" cards.

MRES07 INVALID **** VALUE DETECTED ALLOCATION SUPPRESSED

An invalid field has been specified on an "UPDATE" or "SETUP" card. The valid fields are as follows. The first field must contain the logical data set number (an integer). The second field a valid FORMAT II system function (e.g., "MASTRI", "MASTRO", or "IOUTIL"). The third field must contain the physical device containing the data set. The valid specifications in the field are "TAPE", "DISK", "DRUM", or "CELL". The fourth field must contain the logical channel designation. This consists of a letter A to H. The fifth field must contain the capacity of the data set in basic machine units (e.g., bytes, etc.). This field must be an integer number. The error message indicates which of the five fields is in error.

MRES08 INCORRECT SETUP OR UPDATE CARD ALLOCATION SUPPRESSED

A missing field has been detected on a "SETUP" or "UPDATE" card.

MRES09 INSUFFICIENT I/O UTILITY DATA SETS - ALLOCATION SUPPRESSED

A minimum number of Fortran logical data sets available to FORMAT II must have the FORMAT II system function of "IOUTIL". The FORMAT II preprocessor selects several of the data sets with this function for scratch data sets during preprocessing. This condition can be remedied by specifying additional data sets on "SETUP" or "UPDATE" cards with the FORMAT II system function "IOUTIL".

MRES10 ILLEGAL DEVICE SPECIFIED FOR MASTER INPUT DATA SET

The only valid device types which may be specified for a FORMAT II data set whose system function is "MASTRI" are "TAPE" and "DISK". A "SETUP" or "UPDATE" card is the source of the error.

MRES11 ILLEGAL DEVICE SPECIFIED FOR MASTER OUTPUT DATA SET

The only valid device types which may be specified for a FORMAT II data set whose system function is "MASTRO" are "TAPE" and "DISK". A "SETUP" or "UPDATE" card is the source of the error.

PREP01 INVALID CONTROL CARD OR INCORRECT DECK SETUP

The FORMAT II preprocessor has encountered a control card which is unrecognizable or which is valid but does not occur in its proper place. Recommended corrective action is to check the spelling of all control cards and check the deck set up.

PREP02 NOT A - $ - CONTROL CARD. CARD IGNORED

When an invalid control card is encountered or incorrect deck setup is recognized, the preprocessor searches for the next "$" control card.

PREP03 PREPROCESSING TERMINATED EXECUTION HALTED

Whenever a serious error occurs the preprocessing is terminated and a "NOGO" condition is established.

PROB01 UNRECOGNIZABLE OPTION ON - $RUN - CARD. STANDARD OPTION USED.

An option other than "GO", "NOGO", "LOGIC" or "NOLOGIC" has been specified on the "$RUN" card or a valid option starts before column 16 in the "$RUN" card.

PROB02 CONTRADICTORY EXECUTION OPTIONS - ALLOCATION SUPPRESSED

The options "GO" and "NOGO" have been specified on the "$RUN" card.

PROB03 CONTRADICTORY LGOIC OPTIONS - ALLOCATION SUPPRESSED

The options "LOGIC" and "NOLOGIC" have been specified on the "$RUN" card.

PROB04 MISSING LEFT PARENTHESIS - ALLOCATION SUPPRESSED

A problem specification data card has a missing left parenthesis.

PROB05 UNRECOGNIZABLE CARD

A problem specification data card is unrecognizable. The valid problem specification data cards are the "ANALYSIS" card, the "PROBLEM" card, the "PAGE SIZE" card, the "INPUT TAPE" card, and the "OUTPUT TAPE" card.

PROB06 MISSING COMMA ON MASTER I/O TAPE CARD - ALLOCATION SUPPRESSED

There is a missing field on an "INPUT TAPE" card or on an "OUTPUT TAPE" card in the problem specification data.

PROB07 ILLEGAL MASTER I/O DATA SET NAME - ALLOCATION SUPPRESSED

The master input or master output data set name which has been specified on "INPUT TAPE" card or on "OUTPUT TAPE" card in the problem specification data is invalid. Master Input/Output data set names follow the same rules as matrix names. In particular, the name must be 1-6 characters long and the first character must be alphabetic.

PROB08 ILLEGAL INTEGER ON MASTER I/O TAPE CARD

The second field of an "INPUT TAPE" or "OUTPUT TAPE" card in the problem specification data is not an integer number.

PROB09 ILLEGAL PAGE SIZE - ALLOCATION SUPPRESSED

An invalid page size has been specified on the "PAGE SIZE" card in the problem specification data. The valid page sizes are "11 * 8", "8 * 11" and "14 * 11".

PROB10 MASTER INPUT OR OUTPUT DATA SET USED PREVIOUSLY

All master input and output data set names as specified on "INPUT TAPE" and "OUTPUT TAPE" cards in the problem specification data must be unique.

PROB11 INVALID SIZE SPECIFIED ON SIZE CARD

An integer number must be specified in the only field of the "SIZE" card.

SECTION 2. MISCELLANEOUS ARITHMETIC MODULE ERROR MESSAGE

ASSEM - The order of the assembled - unreduced system, NSYS = *****, the maximum size system can only = ***** D.O.F.

The variable KONST in subroutine MRES must be updated to allow the user to assemble a system with NSYS degrees of freedom.

ASSEMC - Element number *****, generated a LISTEL value of *****, while NSYS = *****.

If this error occurs see the MAGIC system analyst.

ASSEMS - Must update the dimension of the list and format arrays to allow for ***** degrees of freedom.

The dimension of two arrays in subroutine ASSEMS must be updated to assemble more degrees of freedom than allowed. If this error occurs see the MAGIC system analyst.

COLREP - Input matrix ****** exceeds allowable size IMAX = *****.

The number of rows of the input matrix exceeds the value of KONST. IMAX is the number of rows in the input matrix.

DEJNC - The partition number = *****, is greater than or equal to the column dimension = ***** of the input matrix.

An invalid column partition number has been specified in the DEJOIN instruction $1 \leq JPART < ICOL$.

DEJNR - The partition number = *****, is greater than or equal to the row dimension = ***** of the input matrix.

An invalid row partition number has been specified in the DEJOIN instruction $1 \leq JPART < IROW$.

DEJOIN - Invalid partition number = *****

The matrix partition number must be greater than one.

EPRINT - Unable to execute the EPRINT module. The work array is not long enough for execution.

The variable NWORK in subroutine MRES must be updated for more work storage.

EPRINT - The element information is for element number **** - go to next element.

Unable to print out stresses or forces for this element, continue execution. If this error occurs contact the MAGIC system analyst

EPRINT - The number of elements in the input matrices are not the same.

If this error occurs contact the MAGIC system analyst.

EPRINT - Printing for element type *****, are not available, proceeding to next element.

The EPRINT module has not been updated to handle this element type. Contact the MAGIC system analyst.

FORCE1 - Unable to execute the force module. The work array contains ******** words, and ******** words are needed to process the maximum element.

There is not enough work storage to calculate the forces for all elements. The variable NWORK must be updated in subroutine MRES.

FORCE2 - Forces for element type *****, are not available, proceeding to next element.

The FORCE module has not been updated to handle this element type. The MAGIC system analyst should be contacted if this error occurs.

FREEUP - The number of matrices to be kept was input as MATOUT = ******, the number of non-zero elements of MAT = ****.

If this error should occur contact the MAGIC system analyst.

GPRNT1 - The row dimension of TR(transformation matrix for application of boundary conditions) = ******. The number of columns of TR = ******. This should equal row dimension.

An incorrect matrix was input in the .GPRINT. instruction.

GPRNT1 - The analyst has asked for ***** eigenvalues to be printed. Subroutine GPRINT allows a maximum of ***** values to be printed - see a program analyst to correct this error

Subroutine GPRINT must be updated to allow more eigenvalues to be printed.

GPRNT1 - Error while processing matrix ******.

An error has occurred in the GPRINT instruction while processing matrix named.

GPRNT1 - The matrix to be printed has ****** rows while TR indicates that it should have ****** rows.

The input matrix to be printed is incorrect or the input transformation matrix is incorrect.

GPRNT1 - Eigenvector matrix has ***** eigenvectors, while the eigenvalue matrix has ***** eigenvalues.

The eigenvector and eigenvalue matrices input into the GPRINT instruction are not compatable.

STRES1 - Unable to execute the STRESS module. The work array contains ******** words, and ******** words are needed to process the maximum element.

There is not enough work storage to calculate the stresses for all elements. The variable NWORK must be updated in subroutine MRES.

STRES2 - Stresses for element type *****, are not available proceeding to next element.

The STRESS module has not been updated to handle this element type. The MAGIC system analyst should be contacted if this error message occurs.

SECTION 3. .USER04. ERROR MESSAGES

CHEK - Input section **** has not been found. This input section is required for generation of the following matrices.

The named matrices cannot be generated due to the omission of the specified input section.

CONTRL - System information card missing. Cannot allocate storage.

All input data decks must have SYSTEM section to allocate storage for processing of input.

CONTRL - System information card missing. Cannot allocate storage.

The SYSTEM card is missing from the report form input deck.

CONTRL - $END card encountered while reading .USER04. input, indicating absence of end or check card. Check card will be inserted.

END or CHECK card missing from report form input deck.

DEFLEX - .USER04. Module unable to locate matrix ******.

The system is unable to locate a matrix.

DEFLEX - Matrix ****** does not qualify as an input displacement matrix for the .USER04. module. Dimensions are ***** by ***** and should be ***** by *****.

The input displacement matrix used to calculate incrementals is of the wrong order.

DEFLEX - Matrix ****** does not qualify as an input displacement or stress matrix.

The input matrix used to calculate incrementals is of the wrong order. If the matrix was a stress matrix then it must have been generated using the .STRESS. abstraction instruction.

ELEM - Element control error in subroutine ELEM.
Element number **** calls plug number ***.
Plug number should be greater than zero.
Execution terminated.

All element type code numbers are greater than zero. Proper element type cannot be selected.

ELEM - Element control error in subroutine ELEM.
Element number ***** has material number *****.
Material identification must be different from zero.
Execution terminated.

Self-explanatory.

ELEM - Element control error in subroutine ELEM. Element number ***** has number of grid points = ***.
Number of grid points must be greater than zero and no greater than eight. Execution terminated.

Self explanatory.

ELEM - Element control error in subroutine ELEM. Element number ***** has number of input points = **.
Number of input points must be position.
Execution terminated.

Self-explanatory.

ELEM - Input error in subroutine ELEM. Element node point is negative or zero in element number *****.

No element defining point number may be negative and only mid-points may be zero.

ELEM - Input error in subroutine ELEM, after interpolation value of Young's Modulus equals +.******** + ** in material number ******, **********. Value should be greater than 1.0. Execution terminated.

Self-explanatory.

ELEM - Input error in subroutine ELEM, after interpolation Poisson value equals ±.********E + ** in material number ******, **********. Value should be greater than -1.0 and less than 1.0. Execution terminated.

Self-explanatory.

ELEM - Input error in subroutine ELEM, after interpolation thermal coefficient values equals +.********E +.** in material number ******, **********. Value should be greater than -1.0 and less than 1.0. Execution terminated.

Self-explanatory.

ELEM - Input error in subroutine ELEM, after interpolation rigidity value equals ± .********E + ** in material number ******, **********. Value should be greater than 1.0. Execution terminated.

Self-explanatory.

ELEM - Input error in subroutine ELEM. Mass density value equals ± . XXXXXXXXE + ** in material number ******, **********. Value should be greater than zero. Execution terminated.

Self-explanatory.

ELEM - Input error in subroutine ELEM. Value of IP = ***, value of IPRE = *** for element number one. Request to repeat data from element previous to first element is illogical. Execution terminated.

IP and IPRE cannot be negative for first element.

ELEM - Input error in subroutine ELEM. Element number ****** is defined by node points for which no coordinates have been input. Calculation of material temperature impossible. Execution terminated.

Self explanatory.

ELEM - Cannot locate material library.

The system cannot locate the material library matrix.

ELEM - Material error in subroutine ELEM. Material number ****** was not located on material tape. Execution terminated.

The specified material number was not available in the material library.

ELPLUG Input error for element number * * *, element type * * *, element grid points * * * * * * * * * * * *

FMAT - Input error in subroutine FMAT. Material number ******, **********. Number of material temperature points is **. Number of plastic temperature points is **. Number of temperature points in either case cannot exceed 9. Execution terminated.

Self explanatory.

FMAT - Input error in Subroutine FMAT. Mass density value equals +******** + ** in material number ****, **********. Value should be non-negative. Execution terminated.

Self-explanatory.

FMAT - Input error in subroutine FMAT. Poisson value equals ± .******** + ** in material number ******, **********. Value should be greater than -1.0 and less than 1.0. Execution terminated.

Self-explanatory.

FMAT - Input error in subroutine FMAT. Rigidity value equals ± . ********E + ** in material number ******, **********. Value should be greater than 1.0. Execution terminated.

Self-explanatory.

FMAT - Input error in subroutine FMAT. Thermal coefficient value equals ± .********E + ** in material number ******, **********. Value should be greater than -1.0 and less than 1.0. Execution terminated.

Self-explanatory.

FMAT - Input error in subroutine FMAT. Value of Young's modulus equals ± .********E + ** in material number ******, **********. Value should be greater than 1.0.

Self-explanatory.

FMAT - Error message from subroutine FMAT. Attempt to delete material number ****** using lock code **. Incorrect lock code, request ignored.

Self-explanatory.

FMAT - Error message from subroutine FMAT. Attempt to delete material that was not on material tape. Material number ******. Material identification is **********. Input code is ***. Request ignored.

Self-explanatory.

FMAT - Error message from subroutine FMAT. Attempt to revise material number ****** using lock code **. Input lock code does not match tape lock code for this material. Revisions or deletions not allowed without proper lock code. Execution terminated.

Self-explanatory.

FMAT - Error message from subroutine FMAT. Additions requested exceed capacity of material tape. Maximum number of materials cannot exceed ***.

Self-explanatory.

FMAT - Error message from subroutine FMAT. Request for print of material that was not on tape. Material number ******. Material identification is **********. Input code is ***. Request ignored.

Self-explanatory.

FMAT - Error message from subroutine FMAT. Unrecognizable data input code. Legal codes are PI, PO, I. O, P, OUT, ALL, SEE, SUM. Material number ******. Material identification is **********. Input code is ***. Execution terminated.

Self-explanatory.

FMAT - Error message from subroutine FMAT. Number of requests received is zero.

Number of requests must not be zero. Value of zero indicates improper operation of program.

FMAT - Error message from subroutine FMAT. Attempt to input plastic data only for material which was not on tape. Material number ******. Material identification is **********. Input code is ***. Request ignored.

Usage of an input code of "P" requires that the material to be revised already exists in the material library.

FMAT - New material tape not generated. All revisions and/or deletions requested by this case have been ignored.

Due to a previous error, generation of a new material library has been abandoned. Execution will be terminated.

FORMIN - Unexpected label card read - point *****.

Input section label card encountered while reading table form input. Point reflects entry now being processed.

FORMIN - Repeat for first point ignored.

Repeat option on table forms of report form input cannot be used for first value entered.

FRED - There is a mistake in the coordinates for this transformation, we will calculate the remaining in spite of this.

An error has occurred in generating a grid point axes transformation matrix. Execution will continue.

F6211 - The integral of (LN(A+B*X)/X) DX is not allowed for A+B*X=0. A = $\pm$.******E $\pm$ **.,
B = $\pm$.*******E $\pm$ **, X = $\pm$.*******E $\pm$ **

Natural log of zero is undefined.

INDECK - .USER04. input matrix ****** is not a valid deck (word count error).

The specified matrix does not qualify as a valid interpreted input deck.

INDECK - .USER04. input matrix ****** is not a valid deck (compression error).

The specified matrix does not qualify as a valid interpreted input deck.

INPUT - Input error, number of directions of grid points not equal to number of directions of transformation matrix. Execution terminated.

Order of grid point axes transformation matrices must be equal to three.

INPUT - Input error, number of reference points input exceeds ****.

Program cannot accommodate more than the given number of input points.

INPUT - Label card error ******.

Input card read should have been label card. Execution will be terminated.

LOGLO - Logical input error - matrix ****** cannot be generated by .USER04. module due to suppression of fourth input matrix. Execution phase suppressed. Input processing continuing.

The incremental matrices cannot be generated because the input displacement or stress matrix has been suppressed.

PDISP - Input section ****** matrix not generated due to prescribed displacement conditions .NE. 1 and .LT. Load conditions input.

The Prescribed Displacement matrix has not been generated because of an illegal combination of external load conditions and prescribed displacement conditions.

PHASE1 - Unexpected blank label card encountered.

Card read should have contained an input section label. Input processor will attempt to continue.

PHASE1 - No option has been selected for request number *** of material library.

Self-explanatory.

PHASE1 - More than one option has been selected for request number *** of material library. Only the first selection will be retained.

Self-explanatory.

PHASE1 - Maximum number of load conditions allowed is 100. This problem contains ****.

Self-explanatory.

PHASE1 - Load condition *** sub-label is incorrect. Program cannot distinguish between load conditions.

Load condition sub-label in report form input is in error.

PHASE1 - Illegal MODAL card encountered. Card will be ignored.

A MODAL card has been found while reading an input section for which no MODAL card has been defined.

PHASE1 - Due to previously encountered error condition this section is being skipped. Program will flush data deck until next recognizable input section is encountered.

PHASE1	-	Unrecognizable input section. Input section label has been read which is undefined in input processor.
PHASE1	-	Due to above error message this section will be omitted and check card inserted. Self-explanatory.
PHASE2	-	Number of entries read for this section, *****, does not agree with number that was to be read, *****. Actual number read will be used. Self-explanatory.
PHASE2	-	This section has either been omitted or flushed by phase one error. In either case this section is considered critical and execution will not be allowed. Self-explanatory.
PHASE2	-	Due to the omission of this section the following sections may be ignored - ****** ****** ****** ... The final processing of certain sections requires data from other sections which by omission or other input error are not present.
PHASE2	-	This section is to be merged with ****** and ****** for which values have been assigned by both for point number *****. Two values cannot be assigned to the same point. Neither value will be used. Self-explanatory.
PHASE2	-	This section is to be merged with ****** and ****** for which modal cards have been encountered for both. Two values cannot be assigned to the same point. Both modal cards will be ignored. Self-explanatory.
PHASE2	-	Number of elements read ***** is greater than 9999. Number of elements will be set at 9999. Self explanatory, execution will be suppressed.

PHASE2 - No end or check card has been found. Check card will be inserted, suppressing execution.

Self-explanatory.

PHASE2 - Due to above error condition check card will be inserted. Execution will be suppressed.

Self-explanatory.

PHASE2 - Internal tape error has occurred. Processing abandoned.

Report form input preprocessor cannot retrieve information stored on a scratch data set.

PLUG1 - Value of sin (alpha) is zero - run terminated.

Element defining points are in error for Quadrilateral Thin Shell Element.

PLUG5 - For I = XX and N = XX integral does not converge.

No convergence has been obtained for the given integral calculated by the Romberg technique in the Toroidal Ring Element.

PLUG5 - Maximum number of iterations reached in Romberg integration routine.

Convergence was not obtained in 15 iterations for an integral in the toroidal thin shell element. Processing will continue, using 15 iteration result.

PRINT5 - Toroidal ring element with coordinates
R1 = ± .********E ± **, R2 = ± .********E ± **,
Z1 = ± .********E ± **, Z2 = ± .******** ± **
is not diagonally dominant and should be subdivided.

Element stiffness matrices must be diagonally dominant.

P7?RT - PLUG7 error - third point to define plane was not given - input error.

Three element defining points are required for the frame element, the third supplying definition of the plane.

TRAIC - Subroutine MINV has determined array GAMABQ to be singular, execution terminated by subroutine TRAIC.

Transformation matrix to system coordinates in triangular cross-section ring element cannot be inverted, usually because three element defining points do not define a triangle.

USO4A - Available scratch data sets **** is less than the required 4.

The .USER04. module requires at least four scratch data sets. The addition of more data sets is required by the program.

USO4A - Input routine, core storage required ****** exceeds that available ****** to displacement method matrix generator.

Blank common work area is not large enough for processing input.

USO4A - Report routine core storage required ****** exceeds that available ****** to displacement method matrix generator.

Blank common work area is not large enough for processing report form input data.

USO4A - Grid point loads matrix storage required ****** exceeds that available ****** to displacement method matrix generator.

Blank common work area is not large enough for generation of grid point loads matrix.

USO4A - Reduction of transformation matrixes storage ****** exceeds that available to displacement method matrix generator.

Blank common work area is not large enough for generation of reduction transformation matrix.

USO4A - Element generation core storage required ****** exceeds that available ****** to displacement method matrix generator.

Blank common work area is not large enough for generation of element matrices.

USO4A - Assembly transformation matrix size ****** exceeds limit ****** of MAGIC system.

Self-explanatory.

USO4A - Grid point load matrix size ****** exceeds limit ****** of MAGIC system.

Self-explanatory.

USO4A - Reduction transformation matrix size ****** exceeds limit ****** of MAGIC system.

Self-explanatory.

USO4A - Stiffness matrix size ****** exceeds limit of MAGIC system.

Self-explanatory.

USO4A - Stress matrix size ****** exceeds limit ****** of MAGIC system.

Self-explanatory.

USO4A - Number elements size ****** exceeds limit ****** of MAGIC system.

Self-explanatory.

USO4A - Output matrix ****** will be a duplicate of input matrix ******.

The user is saving the interpreted input deck when he already has an interpreted input matrix.

USO4B - Element sort routine core storage required ****** exceeds that available ****** to displacement method matrix generator.

Blank common work area is not large enough for output of generated matrices.

APPENDIX VII

EXAMPLE STATIC AND STABILITY INSTRUCTION SEQUENCES

A. STATICS ANALYSIS INSTRUCTION SEQUENCE

```
1    7                Columns

C ---     GENERATE ELEMENT MATRICES
C
          ,MAT,,XLD,TR,,KEL,FTEL,SEL,STEL,,,SC,EM,=,,,.USER04.
C
C ---     ASSEMBLE ELEMENT STIFFNESS MATRICES
C
          KELA = EM.ASSEM.SC,(1)
C
C ---     ASSEMBLE ELEMENT APPLIED LOAD MATRICES
C
          FTELA = EM.ASSEM.SC,(4)
C
C ---     REDUCE ASSEMBLED STIFFNESS MATRIX
C
          KO, KNO = KELA .DEJOIN. (SC(5,1),1)
          KCO, STIFF = KNO .DEJOIN. (SC(5,1),0)
          PRINT(FORCE,DISP,,)  STIFF
C
          EXTRACT LOAD SCALAR AND APPLY TO ELEMENT LOADS
C
          LSCALE,LOADS = XLD .DEJOIN . (1,1)
          FTELS = FTELA .MULT. LSCALE
C
C ---     TRANSFORM EXTERNAL LOADS TO 0-1-2 ASSEMBLED
C ---     SYSTEM AND FORM TOTAL LOAD COLUMNS
C
          LOADO = TR .MULT. LOADS
          TLOAD = LOADO .ADD. FTELS
          TL,TLOADR = TLOAD . DEJOIN. (SC(5,1),1
C
C ---     SOLVE FOR DISPLACEMENTS
C
```

```
          XX = STIFF.SEQEL.TLOADR
          TR0,TR12 = TR.DEJOIN.(SC(5,1),1)
          X = TR12.TMULT.XX
          XO = TR.MULT.X
C
C ---     SOLVE AND PRINT ELEMENT STRESSES AND FORCES
C
          STRESP = EM, XO,.STRESS. (4,)
          FORCEP = EM, XO,.FORCE.  (4,)
C
C ---     SOLVE FOR SYSTEM REACTIONS
C
          REACTS = KELA.MULT.XO
          REACTP = REACTS.SUBT.TLOAD
C
C ---     PRINT ELEMENT APPLIED LOADS, EXTERNAL LOADS,
C ---     DISPLACEMENTS AND REACTIONS IN ENGINEERING FORMAT
C
          GPRINT(4,,,FX.FY.FZ.MX.MY.MZ,SC,TR) FTELA
          GPRINT(4,,,FX.FY.FZ.MX.MY.MZ,SC,) LOADS
          GPRINT(2,,,U.V.W.THETAX.THETAY.THETAZ,SC,) X
          GPRINT(1,,,FX.FY.FZ.MX.MY.MZ,SC,TR) REACTP
```

B. STABILITY ANALYSIS INSTRUCTION SEQUENCE

```
C ---     GENERATE ELEMENT MATRICES
          ,MAT,INTP,LXD,TR,,KEL,FTEL,SEL,STEL,,,SC,EM,=,,,.USER04.
C
C ---     ASSEMBLE ELEMENT STIFFNESS AND ELEMENT LOAD MATRICES
C
          KELA = EM.ASSEM.SC,(1)
          FTELA = EM.ASSEM.SC,(4)
C
C ---     REDUCE ASSEMBLED STIFFNESS MATRIX
          KO,KNO = KELA.DEJOIN.(SC(5,1),1)
          KCO,STIFF = KNO.DEJOIN. (SC(5,1),0)
          PRINT(FORCE,DISP.,,)STIFF
C
C ---     EXTRACT LOAD SCALARS AND APPLY TO ELEMENT LOADS
C
          LSCALE,LOADS = XLD.DEJOIN.(1,1)
          FTELS = FTELA.MULT.LSCALE
```

```
C
C ---     SOLVE FOR TOTAL LOADS
          LOADO = TR.MULT.LOADS
          TLOAD = LOADO.ADD.FTELS
          TL,TLOADR = TLOAD.DEJOIN.(SC(5,1),1)
C
C ---     CREATE FLEXIBILITY MATRIX
C
          FLEX = STIFF.INVERS.
          PRINT(DISP,FORCE,,)FLEX
C
C ---     SOLVE FOR DISPLACEMENTS
          XR = FLEX.MULT.TLOADR
          TR0,TR12 = TR.DEJOIN.(SC(5,1),1)
          X = TR12.TMULT.XR
          X0 = TR.MULT.X
C
C ---     SOLVE FOR ELEMENT STRESSES
C
          STRESS = EM,X0.STRESS. (4,)
C
C ---     GENERATE ELEMENT INCREMENTAL STIFFNESS MATRIX
C
          ,,,,,,,,,,NEL,,,EL, = ,INTP,,STRESS.USER04.
C
C ---     ASSEMBLE AND REDUCE INCREMENTAL MATRICES
C
          INCRA = EL.ASSEM.SC,(3)
          IO,INO = INCRA.DEJOIN.(SC(5,1),1)
          ICO,INCR = INO.DEJOIN.(SC(5,1),0)
          PRINT(,,,)INCR
C
C ---     CREATE EIGEN MATRIX
C
          EIG = FLEX.MULT.INCR
          PRINT(,,,)EIG
C
C ---     CALCULATE AND PRINT E-VALUES AND E-VECTORS
C
          EVALUE,EVECTR,, = EIG,.EIGEN1.(5,,,)
          GPRINT(3,,,,SC,TR12)EVECTR,EVALUE
C
C ---     PRINT ELEMENT APPLIED LOADS, EXTERNAL LOADS, AND
C ---     DISPLACEMENTS IN ENGINEERING FORM
C
          GPRINT(4,,,FX.FY.FZ.MX.MY.MZ,SC,TR) FTELA
          GPRINT(4,,,FX.FY.FZ.MX.MY.MZ.SC,) LOADS
          GPRINT(2,,,U.V.W.THETAX.THETAY.THETAZ,SC,) X
```

APPENDIX VIII

SUBROUTINE DOCUMENTATION

Subroutine	Page No.
AGENDM	8. 12
AI	8.296
AJ	8.305
AK	8.299
AKNZ	8. 89
AM	8.300
ANALIC	8. 97
APRT	8.332
ASMAIC	8.110
ASMPIC	8.113
ASSEM	8. 48
ASSEMC	8. 50
ASSEMS	8. 51
ASSY2	8.314
AXTRA1	8.437
AXTRA2	8.225
AXTRA3	8.439
BACK	8.141
BCB	8.229
BCB12	8.421
BDEF	8.458
BEQSX	8. 93
BINT	8.298
BMATRX	8.390
BOUND	8.161
CCB	8.337
CC1	8.257
CC2	8.274
CC21	8.241
CDELPQ	8.246
CDF	8.279
CDFX	8.280
CDFY	8.281
CDM	8.261
CECC	8.293
CFFTS	8.282
CFFV	8.284
CFMASS	8.285
CFMTS	8.262
CFMV	8.264
CFP	8.275
CFPB	8.276
CHDEL1	8.247
CHEK	8.201
CHEQS	8. 88

Subroutine	Page No.
CK11	8.250
CK22	8.266
CK26	8.500
CMASS	8.456
CMMASS	8.258
COEF	8.306
COLMRD	8. 72
COLREP	8. 42
CONTRL	8.153
COPYDK	8.154
CSTF	8.277
CSTM	8.259
CTCQ	8.292
CTGB	8.269
CTGRB	8.273
CTGRM	8.256
CTOGB	8.272
CTOGM	8.255
CTS	8.291
CT11	8.253
DCD	8.315
DECODE	8. 76
DECOMP	8.138
DEFLEX	8.212
DEJNC	8. 47
DEJNR	8. 46
DEJOIN	8. 44
DIRCOS	8.418
DISPIC	8.117
DISPL1	8. 81
DISPMT	8.473
DISPPR	8. 78
DMATRX	8.391
DPQINT	8.324
DSPCIC	8.115
DTAPR	8.317
EIG	8. 40
EIGB	8. 39
EIGPPR	8. 79
EIG1	8. 35
ELEM	8.163
ELMAT	8.450
ELPLUG	8.218
ELPRT	8.441
ELREAD	8. 68
ELTEST	8.234

Subroutine	Page No.
EPRINT	8. 65
EPRT	8.347
ERIC	8.413
ESCONT	8. 95
EXPCOL	8.354
EXPSIX	8.355
FARCIC	8.119
FBMP18	8.432
FCURL	8.393
FELEM	8.214
FF100	8.308
FGRLDS	8.170
FINP22	8.435
FJAB	8.302
FLOADS	8.204
FMAT	8.172
FORCE	8. 59
FORCE1	8. 61
FORCE2	8. 63
FORMIN	8.192
FRED	8.159
FREEUP	8. 70
FTELQ	8.425
FTR	8.205
FT26	8.504
FWD	8.140
F4	8.385
F5	8.386
F6	8.387
F6211	8.304
F6219	8.303
F89	8.307
GAMMAG	8.471
GAMMAS	8.472
GAMMAT	8.392
GELS	8.128
GENSM	8.498
GEOMD	8.470
GPRINT	8. 73
GPRNT1	8. 74
HARGEN	8.179
HARGE1	8.182
HARMA	8.181
HDECO	8. 84
HLOGEN	8.491
HMASSG	8.492

Subroutine	Page No.
HRAICS	8.487
HRAIES	8.490
HRAIKS	8.488
HRAISS	8.489
HSUM	8. 83
HTCO	8.235
HTHGEN	8.493
IDNTC	8. 34
IDNTR	8. 33
IFAC	8.301
INCRE	8.288
INDECK	8.152
ININT	8.210
INPTIC	8.103
INPTI1	8.105
INPUT	8.155
INSTWT	8. 10
INSTO4	8. 13
INSTO5	8. 14
INSTO6	8. 16
INSTO7	8. 18
INST43	8. 20
INST44	8. 22
INST45	8. 24
INST56	8. 26
INST60	8. 28
KALCON	8.108
KCALC	8. 95
KMPY	8.412
KOBLIQ	8.422
LAG	8.169
LATCH	8.191
LISTIC	8.106
LOC	8.233
LOGFLO	8.147
MAB	8.227
MABC	8.242
MABS	8.510
MASS20	8.480
MATB	8.231
MATBS	8.511
MATCH	8.167
MATI60	8.254
MATI70	8.271
MATPR	8.318
MATPRT	8. 80
MATSUP	8. 30
MBS	8.509
MFSD	8.129
MINV	8.224
MPOWJF	8.474
MPRD	8.294

Subroutine	Page No.
MSB	8.228
MSBB	8.132
MSTR	8.382
MTDS	8.130
MULTF	8.400
NEWFT	8.244
NEWFT1	8.320
NTEST	8.144
NULL	8. 43
OPEN	8.200
OUTINT	8.202
OUTMAT	8.443
PASSYM	8.481
PDISP	8.198
PFFTS	8.342
PFFV1	8.344
PFMASS	8.313
PFMTS	8.329
PFMV1	8.331
PFP	8.339
PHASE1	8.194
PHASE2	8.192
PKF	8.335
PKM	8.325
PLAS2D	8.348
PLMX	8.394
PLUG1	8.237
PLUG10	8.452
PLUG13	8.465
PLUG14	8.398
PLUG17	8.414
PLUG18	8.427
PLUG2	8.309
PLUG20	8.467
PLUG22	8.433
PLUG23	8.482
PLUG25	8.494
PLUG5	8.378
PLUG6	8.350
PLUG7	8.286
PLUG8	8.403
PL6PRT	8.377
PMMASS	8.312
PNC1	8.345
PNG1	8.346
POOF	8.401
PRINT5	8.397
PRT1	8.265

Subroutine	Page No.
PSTF	8.340
PSTM	8.327
PTBF	8.349
PTBM	8.322
PTFGS	8.334
PTMGS	8.323
PUTL4	8.8
PUTL5	8.9
P1PRTA	8.248
P1OFP	8.461
P1OTM	8.459
P14PRT	8.402
P1718M	8.423
P2OFP	8.476
P2OFT	8.478
P7PRT	8.289
P8MASS	8.406
QUADI	8.388
RDCLOCK	8.134
REACTP	8.77
RECTIC	8.122
REC1	8.145
REC3	8.221
REC4	8.222
REFORM	8.184
REGE2	8.37
REPLAS	3.82
ROMBER	8.388
SCRLM	8.395
SELQ	8.424
SHIFT	8.178
SINVIC	8.131
SIZEIC	8.99
SOLVE	8.396
SQUISH	8.216
STORIC	8.101
STRCIC	8.125
STRESS	8.53
STRES1	8.55
STRES2	8.57
STRPRT	8.67
STRPIC	8.127
STRSIC	8.123
STRS26	8.507
SUBINT	8.408
SUMDIS	8.86
SUMSTR	8.85
SYMPRT	8.232
SZAL26	8.506

Subroutine	Page No.
TCONTX	8. 90
TESTJ	8.357
TFPPRT	8.364
TFSPRT	8.374
TFTPRT	8.366
TIEPRT	8.360
TIKPRT	8.362
TIMER	8.135
TIMPRT	8.372
TISPRT	8.368
TPRD	8.295
TRAAE	8.469
TRAIC	8.356
TRAIE	8.359
TRAIFP	8.363
TRAIFS	8.373
TRAIFT	8.365
TRAIK	8.361
TRAIM	8.371
TRAIS	8.367
TRAIST	8.375
TRAITS	8.369
TRCPRT	8.358
TR18ST	8.431
TSTPRT	8.376
TSUM	8. 32
TSYS	8.206
TTRI	8. 91
TTSPRT	8.370
TUDJAC	8.502
TXOUT	8.477
UN1TNM	8.107
USO4	8.142
USO4A	8.149
USO4B	8.208
US461	8.446
US462	8.448
US463	8.449
VOL	8.455
WAVEFM	8.136
WCLOCK	8.133
XCALK	8. 94
ZMRD	8.410
ZTRD	8.411

1. Subroutine Name: PUTL4

2. Purpose:

This routine converts an array of B C D characters into an array containing a valid matrix name with an appropriate sign addended.

3. Equations and Procedures:

The array to contain the valid matrix name is first filled with B C D blanks. Next all blank characters are compressed out of the array which is to be converted to a matrix name. If the first location of the input array contains a B C D plus sign or a B C D minus sign, an appropriate flag is initialized. The first location of the input array is then checked against an array containing all the alphabetic characters. When a match is found, all the characters of the input array except for the possible B C D plus or B C D minus in the first location are placed on the output array. If a slash is encountered a utility routine is called to determine the subscript. The subscript is then placed on the seventh location of the output array. An integer one is placed in the eighth location of the output array. If a slash is not found, an integer plus one or a minus one is placed in the seventh location of the output array. Finally an integer zero is placed in the eighth location of the output array.

4. Input Arguments:

N - length of CARD array
CARD - array of B C D characters to be converted to a new name

5. Output Arguments:

XNAME - an eight character name with a sign in the seventh position and an integer one or zero in the last position

ERROR - logical flag indicating presence of an error

6. Error Returns:

If the first character of the matrix name excluding the possible B C D plus of B C D minus is not an alphabetic character, an error condition results. Also if the matrix name is longer than six characters long, an error condition results.

7. Calling Sequence:

PUTL4 (CARD, XNAME, N, ERROR)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Subroutine Required:

PUTL1
PUTL5

1. Subroutine Name: PUTL5

2. Purpose:

   This routine extracts the subscript of a matrix name

3. Equations and Procedures:

   The abstraction instruction card is scanned beginning in the first card column follow the first slash after a matrix name. The scan continues until a second slash is encountered. The number of nonblank characters is counted. This sub-field should contain a one to four digit decimal integer. A utility routine is called which converts this sub-field to a binary integer. Control is then returned to the calling routine.

4. Input Arguments:

   CARD - an array containing the last card image

   LIMIT - length of the array containing the card image

5. Output Arguments:

   ERROR - logical flag indicating the presence of an error

   INTG - binary integer is in this case the subscript of the matrix name

6. Error Returns:

   An error condition occurs when the second slash cannot be located or when no digits can be found between the slashes.

7. Calling Sequence:

   PUTL5 (CARD, INTG, LIMIT, ERROR)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Subroutine Required:

    PUTL3

1. Subroutine Name: INSTWT

2. Purpose:

This routine writes the looped instructions on the preprocessor utility data set NPREP.

3. Equations and Procedures:

a. Algorithm
The coded looped instructions in the working storage array is analyzed and stored in a temporary array, decoded. Subscripts of matrix names are incremented if necessary. Then the matrix names and their subscripts are stored in the temporary array. Scalars are packed and stored in the temporary array. The temporary array is written on the preprocessor scratch data set NPREP. This process is repeated for all the instructions in the range of the REPEAT loop and repeated for all the instructions in the range of the REPEAT loop and this sequence of instructions is repeated the specified number of times. Finally, when the REPEAT loop is satisfied, the loop flag is set back to .FALSE.

4. Input Arguments:

IBUFER - working storage array containing the analyzed instructions within the range of the REPEAT loop

ITEMP - temporary array containing the record to be written on the preprocessor scratch data set NPREP

NOSTAT - the number of statements in the range of the REPEAT loop

NTIMES - the number of times the sequence of looped instructions is to be repeated

LOOP - logical flag indicating whether or not the instruction is in the range of the REPEAT loop

NPREP - the number of the preprocessor scratch data set NPREP

5. Output Arguments: None

6. Error Returns: None

7. Calling Sequence:

INSTWT (IBUFER, ITEMP, NOSTAT, NTIMES, LOOP, NPREP)

8. Input Tapes: None

9. Output Tapes:

The coded looped instructions are written on the preprocessor scratch data set NPREP.

10. Scratch Tapes: None

11. Subroutine User: INST

12. Subroutine Required: None

1. Subroutine Name: AGENDM

2. Purpose: To locate in the Agendum library the abstraction instructions specified by the user on the $INSTRUCTION control card in MAGIC.

3. Equations and Procedures: The name of the desired Agendum on the $INSTRUCTION card is passed to AGENDM by INST. The specified name is compared against all available agendum names in the TYPE array. If the specified option is a valid name then the agendum library is searched until the correct abstraction instruction sequence is found, if it is not found an error occurs. If it is found then NPIT is redefined to be NSETA and control is passed to INST.

4. Input Arguments:

OPTION	-	agendum name on $INSTRUCTION card
LENOP	-	length of agendum name on $INSTRUCTION card
NPIT	-	logical unit number defining system card reader
NSETA	-	logical unit number defining data set of agendum library
WORK	-	work storage

5. Output Arguments: None

6. Error Returns:

ERROR	-	TRUE, if the option specified on the $INSTRUCTION card is unavailable or unrecognizable.

7. Calling Sequence:

```
AGENDM(OPTION,LENOP,NPIT,NSETA,WORK,ERROR)
```

8. Input Tapes:

NSETA - agendum library

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

Total storage required is $4D4_{16}$ Bytes.

12. Subroutine User: INST

13. Subroutine Required: PUTL2

14. Remarks: MAGIC

1. Subroutine Name: INSTO4

2. Purpose:

   This routine analyzes the REPEAT abstraction instruction.

3. Equations and Procedures:

   The abstraction instruction is scanned starting in the card column following the left parenthesis, and continuing up until a comma is encountered. This sub-field should contain a B C D integer specifying the number of abstraction instructions to be included in the range of the REPEAT loop. This BCD integer is converted to a binary integer using a utility routine. The scan continues starting in the card column following the comma, and continuing up until the right parenthesis is encountered. This sub-field should contain a B C D integer specifying the number of times the sequence of abstraction instructions in the range of the REPEAT loop is to be repeated. A utility routine is called to convert the B C D integer to a binary integer. Finally the LOOP flag is set to .TRUE.

4. Input Arguments:

   CARD - an array containing the image of the last card read

   LIMIT - intermediate counter

   NOSTAT - the number of abstraction instructions to be repeated

   NTIMES - the number of times the sequence of abstraction instructions is to be repeated

   LOOP - logical flag indicating the presence of a loop

5. Output Arguments:

   ERROR - logical flag indicating the occurance of an error

6. Error Returns:

   An error condition occurs when the two sub-fields are omitted or when the comma and/or the right parenthesis are omitted.

7. Calling Sequence:

   INST04 (CARD, LIMIT, NOSTAT, NTIMES, LOOP, ERROR)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Subroutine Required:

    PUTL3

1. Subroutine Name: INST05

2. Purpose: To analyze the EPRINT instruction which is of the form:

   EPRINT(N,SZERO,NAMIN1)NAMIN2

3. Equations and Procedures: This subroutine uses the same procedure as all the other MAGIC instruction analyzers. The card image with blanks suppressed and starting one column to the right of the first (is broken into 4 fields as defined within successive delimiters.

Field Defined By		Checked For
(	,	Integer Scalar
,	,	Real Scalar
,	)	Matrix Name
)	Δ-blank	Matrix Name

   Each field is examined and checked in turn. Detection of an error results in an error return. If the card image for the instruction is syntactically correct, information required for execution is written on tape and control is returned to INST.

4. Input Arguments:

NPREP	-	output tape number
NOPC	-	opcode of instruction (05)
INSTNO	-	statement number of instruction
CARD	-	card image (starting in column to right of first (, blanks suppressed)
NONBLK	-	number of non-blank characters on card.

5. Output Arguments:

NUMIN	-	number of input matrices
ERROR	-	error control

6. Error Returns: The logical variable ERROR is set to .TRUE. if an error is detected and control returns to INST. Additional diagnostics and warnings are printed for invalid values of parameters and illegal suppression of parameters.

7. Calling Sequence:

   Call INST05(NPREP,NOPC,INSTNO,CARD,NONBLK,NUMIN,ERROR)

8. Input Tapes: None

9. Output Tapes: NPREP

10. Scratch Tapes: None

11. Storage Required:

SYMBOL(4)

Total Storage is $6D8_{16}$ Bytes.

12. Subroutine User: INST

13. Subroutine Required:

INSTFP
PUTL3
PUTL4

14. Remarks: This is a special instruction analyzer.

1. Subroutine Name: INST06

2. Purpose:

This routine analyzes the PLOT instruction.

3. Procedure:

The abstraction instruction card is scanned starting in the card column following the left parenthesis, and continuing up until a right parenthesis is encountered. This sub-field should contain a B C D integer greater than zero and less than 8. This field is checked for validity (PUTL3) and an error message is printed if the field is not valid. The scan continues starting with the card column after the right parenthesis until a comma or a blank character is encountered. This sub-field contains a matrix name. A utility routine is called PUTL4 which checks the validity of the name and converts this sub-field into the proper format. If the scan was interrupted by a blank, the scan is terminated. If the scan was interrupted by a comma, the scan continues in the first card column following the comma up until another comma or blank is encountered. The second field again should contain a valid matrix name and the matrix name is processed in the same manner in which the first matrix name was processed. This process is repeated at least three times or until finally a blank character is encountered at which point the scan is terminated. If the PLOT instruction is within the range of a REPEAT loop, the analyzed abstraction instruction is stored in the working storage array, later to be written on the preprocessor scratch data set NPREP. However, if the PLOT instruction is written in coded form on the preprocessor scratch data set NPREP.

4. Input Arguments:

NPREP - preprocessor scratch data set -- logical unit number
NOPC - operation code for PLOT instruction
ISTNO - instruction number from abstraction instruction card label field

CARD - array containing an B C D character/word of abstraction instruction card starting with card column following the left parenthesis

NONBLK - length of CARD array
LOOP - logical (if true then instruction is in range of a repeat loop)

JWORK - working storage array
IPNT - location counter for working storage array

5. Output Arguments:

ERROR - error return indicator
NUMIN - number of input matrices in the instruction

6. Error Returns:

An error condition occurs when:

a) an invalid character in a matrix name
b) wrong SYNTAX
c) number of matrices not correct for value of input integer
d) invalid character or value in integer field

7. Calling Sequence:

INST06 (NPREP,NOPC,ISTNO,CARD,NONBLK,MUMIN,ERROR,LOOP,IWORK,IPNT)

8. Subroutine User: INST

9. Subroutine Required:

PUTL4
PUTL3

1. Subroutine Name: INST07

2. Purpose: To analyze the GPRINT instruction which is of the form:

```
GPRINT(NPRT,EZERO,ROWL,COL1.COL2.COL3. ... .COL12,
     TSYS,TR)XX1,XX2
```

3. Equations and Procedures: This subroutine uses the same procedure as all the other MAGIC special instruction analyzers. The card image with blanks suppressed and starting one column to the right of the first (is broken in 3 groups. The first group is checked for the 3 fields defined by scalars.

Field	Checked For
(,	Scalar
, ,	Scalar
, ,	Scalar

Next a check is made for the 12 column labels. These labels are positional and may be suppressed. After the labels have been determined, the third group is checked for matrix names. Two, three, or four matrices may be specified depending on use.

Field		Checked for
end of labels	,	Matrix Name
,	)	Matrix Name
)	,	Matrix Name
,	Δ-blank	Matrix Name

Each field is checked in turn and detection of an error results in an error return. If the card image for the instruction is syntactically correct, information required for execution is written on tape. Control is returned to INST.

4. Input Arguments:

NPREP	-	output tape number
NOPC	-	opcode of instruction (04)
ISTNO	-	statement number of instruction
CARD	-	card image (starting in column to right of first (, blanks suppressed)
NONBLK	-	number of non-blank characters on card

5. Output Arguments:

NUMIN - number of input matrices
ERROR - error control

6. Error Returns: Logical variable ERROR is set to .TRUE. if an error is detected and control returns to INST. Additional diagnostics are printed for illegal values of parameters, invalid specification of matrices and illegal specification of column headers.

7. Calling Sequence:

```
Call INST04(NPREP,NOPC,INSTNO,CARD,NONBLK,NUMIN,ERROR)
```

8. Input Tapes: None

9. Output Tapes: NPREP

10. Scratch Tapes: None

11. Storage Required:

SYMBOL(3)
TYMBOL(4)

$D88_{16}$ Bytes

12. Subroutine User: INST

13. Subroutine Required: INSTFP, PUTL3, PUTL4

14. Remarks: This is a special instruction analyzer.

1. Subroutine Name: INST43

2. Purpose: To analyze the .DEJOIN. instruction.

A1,A2 = B.DEJOIN.(C(I,J),KODE)
A1,A2 = B.DEJOIN.(K,KODE)

3. Equations and Procedures: This subroutine uses the same procedure as all the other analyzers in MAGIC. The card image with blanks suppressed and starting in column 7 is broken into 6 fields as defined within successive delimiters.

Field Defined By		Checked For
Column 7	,	Matrix Name
,	=	Matrix Name
=	.	Matrix Name
o	o	Not Checked
o	(	Not Checked
(	)	Checked For Matrix Name and 3 Scalars or 2 Scalars

Each field is examined and checked in turn. Detection of an error results in an error return. If the card image for the instruction is symtactically correct, information required for execution of the instruction is written on tape and control is returned to INST.

4. Input Arguments:

NPREP - output tape number
NOPC - opcode of instruction (43)
ISTNO - statement number of instruction
CARD - card image (starting in column 7, blanks suppressed)
NONBLK - number of non-blank characters in card

5. Output Arguments:

NUMOT - number of output matrices
NUMIN - number of input matrices
NUMSC - number of scalars
ERROR - error control

6. Error Returns: Logical variable ERROR is set to .TRUE. if an error is detected in this routine and a return is made to INST. Additional messages are printed out for invalid matrix names and invalid indices.

7. Calling Sequence:

```
Call INST43(NPREP,NOPC,ISTNO,CARD,NONBLK,NUMOT,NUMIN,NUMSC,
           ERROR)
```

8. Input Tapes: None

9. Output Tapes: NPREP

10. Scratch Tapes: None

11. Storage Required:

MATRIX(7,4)
SYMBOL(6)
INDEX(3)

Total Storage is $A58_{16}$ Bytes.

12. Subroutine User: INST

13. Subroutine Required: PUTL3, PUTL4

14. Remarks: This is an arithmetic type instruction analyzer.

1. Subroutine Name: INST44

2. Purpose: To analyze the ANALIC instruction which is of the form

DISP,STR,FOR,REC = TR,SC,EM,XLD,PD,KR,PR,TRANS,W,.ANALIC.
(KALC,NNOM,NRSELM)

3. Equations and Procedures:

This subroutine uses the same procedure as the other MAGIC analyzers. The card image with blanks suppressed, starting in column 7 is broken into 16 fields as defined inside successive delimiters. Optionally, 1 to 4 output matrices may be specified. Any number of input matrices may be indicated. Each matrix and scalar present is indicated by a value of 1 in the NUMSCL array. Suppressed matrices and scalars are denoted by a 0 in NUMSCL. Each field is examined and checked in turn. Detection of an error results in an error return. If the card image for the instruction is syntactically correct, information required for execution is written on tapes. Control is returned to INST.

4. Input Arguments:

NPREP	- Output tape number
NOPC	- OP code of instruction (44)
ISTNO	- Statement number on instruction
CARD	- Card image (starting in column 7, blanks suppressed)
NONBLK	- Number of non-blank characters on card

5. Output Arguments:

NUMOT	- Number of output matrices
NUMIN	- Number of input matrices
NUMSC	- Number of scalars (16)
ERROR	- Error flag

6. Error Returns:

ERROR - Set to true, if an error is detected inside INST44 routine

7. Calling Sequence:

```
Call INST44(NPREP,NOPC,INSTNO,CARD,NONBLK,NUMOT,NUMIN,NUMSC,
            ERROR)
```

8. Input Tapes: None

9. Output Tapes: NPREP

10. Scratch Tapes: None

11. Storage Required: 916_{16} bytes

12. Subroutine User: INST

13. Subroutines Required:

PUTL3, PUTL4

14. Remarks: None

1. Subroutine Name: INST45

2. Purpose:

This routine analyzes the abstraction instruction. EIGEN2.

3. Equations and Procedures:

The field consisting of nonblank characters of the of the abstraction instruction card is scanned beginning in card Column 7 and continuing up until a comma is encountered. This sub-field is tested for validity as a matrix name. This process is repeated until four commas have been encountered and four matrix names read. The scan begins again in the first card column following the last comma and continues up until an equal sign is encountered. This sub-field is tested for validity as a matrix name. The scan begins again in the card column following the equal sign and continues until a period is encountered. This bus-field is tested for validity as a matrix name. The scan begins again with the card column following the first period until a second period is encountered. This sub-field is ignored. The scan begins again following the period and continues until a left parenthesis is encountered. At this point a conditional test is made. The test determines whether or not a matrix name exists in the sub-field. If there is a matrix name, a test is made on its validity as a matrix name. If the matrix name is valid, the number of input matrices is set equal to 2. If a name does not exist in the sub-field, the number of input matrices is set to 1. The scan begins again in the card column following the left parenthesis and continues until a comma is encountered. The sub-field is converted to a binary integer. This process is repeated until four integers, separated by commas, have been read. The scan begins in the card column following the last comma and continues until another comma has been encountered. This sub-field is converted to a floating point number. The scan begins again in the card column and continues until a right parenthesis is encountered. The sub-field is also converted to a floating point number. This completes the abstraction instructuion analysis. If it is determined that this abstraction instruction is within the range of the REPEAT loop, the instruction is stored in the working storage array in coded form. However, if it is determined that the analyzed instruction is not within the range of the REPEAT loop, the instruction is written in coded form on the preprocessor scratch data set, NPREP.

4. Input Arguments.

NPREP - the preprocessor utility data set

NOPC - the number of the operation code

ISTNO - the statement number of this abstraction instruction

CARD - an array containing the last card image

NONBLK - the number of nonblk characters in this field

NUMOT - the number of output matrices in this instruction

NUMIN - the number of input matrices in this instruction

NUMSC - the number of scalars in this instruction

LOOP - logical flag indicating whether this instruction is within the range of the REPEAT loop

IWORK - working storage array

IPNT - an integer pointer indicating the location of the next word of the working storage array

5. Output Arguments:

ERROR - logical flag indicating the presence of an error

6. Error Returns:

An error condition occurs when an error is returned from a subroutine, that is, when a matrix name contains an invalid character, or too many characters, or when a sub-field which is to be converted to an integer or real number does not contain all B C D integers.

7. Calling Sequence:

```
INST45 (NPREP,NOPC,ISTNO,CARD,NONBLK,NUMOT,NUMIN,NUMSC,ERROR,
LOOP,IWORK,IPNT)
```

8. Input Tapes: None

9. Output Tapes:

The statement number, the operation code, the number of input matrices, the number of output matrices, the number of scalars, the matrix names and the scalars are written on the preprocessor scratch data set, NPREP.

10. Scratch Tapes: None

11. Subroutines Required:

PUTL3
PUTL4
INSTFP

1. Subroutine Name: INST56

2. Purpose: This routine analyzes the abstraction instruction DJOIN.

3. Equations and Procedures:

This field of nonblank characters of the abstraction instruction card is scanned beginning in card column 7 and continuing up until a comma is encountered. This sub-field is tested for validity as a matrix name. The scan begins again the in card column following the comma and continues up until an equal sign is encountered. This sub-field is tested for validity as a matrix name. The scan begins again in the card column following the equal sign is encountered and continues up until a period is encountered. This sub-field is tested for validity as a matrix name. The scan again begins in the card column following the period upo until a second period is encountered. This sub-field is ignored. The scan continues until a left parenthesis is encountered. This sub-field is ignored. The scan starts again and continues up until a comma is encountered. This sub-field is converted into a binary integer. The scan resumes until a right parenthesis is encountered. This sub-field is converted to a binary integer. If this instruction is determined to be within the range of the REPEAT loop, the instruction in coded form is stored in the working storage array. However, if the analyzed instruction is determined not to be within the range of the REPEAT loop, the instruction in coded form is written on the preprocessor scratch data set, NPREP.

4. Input Arguments:

NPREP - the number of the preprocessor utility data set

NOPC - the number of the operation code

ISTNO the statement number of this instruction

CARD - an array containing the last card image

NONBLK - the number of nonblank characters in the array

LOOP - logical flag indicating whether or not the analyzed instruction is within the range of the REPEAT loop

IWORK - working storage array

IPNT - an integer pointer indicating the location of the next word in the working storage array

5. Output Arguments:

NUMOT - the number of output matrices in this instruction
NUMIN - the number of input matrices in this instruction
NUMSC - the number of scalars in this instruction
ERROR - a logical flag indicating the presence of an error

6. Error Returns:

An error condition occurs when an error is returned from a subroutine, that is, when a matrix name contains an invalid character or when one of the sub-fields which is to be converted to a binary integer does not contain all BCD integers.

7. Calling Sequence:

INST56 (NPREP,NOPC,ISTNO,CARD,NONBLK,NUMOT,NUMIN,NUMSC,LOOP, IWORK,IPNT)

8. Input Tapes: None

9. Output Tapes:

The statement number, the operation code, the number of output matrices, the number of input matrices, the number of scalars, the matrix names and the scalars are written on the preprocessor utility data set, NPREP.

10. Scratch Tapes: None

1. Subroutine Name: INST60

2. Purpose: To analyze instructions of the form

(±)NAMOUT = ±NAMIN1, ±NAMIN2.OPCODE.(NPRT,EZERO)

.FORCE. and .STRESS. are presently of this form.

3. Equations and Procedures: The subroutine uses the same procedure as all other analyzers in MAGIC. The card image with blanks suppressed, and starting at column 7 is broken into 7 fields as defined inside successive delimiters.

Field		Checked For
Column 7 -------	=	Matrix Name
= -------	,	Matrix Name
,	.	Matrix Name
.	.	Not Checked
.	(	Not Checked
(	,	Integer
,	)	Real Number

Each field is examined and checked in turn. Detection of an error results in an error return. If the card image for the instruction is syntactically correct, information required for execution is written on tape. Control is returned to INST.

4. Input Arguments:

NPREP	-	output tape number
NOPC	-	opcode of instruction (61 or 62)
ISTNO	-	statement number on instruction
CARD	-	card image (starting in column 7, blanks suppressed)
NONBLK	-	number of non-blank characters in card

5. Output Arguments:

NUMOT	-	number of output matrices
NUMIN	-	number of input matrices
NUMSC	-	number of scalars
ERROR	-	error control

6. Error Returns: Logical variable ERROR is set to .TRUE. if an error is detected in this routine and control returns to INST. Additional messages printed out for illegal values of scalars NPRT and EZERO.

7. Calling Sequence:

```
Call INST60(NPREP,NOPC,ISTNO,CARD,NONBLK,NUMOT,NUMIN,NUMSC,
            ERROR)
```

8. Input Tapes: None

9. Output Tapes: NPREP

10. Scratch Tapes: None

11. Storage Required:

```
MATRIX(7,3)
SYMBOL(7)
```

Total Storage is $7A4_{16}$ Bytes.

12. Subroutine User: INST

13. Subroutine Required:

```
PUTL3
PUTL4
INSTFP
```

14. Remarks: This is an arithmetic type instruction analyzer.

1. Subroutine Name: MATSUP

2. Purpose: Insert suppressed input matrix names into the Format System

3. Equations and Procedures: Scratch unit NPREP is backspaced to the beginning of the instruction section. If scratch unit NDATA already contains matrices then it is positioned at the data set trailer; otherwise it is rewound and a data set header written upon it. Each instruction record is then read to determine if the op-code is capable of containing input suppressed matrices as indicated in the array LEGAL. If the operation is capable of containing suppressed input matrices then the input matrix names are checked to see if they contain a slash in the first position. If this is the case the suppression name is entered as a null matrix on NDATA. NDATA is then returned to the first suppressed matrix name and re-read so that each added matrix on NDATA is recorded on NPREP after the instructions. Control is then returned to the calling program.

4. Input Arguments:

NUMD	: Number of matrices on NDATA
NUMSUP	: Number of suppressed input matrices to be added to NDATA
NDATA	: Logical unit containing card input matrices
NPREP	: Logical unit containing preprocessor data
NUMI	: Number of instructions on NPREP
IWORK	: Work storage area

5. Output Arguments: None

6. Error Returns: None

7. Calling Sequence:

(NUMD, NUMSUP, NDATA, NPREP, NUMI, IWORK)

8. Input Tapes:

NDATA	: contains card input matrices, if present
NPREP	: contains input abstraction instructions in coded form

9. Output Tapes:

NDATA	: will contain suppressed input matrices
NPREP	: will contain suppressed input matrix names

10. Scratch Tapes: None

11. Storage Required: Total storage is 740_{16} Bytes.

12. Subroutine User. PREP

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: TSUM

2. Purpose: To generate a summary of the matrices on a format tape if EUTL3 cannot find a matrix on the specified tape.

3. Equations and Procedures: The data set header and modifier are printed out. Then each matrix header is printed out giving the matrix name, the sign of the matrix and the row and column dimension of the matrix. A record count is also provided so the number of columns in a matrix can be calculated.

4. Input Arguments:

   NSET = The logical unit number of the format tape to be summarized

5. Output Arguments: None

6. Error Returns: None

7. Calling Sequence: TSUM(NSET)

8. Input Tapes: NSET

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required Total Storage required is 560_{16} Bytes.

12. Subroutine User: EUTL3

13. Subroutine Required: None

14. Remarks: None

1. Subroutine Name: IDNTR

2. Purpose: To form an identity matrix of the same order as the row dimension of the input matrix.

3. Equations and Procedures: The input matrix is located by EUTL3 and an identity matrix is formed. The order of the identity matrix is the same as the row dimension of the input matrix.

4. Input Arguments:

NUMOT	-	the number of output matrices
OUTPUT	-	array containing the names of the output matrices
IOSPEC	-	array containing output data set numbers
NUMIN	-	the number of input matrices
INPUT	-	array containing the names of the input matrices
INSPEC	-	array containing input data set numbers
NUMSR	-	the number of scratch data sets
ISSPEC	-	array containing scratch data set numbers
NUMSC	-	the number of input scalars
SCALAR	-	array containing the input scalars
IERROR	-	error return code
NWORKR	-	the number of words of available work storage
WORKR	-	working storage array

5. Output Arguments: None

6. Error Returns: IERROR = 11, if the input matrix cannot be found.

7. Calling Sequence:

```
IDNTR(NUMOT,OUTPUT,IOSPEC,NUMIN,INPUT,INSPEC,NUMSR,ISSPEC,
     NUMSC,SCALAR,IERROR,NWORKR,WORKR)
```

8. Input Tapes: INSPEC

9. Output Tapes: IOSPEC

10. Scratch Tapes: None

11. Storage Required Total Storage required is $50E_{16}$ Bytes.

12. Subroutine User: EXEQ

13. Subroutine Required: EUTL3, EUTL5, EUTL6

14. Remarks: A = B.IDENTR.

1. Subroutine Name: IDNTC

2. Purpose: To form an identity matrix of the same order as the column dimension of the input matrix.

3. Equations and Procedures: The input matrix is located by EUTL3 and an identity matrix is generated. The order of the identity matrix is the same as the column dimension of the input matrix.

4. Input Arguments:

NUMOT	-	the number of output matrices
OUTPUT	-	array containing the names of the output matrices
IOSPEC	-	array containing output data set numbers
NUMIN	-	the number of input matrices
INPUT	-	array containing the names of the input matrices
INSPEC	-	array containing input data set numbers
NUMSR	-	the number of scratch data sets
ISSPEC	-	array containing scratch data set numbers
NUMSC	-	the number of input scalars
SCALAR	-	array containing the input scalars
IERROR	-	error return code
NWORKR	-	the number of words of available work storage
WORKR	-	working storage array

5. Output Arguments: None

6. Error Return: IERROR = 11, if the input matrix cannot be found.

7. Calling Sequence:

```
IDNTC(NUMOT,OUTPUT,IOSPEC,NUMIN,INPUT,INSPEC,NUMSR,ISSPEC,
     NUMSC,SCALAR,IERROR,NWORKR,WORKR)
```

8. Input Tapes: INSPEC

9. Output Tapes: IOSPEC

10. Scratch Tapes: None

11. Storage Required: Total Storage required is $50E_{16}$ Bytes.

12. Subroutine User: EXEQ

13. Subroutine Required: EUTL3, EUTL5, EUTL6

15. Remarks: A = B.IDENTC.

1. Subroutine Name: EIG1

2. Purpose: To create dynamic storage for eigenvalue and eigenvector calculations and locate input matrix.

3. Equations and Procedures:

   1) Dynamic storage is allocated.
   2) REGE2 is called to transfer matrix to scratch tape.
   3) EIGB is called to iteration on a matrix.
   4) Storage required is 5 vectors of equal length (order of matrix).
   5) If the NWORK storage is too small for this, an error message is printed out.
   6) If the eigenmatrix cannot be located, another error message is written.

4. Input Arguments:

NMOUT	-	the number of output matrices
NAMOT	-	array containing the names of the output matrices
IODS	-	array containing output data set numbers
NMIN	-	the number of input matrices
INPT	-	array containing the names of the input matrices
INSP	-	array containing input data set numbers
NSCR	-	the number of scratch data sets
ISSP	-	array containing scratch data set numbers
NMSCL	-	the number of input scalars
NAMSC	-	array containing the input scalars
ERR -	-	error return code
NWKR	-	the number of words of available work storage
WKR	-	working storage array

5. Output Arguments: ERR

6. Error Returns:

   ERR = true if input matrix can't be found
   = true if not enough storage to calculate eigenvalue and vector.

7. Calling Sequence:

```
Call EIG1(NMOUT,NAMOT,IODS,NMIN,INPT,INSP,NSCR,ISSP,
          NMSCL,NAMSC,ERR,NWKR,WKR)
```

8. Input Tapes: INSP

9. Output Tapes: IODS, NPOT

10. Scratch Tapes: ISSP (4 scratch tapes needed)

11. Storage Required:
    Total Storage Required is $A3C_{16}$ Bytes.

12. Subroutine User: EXEQ

13. Subroutines Required:

    REGE2
    EIGB
    EUTL3

14. Remarks:

1. Subroutine Name: REGE2

2. Purpose: This routine takes compressed (Format) Eigen-matrix and transfers it (exponded) to a scratch data set. Storage on the scratch data set is optimized by placing as many columns into a record (which has NLEFT words max) as possible.

3. Equations and Procedures:

   1) Compute number of columns/NLEFT record = NCR: maximum NCOL records.
   2) Compute number of columns in last record = NRR
   3) Compute total number of records = NR
      a) Number of full records NFR
      b) Number of columns in last record NRR
   4) Read compressed matrix from I13 expand column using EUTL9. Provide for suppressed column.
   5) Take care of full records first.
   6) Next write final clean-up record containing remaining matrix columns.

4. Input Arguments:

I12	-	data set to which eigen-matrix is transferred
I13	-	data set with compressed (Format) eigen-matrix
ARRAY	-	work storage
DARRAY	-	work storage
NCOL	-	order of matrix
NR	-	total number of records on scratch data set
NLEFT	-	maximum record length

5. Output Arguments: None

6. Error Returns: None

7. Calling Sequence:

   Call REGE2(I12,I13,ARRAY,DARRAY,NCOL,NR,NLEFT)

8. Input Tapes: I13 contains original format compressed eigenmatrix.

9. Output Tapes: I12 contains expanded eigenmatrix each record is up to "NLEFT" words and contains an integer number of matrix columns/record.

10. Scratch Tapes: None

11. Storage Required:

    Total Storage required is $6D4_{16}$ Bytes.

12. Subroutine User:

    EIG1

13. Subroutine Required:

    EUTL9

14. Remarks:

1. Subroutine Name: EIGB

2. Purpose: Control iteration routine EIG. . Writes eigenvalue, eigenvector matrices on tape.

3. Equations and Procedures:

   1) Write out controls used in iteration:
      NE - number of eigenvalues requested — Defaults are
      IFLAG - row or column iteration — NOIT = 500
      NOIT - number of iterations per criteria update — CRIT = .001
      CRIT - convergence criteria
   2) Locate and expand input vecotrs using EUTL3
   3) Call routine EIG
   4) Print out frequency in CPS and radians/sec and the normalized eigenvector
   5) If output vecotrs are requested write them on an output tape when vectors are written.

4. Input Arguments: See calling sequence.

5. Output Arguments: None

6. Error Returns:

7. Calling Sequence:

```
Call EIGB(NE,IBEG,IEND,WKR(N1),WKR(N5),WKR(N3),WKR(N4),
          WKR(N2),NMDB,NEIGL,NEIGV,NAMOT,NMOUT,WKR(N3),
          WKR(N4),NSAVE,INVEC,INPT,NMIN,ERR,IFLAG,NOIT,
          NRIT,NVECT,NR,NLEFT)
```

8. Input Tapes:

9. Output Tapes: NSAVE, NVECT, NPOT, NEIGL, O

10. Scratch Tapes: NSKRAT

11. Storage Required:
    Total Storage required is $18B2_{16}$ Bytes.

12. Subroutine User: EIG1

13. Subroutines Required: EUTL3, EIG, EUTL5, EUTL6

14. Remarks:

1. Subroutine Name: EIG

2. Purpose: This routine computes only one eigenvalue and vector for each call from EIGB.

3. Equations and Procedures:

1) Power method iteration with hotteling deflation to remove dominant root.
2) Iterate on column vector, get vector and value.
3) If another value is desired, iterate on row vector and value.
4) Use row and column vectors to deflate matrix.
5) Use deflate matrix when iterating for next column vector
6) If the convergence must be updated (CRITZ = CRITZ+CRIT)
7) Return to routine EIGB.

4. Input Arguments:

N	-	order of characteristic matrix
IPRINT	-	= 0 no iteration print; = 1 print iterations
NEIG	-	= always = 1
CRIT	-	convergence criteria
NOIT	-	number of iterations
IBEG	-	location (unit) of col (characteristic) vector matrix
IBEND	-	unit on which deflated matrix is placed

5. Output Arguments:

ROOTS	-	returned eigenvalue
XIN	-	returned eigenvector
NERR	-	error indicator = 0 no error; =1 col do not converge
ICOUNT	-	= 1 if value converged =2 row does not converge
IFLAG	-	= both input and output =3 row root ≠ col root
		=4 machine or input error

= 0 go directly to col iteration
= 1 continue row iteration indicates row iteration failed previously and criteria has been increased

6. Error Returns:

NERR = 1 no error; = 2 eigencols do not converge; =3 eigenrows do not converge; = 4 row root not equal to col. root; = 5 no nonzero element in (col); = 6 no non-zero element in row; = 7 scalar product of row and column vectors = zero.

7. Calling Sequence:

```
Call EIG(N,IPRINT,NEIG,ROOTS,XIN,NERR,CRIT,NOIT,ICOUNT,
         IBEG,IEND,A,XI,SIMIN,XINP,NMDB,XIP,XIMINP,NE,
         IFLAG,NUMR,NLEFT,NOFF,NTR
```

8. Input Tapes:

9. Output Tapes:

10. Scratch Tapes:

IBEG - initial (A) matrix location
IEND - location of swept (A) matrix after 1st eigenvalue is found. This unit then becomes the input for calculating the next eigenvalue and IBEG will receive the resulting swept matrix.

11. Storage Required:
Total Storage required is $1B06_{16}$ Bytes.

12. Subroutine User: EIGB

13. Subroutines Required: None

14. Remarks:

1. Subroutine Name: COLREP

2. Purpose: To generate a matrix by repeating the first input column matrix K number of times where K is the column dimension of the second input matrix.

3. Equations and Procedures: The second input matrix is located and its column dimension, NCOL, is noted. The first input matrix is located and stored in core and its row dimension, IROW, is noted. A matrix header for the output matrix of order IROW by NCOL is written. The input column is repeated NCOL times and the matrix trailer for the output matrix is written.

4. Input Arguments:

NUMOT	-	the number of output matrices
NAMIO	-	array containing the names of the output matrices
IOSPEC	-	array containing output data set numbers
NUMIN	-	the number of input matrices
NAMIN	-	array containing the names of the input matrices
INSPEC	-	array containing input data set numbers
NUMSR	-	the number of scratch data sets
ISSPEC	-	array containing scratch data set numbers
NUMSC	-	the number of input scalars
SCALAR	-	array containing the input scalars
IERROR	-	error return code
NWORK	-	the number of words of available work storage
WORK	-	working storage array

5. Output Arguments: IERROR - error flag.

6. Error Returns:

IERROR = 11, if first input matrix can't be found
= 12, if second input matrix can't be found
= 21, if output matrix can't be generated

7. Calling Sequence:

```
COLREP(NUMOT,NAMIO,IOSPEC,NUMIN,NAMIN,INSPEC,NUMSR,ISSPEC,
      NUMSC,SCALAR,IERROR,NWORK,WORK)
```

8. Input Tapes: INSPEC

9. Output Tapes: IOSPEC

10. Scratch Tapes: None

11. Storage Required: Total Storage required is 644_{16} Bytes.

12. Subroutine User: EXEQ

13. Subroutine Required: EUTL3, EUTL5, EUTL6

14. Remarks: A = B.COLREP.C

1. Subroutine Name: NULL

2. Purpose: To generate a null matrix of order n x m.

3. Equations and Procedures: The first input matrix is located and the row dimension of this matrix is saved in KROW. The second input matrix is located and the column dimension of this matrix is saved in KCOL. Then a matrix header and trailer is written. The dimension of the output matrix is KROW x KCOL.

4. Input Arguments:

NUMOT	-	the number of output matrices
NAMIO	-	array containing the names of the output matrices
IOSPEC	-	array containing output data set numbers
NUMIN	-	the number of input matrices
NAMIN	-	array containing the names of the input matrices
INSPEC	-	array containing input data set numbers
NUMSR	-	the number of scratch data sets
ISSPEC	-	array containing scratch data set numbers
NUMSC	-	the number of input scalars
SCALAR	-	array containing the input scalars
IERROR	-	error return code
NWORK	-	the number of words of available work storage
WORK	-	working storage array

5. Output Arguments: IERROR - error flag

6. Error Returns:

IERROR = 11, if first input matrix can't be found
= 12, if second input matrix can't be found

7. Calling Sequence:

```
NULL(NUMOT,NAMIO,IOSPEC,NUMIN,NAMIN,INSPEC,NUMSR,ISSPEC,
    NUMSC,SCALAR,IERROR,NWORK,WORK)
```

8. Input Tapes: INSPEC

9. Output Tapes: IOSPEC

10. Scratch Tapes: None

11. Storage Required: Total Storage required is $4BA_{16}$ Bytes.

12. Subroutine User: EXEQ

13. Subroutine Required: EUTL3, EUTL5, EUTL6

14. Remarks: A = B.NULL.C

1. Subroutine Name: DEJOIN

2. Purpose: This routine is the controlling routine to provide matrix column or row partitioning.

3. Equations and Procedures: First, the input and output data sets are defined. Next a check is made to determine if the input data set is the same as either output data set. If either or both of the output data sets are the same, the output data set is redefined as a unique scratch data set. Now a test is made to determine if the partition number was input or if it must be found. If it was not input then EUTL7 extracts the partitioning scalar. Now a test of whether a column or a row DEJOIN is desired is performed. If it is a column DEJOIN, subroutine DEJNC is called. If it is a row DEJOIN, subroutine DEJNR is called. If either or both output data sets are different from the originally allocated output data sets, a copy of the output data set is made onto the originally allocated data set by a call to EUTL4.

4. Input Arguments:

NUMOT	-	the number of output matrices
NAMIO	-	array containing the names of the output matrices
IOSPEC	-	array containing output data set numbers
NUMIN	-	the number of input matrices
NAMIN	-	array containing the names of the input matrices
INSPEC	-	array containing input data set numbers
NUMSR	-	the number of scratch data sets
ISSPEC	-	array containing scratch data set numbers
NUMSC	-	the number of input scalars
ISCALE	-	array containing the input scalars
IERROR	-	error return code
NWORK	-	the number of words of available work storage
WORK	-	working storage array

5. Output Arguments: IERROR - error flag

6. Error Returns: An error condition occurs when a matrix cannot be located, the subscripts used to extract the partition number exceed the dimension limit, or when the partition number is invalid.

7. Calling Sequence:

```
DEJOIN(NUMOT,NAMIO,IOSPEC,NUMIN,NAMIN,INSPEC,NUMSR,ISSPEC,
      NUMSC,ISCALE,IERROR,NWORK,WORK)
```

8. Input Tapes: One or two input data sets in the INSPEC array.

9. Output Tapes: Two output data sets in the IOSPEC array.

10. Scratch Tapes: Two scratch data sets in the ISSPEC array.

11. Storage Required: Total Storage required is 918_{16} Bytes.

12. Subroutine User: EXEQ

13. Subroutine Required:

EUTL1
EUTL3
EUTL7
DEJNC
DEJI
EUTL4

14. Remarks: A,B = C.DEJOIN.(d,e)

1. Subroutine Name: DEJNR

2. Purpose: This routine row partitions a matrix at a specified row.

3. Equations and Procedures: First the partition number is tested against the row dimension of the matrix to be partitioned if it is greater than the number of rows an error occurs. If it is less than or equal to the row dimension then the input matrix A is partioned to form two output matrices C1 on C2.

   $A(MXN) = C1(J-1 \times n), C2(m-J+1 \times n)$ where $1 < J \leq m$

4. Input Arguments:

NAME	-	the names of the output matrices
NSET	-	the data set number of the input matrix to be partitioned
NSET1	-	the data set number of the first output matrix
NSET2	-	the data set number of the second output matrix
JPART	-	the row number at which the input matrix is to be partitioned
IROW	-	the row dimension of the input matrix
ICOL	-	the column dimension of the input matrix
NWORK	-	the number of words of available working storage
WORK	-	working storage array
ERROR	-	error flag.

5. Output Arguments: ERROR

6. Error Returns: An error condition occurs when JPART is greater than the row dimension of the input matrix.

7. Calling Sequence:

```
DEJNR(NAME,NSET,NSET1,NSET2,JPART,IROW,ICOL,NWORK,WORK,
     ERROR)
```

8. Input Tapes: NSET

9. Output Tapes: NSET1, NSET2

10. Scratch Tapes: None

11. Storage Required: Total Storage required is $5F6_{16}$ Bytes.

12. Subroutine User: DEJOIN

13. Subroutine Required: EUTL5, EUTL9, EUTL8, EUTL6

14. Remarks: None

1. Subroutine Name: DEJNC

2. Purpose: This routine column partitions a matrix at a specified column.

3. Equations and Procedures: First the partition number is tested against the column dimension of the matrix to be partitioned. If it is greater than the number of columns an error occurs. If it is less than or equal to the column dimension the input matrix A is partitioned to form two output matrices C1 and C2.

   A(MXN) = C1(M x J-1), C2(m x n-J+1) where $1 < J \leq n$

4. Input Arguments:

NAME	-	the names of the output matrices
NSET	-	the data set number of the input matrix to be partitioned
NSET1	-	the data set number of the first output matrix
NSET2	-	the data set number of the second output matrix
JPART	-	the column number at which the input matrix is to be partitioned
IROW	-	the row dimension of the input matrix
ICOL	-	the column dimension of the input matrix
NWORK	-	the number of words of available working storage
WORK	-	working storage array
ERROR	-	error flag

5. Output Arguments: ERROR

6. Error Returns: An error condition occurs when JPART is greater than the column dimension of the input matrix.

7. Calling Sequence:

```
DEJNC(NAME,NSET,NSET1,NSET2,JPART,IROW,ICOL,NWORK,WORK,
     ERROR)
```

8. Input Tapes: NSET

9. Output Tapes: NSET1, NSET2

10. Scratch Tapes: None

11. Storage Required: Total Storage required is 638_{16} Bytes.

12. Subroutine User: DEJOIN

13. Subroutine Required: EUTL5, EUTL6

14. Remarks: None

1. Subroutine Name: ASSEM

2. Purpose: To assemble the element matrices generated by the USER04 module.

3. Equations and Procedures: The matrix containing the system constants is found to generate the value NSYS. The assembled matrices will be of order NSYS, that is, they will not be reduced. Next, the variable ITYPE is tested to see what type of matrices are to be assembled. Depending on the value of ITYPE control is transferred to either ASSEMC or ASSEMS to assemble and write the matrices

   ITYPE = 1, for element stiffness assembly
   = 2, for element mass assembly
   = 3, for element incremental assembly
   = 4, for element applied load assembly.

4. Input Arguments:

NUMOT	-	the number of output matrices
NAMIO	-	array containing the names of the output matrices
IOSPEC	-	array containing output data set numbers
NUMIN	-	the number of input matrices
NAMIN	-	array containing the names of the input matrices
INSPEC	-	array containing input data set numbers
NUMSR	-	the number of scratch data sets
ISSPEC	-	array containing scratch data set numbers
NUMSC	-	the number of input scalars
ISCALE	-	array containing the input scalars
IERROR	-	error return code
NWORK	-	the number of words of available work storage
WORK	-	working storage array

5. Output Arguments: None

6. Error Returns:

   IERROR = 21, if the matrix containing the system constants can't be found
   = 15, if there is not enough work storage for the assembled matrix

7. Calling Sequence:

```
ASSEM(NUMOT,NAMIO,IOSPEC,NUMIN,NAMIN,INSPEC,NUMSR,ISSPEC,
     NUMSC,ISCALE,IERROR,NWORK,WORK)
```

8. Input Tapes: The data set numbers are contained in the INSPEC array.

9. Output Tapes: The data set numbers are contained in the IOSPEC array.

10. Scratch Tapes: The data set numbers are contained in the ISSPEC array. This module uses at most two scratch tapes.

11. Storage Required: Total Storage required is $72C_{16}$ Bytes.

12. Subroutine User: EXEQ

13. Subroutine Required:

    EUTL3
    ASSEMC
    ASSEMS

14. Remarks: A = B.ASSEM.C,(d)

1. Subroutine Name: ASSEMC

2. Purpose: To assemble the element applied load columns.

3. Equations and Procedures: The tape containing the element matrices is read and the LISTEL and FTEL arrays are stored for each element. Using the LISTEL array the FTEL arrays is assembled into a master applied load array. This process is repeated for each element.

4. Input Arguments:

NSET1	-	data set on which the input element matrices are stored
NSET2	-	data set number of output matrix
NAME1	-	array containing name of matrix on NSET1
NAME2	-	array containing name of matrix on NSETL
NSYS	-	order of assembled matrix
LISTEL	-	storage for the LISTEL array
FTEL	-	storage for the element applied loads array
FCOL	-	storage for the assembled FTEL
NWORK	-	number of words of work storage
WORK	-	work storage
IERROR	-	error return

5. Output Arguments: None

6. Error Returns:

IERROR = 11, if the input matrix can't be found
= 15, if a value of LISTEL is greater than NSYS

7. Calling Sequence:

```
ASSEMC(NSET1,NAME1,NSET2,NAME2,NSYS,LISTEL,FTEL,FCOL,
      NWORK,WORK,IERROR)
```

8. Input Tapes: NSET1

9. Output Tapes: NSET2

10. Scratch Tapes: None

11. Storage Required: Total Storage required is 678_{16} Bytes.

12. Subroutine User: ASSEM

13. Subroutine Required: EUTL3, EUTL5, EUTL6

14. Remarks: None

1. Subroutine Name: ASSEMS

2. Purpose: To assemble the element stiffness, element mass or element incremental matrices as generated by the USER04 module.

3. Equations and Procedures: The matrix containing the input element matrices is found and depending on what type of matrices are to be assembled a different read statement is initiated. The LISTEL array and element matrix is then stored in core. Then using LIST processing techniques the element matrix is assembled in core. Only non-zero values are considered. If all non-zero values can't fit in core then the values in core are written on tape until more elements are assembled in core. These non-zero values are then merged with the ones on tape to produce the output assembled matrix.

4. Input Arguments:

NSET1	-	data set number of tape containing element matrices
NAMIN	-	array containing name of matrix on NSET1
NSET2	-	data set num-er of output matrix
NAMOUT	-	array containing name of output matrix
NS1	-	scratch tape 1
NS2	-	scratch tape 2
NSYS	-	order of assembled matrix
NCORE	-	number of available words of core storage
ITYPE	-	indicates type of matrices to be assembled
ICOLPT	-	storage needed for assembly
VALUE	-	storage needed for assembly
IERROR	-	error flag

5. Output Arguments: None

6. Error Returns:

IERROR = 11, if the input matrix on NSET1 cannot be found

7. Calling Sequence:

```
ASSEMS(NSET1,NAMIN,NSET2,NAMOUT,NS1,NS2,NSYS,NCORE,ITYPE,
      ICOLPT,VALUE,IERROR)
```

8. Input Tapes: NSET1

9. Output Tapes: NSET2

10. Scratch Tapes: NS1, NS2

11. Storage Required. Total Storage required is 2924_{16} Bytes.

12. Subroutine User: ASSEM

13. Subroutine Required:

EUTL3
EUTL5
EUTL6

14. Remarks: For a more detailed documentation see the source listing os subroutine ASSEMS.

1. Subroutine Name: STRESS

2. Purpose: This is the control routine for computing the net element stress matrix. It also controls the optional engineering print of apparent element stresses, element applied stresses and net element stresses.

3. Equations and Procedures: This module first tests the allocation of the input and output matrices. If both input matrices are on the same data set, but not on the data set to contain the output matrix, then one of these input matrices is copied onto a scratch data set. If both input matrices are on the same data set as the output matrix, then each input matrix is located and copied onto a scratch data set. When this has been completed both input matrices are positioned and the matrix header for the output matrix is written.

   Pointers are next set up indicating positions in the work area for arrays needed to compute the stresses.

   Subroutine STRES1 is called to read element data and displacements contained in the input matrices.

4. Input Arguments:

NUMOT	-	the number of output matrices
NAMIO	-	array containing the names of the output matrices
IOSPEC	-	array containing output data set numbers
NUMIN	-	the number of input matrices
NAMIN	-	array containing the names of the input matrices
INSPEC	-	array containing input data set numbers
NUMSR	-	the number of scratch data sets
ISSPEC	-	array containing scratch data set numbers
NUMSC	-	the number of input scalars
SCALAR	-	array containing the input scalars
IERROR	-	error return code
NWORK	-	the number of words of available work storage
WORK	-	working storage array

5. Output Arguments: IERROR

6. Error Returns:

   IERROR = 11 or 21, if either the first or second input matrix can't be found by EUTL3

7. Calling Sequence:

```
STRESS(NUMOT,NAMIO,IOSPEC,NUMIN,NAMIN,INSPEC,NUMSR,ISSPEC,NUMSC,
      SCALAR,IERROR,NWORK,WORK)
```

8. Input Tapes: INSPEC

9. Output Tapes: IOSPEC

10. Scratch Tapes: ISSPEC

11. Storage Required Total Storage required is 704_{16} Bytes.

12. Subroutine User: EXEQ

13. Subroutine Required:

EUTL1
EUTL3
EUTL4
EUTL5
STRES1

14. Remarks: C = A,B.STRESS.(d,e)

1. Subroutine Name: STRES1

2. Purpose: This routine reads element data and displacements, calls STRES2 to calculate the stresses, then writes the net element stresses for each element.

3. Equations and Procedures: A test is first made to see if enough work space is available to process all elements successfully. Then for each element this module:

   (a) Reads a column of the input matrix containing element data on NSET1.
   (b) Compresses this column, keeping only the element data necessary to calculate the stress.
   (c) Calls STRES2 to calculate the stresses and print them out.
   (d) Writes the calculated net element stresses on the output data set. One column is written for each element, such that each column contains net stresses for each load condition.

4. Input Arguments:

NELEM	-	the number of elements
NLOAD	-	the number of load conditions
NMDB	-	the order of the displacement array
MAXEL	-	the length of work storage needed to process the maximum size element
NL48	-	NLOAD*48
NSET1	-	the data set number of the input matrix containing element data
NSET2	-	the data set number of the input matrix containing the displacements
NSET3	-	the data set number of the output matrix
NAME	-	the name of the matrix on NSET2
SCALAR	-	an array containing the input scalars
MAT	-	a work array local to STRES1
IPM	-	a work array local to STRES1
STRESN	-	work storage for the net element stresses
NWORK	-	the number of words of available working storage
WORK	-	working storage array
IERROR	-	error return

5. Output Arguments: IERROR - error return

6. Error Returns:

   IERROR = 15, if not enough work storage to process all elements.

7. Calling Sequence:

```
STRES1(NELEM,NLOAD,NMDB,MAXEL,NL48,NSET1,NSET2,NSET3,
       NAME,SCALAR,MAT,IPM,STRESN,NWORK,WORK,IERROR)
```

8. Input Tapes: NSET1, NSET2

9. Output Tapes: NSET3

10. Scratch Tapes: None

11. Storage Required: Total Storage required is $75A_{16}$ Bytes.

12. Subroutine User: STRESS

13. Subroutine Required:

ELREAD
FREEUP
STRES2

14. Remarks: None

1. Subroutine Name: STRES2

2. Purpose: This routine calculates the net element stresses for each load condition. Then calls STRPRT to print the apparent, applied end net element stresses.

3. Equations and Procedures: A test is first made to see if the displacements for all load conditions can fit in core. If they can, then they are read into core. If the displacements for all load conditions do not fit into core then the displacements for each load condition are read into core one at a time. For each load condition the net element stresses are calculated and depending on the option specified the apparent, applied or net stresses are printed for each element.

4. Input Arguments:

IEL	-	the element number
IPL	-	the element type (new plug number)
NMDB	-	the order of the displacement array
NLOAD	-	the number of load conditions
NRSEL	-	the order of the element stress array
NORD	-	the order of the LISTEL array
NNO	-	the order of the NODES array
NSET2	-	the data set number of the displacement matrix
INCORE	-	a logical variable indicating in all displacements are INCORE
FIRST	-	a logical variable
NAME	-	the name of the matrix on NSET2
SCALAR	-	an array containing the input scalars
LISTEL	-	a decoding array to go from reduced degrees of freedom to system degrees of freedom
SEL	-	the element stress matrix
SZALEL	-	applied element stress matrix
NODES	-	an array containing the element node points
STRESN	-	net element stress matrix
NWORK	-	the number of words of available working storage
DISPL	-	the displacement array
IERROR	-	error return

5. Output Arguments: STRESN, IERROR

6. Error Returns:

IERROR = 21, if EUTL3 can't find the displacement matrix

7. Calling Sequence:

```
STRES2(IEL,IPL,NMDB,NLOAD,NRSEL,NORD,NNO,NSET2,INCORE,FIRST,
      NAME,SCALAR,NSC,LISTEL,NODES,SEL,SZALEL,STRESN,NWORK,
      DISPL,IERROR)
```

8. Input Tapes: NSET2

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is $117A_{16}$ Bytes.

12. Subroutine User: STRES1

13. Subroutine Required:

```
COLMRD
EUTL3
STRPRT
```

14. Remarks: None

1. Subroutine Name: FORCE

2. Purpose: This is the control routine for computing the net element force matrix. It also controls the optional engineering print of apparent element forces, element applied forces and net element forces.

3. Equations and Procedures: This module first test the allocation of the input and output matrices. If both input matrices are on the same data set, but not on the data set to contain the output matrix, then one of these input matrices is copied onto a scratch data set. If both input matrices are on the same data set as the output matrix, then each input matrix is located and copied onto a scratch data set. When this has been completed both input matrices are positioned and the matrix header for the output matrix is written.

   Pointers are next set up to indicating positions in the work area for arrays needed to compute the forces.

   Subroutine FORCE1 is called to read element data and displacements contained in the input matrices

4. Input Arguments:

NUMOT	-	the number of output matrices
NAMOT	-	array containing the names of the output matrices
IOSPEC	-	array containing output data set numbers
NUMIN	-	the number of input matrices
NAMIN	-	array containing the names of the input matrices
INSPEC	-	array containing input data set numbers
NUMSR	-	the number of scratch data sets
ISSPEC	-	array containing scratch data set numbers
NUMSC	-	the number of input scalars
SCALAR	-	array containing the input scalars
IERROR	-	error return code
NWORK	-	the number of words of available work storage
WORK	-	working storage array

   Output Arguments: IERROR

   Error Returns:

   IERROR = 11 or 21, if either the first or second input matrix can't be found by EUTL3

   Calling Sequence:

```
FORCE(NUMOT,NAMOT,IOSPEC,NUMIN,NAMIN,INSPEC,NUMSR,ISSPEC,
     NUMSC,SCALAR,IERROR,NWORK,WORK)
```

8. Input Tapes: INSPEC

9. Output Tapes: IOSPEC

10. Scratch Tapes: ISSPEC

11. Storage Required: Total Storage required is 704_{16} Bytes.

12. Subroutine User: EXEQ

13. Subroutine Required:

EUTL1
EUTL3
EUTL4
EUTL5
FORCE1

14. Remarks: C = A,B.FORCE.(d,e)

1. Subroutine Name: FORCE1

2. Purpose: This routine reads element data and displacements, calls FORCE2 to calculate the stresses, then writes the net element forces for each element.

3. Equations and Procedures: A test is first made to see if enough work space is available to process all elements successfully. Then for each element this module:

   (a) Reads a column of the input matrix containing element data on NSET1.
   (b) Compresses this column, keeping only the element data necessary to calculate the forces.
   (c) Calls FORCE2 to calculate the forces and print them out.
   (d) Writes the calculated net element forces on the output data set. One column is written for each element, such that each column contains net stresses for each load condition.

4. Input Arguments:

NELEM	-	the number of elements
NLOAD	-	the number of load conditions
NMDB	-	the order of the displacement array
MAXEL	-	the length of work storage needed to process the maximum size element
NL48	-	NLOAD*48
NSET1	-	the data set number of the input matrix containing element data
NSET2	-	the data set number of the input matrix containing the displacements
NSET3	-	the data set number of the output matrix
NAME	-	the name of the matrix on NSET2
SCALAR	-	an array containing the input scalars
MAT	-	a work array local to FORCE1
IPM	-	a work array local to FORCE1
FORCEN	-	work storage for the net element forces
NWORK	-	the number of words of available working storage
WORK	-	working storage array
IERROR	-	error return

5. Output Arguments: IERROR - error return

6. Error Returns:

   IERROR = 15, if not enough work storage to process all elements

7. Calling Sequence:

```
FORCE1(NELEM,NLOAD,NMDB,MAXEL,NL48,NSET1,NSET2,NSET3,NAME,
      SCALAR,MAT,IPM,FORCEN,NWORK,WORK,IERROR)
```

8. Input Tapes: NSET1, NSET2

9. Output Tapes: NSET3

10. Scratch Tapes: None

11. Storage Required: Total Storage required is 756_{16} Bytes.

12. Subroutine User: FORCE

13. Subroutine Required:

    ELREAD
    FREEUP
    FORCE2

14. Remarks: None

1. Subroutine Name: FORCE2

2. Purpose: This routine calculates the net element forces for each load condition. Then calls STRPRT to print the apparent, applied and net element forces.

3. Equations and Procedures: A test is first made to see if the displacements for all load conditions can fit in core. If they can, then they are read into core. If the displacements for all load conditions do not fit into core then the displacements for each load condition are read into core one at a time. For each load condition the net element forces are calculated and depending on the option specified the apparent, applied or net forces are printed for each element.

4. Input Arguments:

IEL	-	the element number
IPL	-	the element type (new plug number)
NMDB	-	the order of the displacement array
NLOAD	-	the number of load conditions
NOINK	-	the order of the element stiffness array
NORD	-	the order of the LISTEL array
NNO	-	the order of the nodes array
NSET2	-	the data set number of the displacement matrix
INCORE	-	a logical variable indicating if all displacements are in core
FIRST	-	a logical variable
NAME	-	the name of the matrix on NSET2
SCALAR	-	an array containing the input scalars
NSC	-	an array containing the number of stress components for each element type
LISTEL	-	a decoding array to go from reduced degrees of freedom to system degrees of freedom
AKEL	-	the element stiffness array
FTEL	-	an array containing element applied force
NODES	-	an array containing the element node point
FORCEN	-	net element force matrix
NWORK	-	the number of words of available working storage
DISPL	-	the displacement array
IERROR	-	error return

5. Output Arguments: FORCEN, IERROR

6. Error Returns:

IERROR = 21, if EUTL3 can't find the displacement matrix

7. Calling Sequence:

```
FORCE2(IEL,IPL,NMDB,NLOAD,NOINK,NORD,NNO,NSET2,INCORE,
       FIRST,NAME,SCALAR,NSC,LISTEL,AKEL,FTEL,NODES,
       FORCEN,NWORK,DISPL,IERROR)
```

8. Input Tapes: NSET2

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is $10CA_{16}$ Bytes.

12. Subroutine User: FORCE1

13. Subroutine Required:

```
COLMRD
EUTL3
STRPRT
```

14. Remarks: None

1. Subroutine Name: EPRINT

2. Purpose: To print out the net element forces or net element stresses calculated by the FORCE or STRESS modules.

3. Equations and Procedures: This module first tests the allocation of the input matrices. If both input matrices are on the same data set, then the first input matrix is copied onto a scratch data set.

   The input matrices are found and tested for compatability and the first input matrix is copied if necessary.

   The matrix containing element information is read a column at a time as is the matrix containing the net element stress or forces. Then the input print control is tested in order to write out the correct heading for either the forces or stresses. Subroutine STRPRT is called for each load condition to print out the values in the second input matrix.

4. Input Arguments:

NUMOT	-	the number of output matrices
NAMIO	-	array containing the names of the output matrices
IOSPEC	-	array containing output data set numbers
NUMIN	-	the number of input matrices
NAMIN	-	array containing the names of the input matrices
INSPEC	-	array containing input data set numbers
NUMSR	-	the number of scratch data sets
ISSPEC	-	array containing scratch data set numbers
NUMSC	-	the number of input scalars
SCALAR	-	array containing the input scalars
IERROR	-	error return code
NWORK	-	the number of words of available work storage
WORK	-	working storage array

5. Output Arguments: IERROR - error return.

6. Error Returns:

   IERROR = 11 if EUTL3 can't find first input matrix
   = 12 if EUTL3 can't find second input matrix.

7. Calling Sequence:

```
EPRINT(NUMOT,NAMIO,IOSPEC,NUMIN,NAMIN,INSPEC,NUMSR,ISSPEC,
      NUMSC,SCALAR,IERROR,NWORK,WORK)
```

8. Input Tapes: INSPEC

9. Output Tapes: None

10. Scratch Tapes: This routine uses at most one scratch tape.

11. Storage Required: Total Storage required is $1D28_{16}$ Bytes.

12. Subroutine User: EXEQ

13. Subroutines Required:

EUTL3
ELREAD
FREEUP
STRPRT

14. Remarks: EPRINT(a,b,c)D

1. Subroutine Name: STRPRT

2. Purpose: To write on the system output data set the values calculated by the FORCES and STRESS modules.

3. Equations and Procedures:

   (a) Test the input variable IFMT to write out the correct heading for the element type being processed.
   (b) Calculate the number of stress or force points to be printed.
   (c) If ABS(STRESS(I)) $\leq$ EZERO then STRESS(I)=0.0.
   (d) Write out the values in array STRESS according to the input format.

4. Input Arguments:

IFMT	-	indicates element type and either stress or force print
EZERO	-	suppression value
NRSEL	-	length of STRESS array
FMT	-	format used in printer
NSC	-	number of force or stress component
STRESS	-	input array containing force or stress to be printed

5. Output Arguments: None

6. Error Returns: None

7. Calling Sequence:

   STRPRT(IFMT,EZERO,NRSEL,FMT,NSC,STRESS)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is $E20_{16}$ Bytes.

12. Subroutine User: STRES2, FORCE2, EPRINT

13. Subroutine Required: None

14. Remarks: None

1. Subroutine Name: ELREAD

2. Purpose: This routine reads one column of the matrix which contains element information and puts that column in working storage and returns element variables.

3. Equations and Procedures: Reads one column of the input matrix which contains:

IEL,IPL,
NORD,(LISTEL(I),I=1,NORD),
NOINK,(AKEL(I),I=1,NOINK),
NORD,(FTEL(I),I=1,NORD),
NNO,(NODES(I),I=1,NNO),
NSEL,(SEL(I),I=1,NSEL),
NRSEL,(SZALEL(I),I=1,NRSEL),
NOINK,(ANEL(I),I=1,NOINK),
NMASS,(AMASS(I),I=1,NMASS)

Then decodes and returns the variables

IEL,IPL,NORD,NOINK,NNO,NSEL,NRSEL and NMASS

where

LISTEL	-	contains boundary condition information
AKEL	-	is the element stiffness matrix
FTEL	-	is the applied load matrix
NODES	-	contains the grid points defining the element
SEL	-	is the element stress array
SZALEL	-	is the thermal stress array
ANEL	-	is the incremental stiffness array
AMASS	-	is the element mass matrix

4. Input Arguments:

NSET	-	data set number of input matrix
WORK	-	working storage into which element data is read
NWORK	-	number of words available in the work array
IEL	-	the element number
IPL	-	the element type (plug number)
NORD	-	the order of the LISTEL and FTEL arrays
NOINK	-	the order of the AKEL and ANEL arrays
NNO	-	the order of the nodes array
NSEL	-	the order of the SEL array
NRSEL	-	the order of the SZALEL array
NMASS	-	the order of the AMASS array

5. Output Arguments:

NLEFT	-	the number of work remaining in the work array
NEXT	-	the next useable position in the work array

6. Error Returns: None

7. Calling Sequence:

```
ELREAD(NSET,WORK,NWORK,NLEFT,NEXT,IEL,IPL,NORD,NOINK,NNO,
       NSEL,NRSEL,NMASS)
```

8. Input Tapes: NSET

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is 434_{16} Bytes.

12. Subroutine User: STRES1, FORCE1

13. Subroutine Required: None

14. Remarks: None

1. Subroutine Name: FREEUP

2. Purpose: This routine is used to compress the work array by compressing out unwanted matrices and freeing up more storage used after a call to ELREAD.

3. Equations and Procedures: This routine will only compresss an array containing submatrices which are preceeded by the length of the submatrix.

   The number of non-zero elements of MAT is tested against NMAT. If they aren't equal then an error occurs.

   The work array is then compressed by searching for those submatrices to be saved as indicated by a non-zero position in the MAT array. When a submitrix to be kept is found it is moved up in the work array and its initial position in the work array is kept track of in the IPM array.

   The space taken up by submatrices not wanted is now freed-up for use by someone else.

4. Input Arguments:

WORK	-	the input matrix to be compressed up
ISTART	-	the position of the dimension of the first submatrix in the work array
IPWORK	-	the position in the work array at which the submatrices are to be moved up to
MATOUT	-	an integer indicating the number of submatrices to be kept, should equal the number of non-zero elements in the MAT array
NMAT	-	the length of the MAT and IPM arrays
MAT	-	If MAT(I) is non-zero then the sub-matrix in the I^{th} position will be kept, if MAT(I)=0 then that submatrix will be compressed out.

5. Output Arguments:

WORK	-	the cleaned-up input array
IPM	-	contains the initial position of the saved submatrix in the cleaned-up work array
NEXT	-	the next useable position in the work array
IERROR	-	error return

6. Error Returns:

   IERROR = 15, if there is an input error

7. Calling Sequence:

   FREEUP(WORK,ISTART,IPWORK,MATOUT,NMAT,MAT,IPM,NEXT,IERROR)

8. Input Tape: None

9. Output Tape: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is $4B4_{16}$ Bytes.

12. Subroutine User: STRES1, FORCE1

13. Subroutine Required: None

14. Remarks: None

1. Subroutine Name: COLMRD

2. Purpose: This routine is a utility routine used to read a column and uncompress it if necessary. Used when storing more than one column in the work array.

3. Equations and Procedures: One column of the input data set is read and EUTL9 is called to uncompress the column if necessary.

4. Input Arguments:

   WORK - working storage array, used to input and output the column read
   NSET - the data set number of the matrix to be read
   LENGTH - the length of storage available to EUTL9

5. Output Arguments: WORK

6. Error Returns: None

7. Calling Sequence:

   COLMRD(WORK,NSET,LENGTH)

8. Input Tapes: NSET

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is $1DA_{16}$ Bytes.

12. Subroutine User: STRES2, FORCE2

13. Subroutine Required:

    EUTL9

14. Remarks: None

1. Subroutine Name: GPRINT

2. Purpose: This is the control routine for engineering printout of grid point data of reactions, displacements and eigenvectors. It can also be used for printout of user matrices.

3. Equations and Procedures: Index pointer indicating the initial position in the work array are calculated to make use of dynamics storage allocation.

   Subroutine GPRINT1 is called to process input matrices.

4. Input Arguments:

NUMOT	-	the number of output matrices
NAMIO	-	array containing the names of the output matrices
IOSPEC	-	array containing output data set numbers
NUMIN	-	the number of input matrices
NAMIN	-	array containing the names of the input matrices
INSPEC	-	array containing input data set numbers
NUMSR	-	the number of scratch data sets
ISSPEC	-	array containing scratch data set numbers
NUMSC	-	the number of input scalars
SCALAR	-	array containing the input scalars
NWORK	-	the number of words of available work storage
WORK	-	working storage array

5. Output Arguments: IERROR

6. Error Returns: None

7. Calling Sequence:

```
GPRINT(NUMOT,NAMIO,IOSPEC,NUMIN,NAMIN,INSPEC,NUMSR,ISSPEC,
      NUMSC,SCALAR,IERROR,NWORK,WORK)
```

8. Input Tapes: INSPEC

9. Output Tapes: IOSPEC

10. Scratch Tapes: ISSPEC - one scratch tape required

11. Storage Required: Total Storage required is $3C8_{16}$ Bytes.

12. Subroutine User: EXEQ

13. Subroutine Required: GPRINT1

14. Remarks: GPRINT(a,b,c,C1.C2.C3.C4.C5.C6.C7.C8.C9.C10.C11.C12,
    D,E,F)G,H

1. Subroutine Name: GPRNT1

2. Purpose: This routine processes the input matrices and calls the appropriate subroutines to print either reactions, displacements, eigenvalues and eigenvectors or the user input matrix.

3. Equations and Procedures: The input matrices are all found and processed as they are found. If an input matrix can't be found then IERROR is set to indicate which matrix could not be found. Processing is terminated.

   (a) Process first input matrix -
   This matrix contains system constants:

   NDIR - the number of directions
   NDEG - the number of types of degrees of freedom
   NREF - the number of reference points

   These are used to calculate the number of degrees of freedom in the system

   NSDOF = NDIK*NDEG*NREF

   (b) Process second input matrix -
   This is the transformation matrix for application of boundary conditions from which the LIST array can be calculated. If this matrix is suppressed then generate a dummy list array.

   (c) Process third and fourth input matrix -
   This matrix is either the reaction, displacement, eigenvector or user matrix to be printed in eigineering format. If it is the eigenvector matrix then the fourth input matrix is the eigenvector matrix. Depending on the input scalar KPRT control is transferred to the section which decodes one of the above matrices for constants. Then the matrix is stored in a scratch tape and control transfers to the subroutine which prints out the matrix.

4. Input Arguments:

NUMOT	-	the number of output matrices
NAMIO	-	the names of the output matrices
IOSPEC	-	an array containing output data set information
NUMIN	-	the number of input matrices
NAMIN	-	the names of the input matrices
INSPEC	-	an array containing input data set information
NUMSR	-	the number of scratch data sets available
ISSPEC	-	an array containing scratch data set information

4. Input Arguments, Contd.

NUMSC	-	the number of input scalars
SCALAR	-	an array containing the input scalar
MAXEIG	-	maximum number of eigenvalues that can be asked for
LIST	-	array used for boundary condition information. Decoding list to go from reduced degrees of freedom to total degree of freedom.
DISPL	-	working storage for third input matrix
EIGVAL	-	array to contain eigenvector
NWORK	-	number of words of available working storage
WORK	-	working storage array

5. Output Arguments: IERROR

6. Error Returns:

IERROR = 15, user type error
= 10*K+1, where K is the position of the input matrix not found.

7. Calling Sequence:

```
GPRNT1(NUMOT,NAMIO,IOSPEC,NUMIN,NAMIN,INSPEC,NUMSR,ISSPEC,
      NUMSC,SCALAR,IERROR,MAXEIG,LIST,DISPL,EIGVAL,NWORK,
      WORK)
```

8. Input Tapes: INSPEC

9. Output Tapes: IOSPEC

10. Scratch Tapes: ISSPEC

11. Storage Required: Total Storage required is 1000_{16} Bytes.

12. Subroutine User: GPRINT

13. Subroutine Required:

EUTL3	DISPPR
EUTL9	EIGPPR
DECODE	MATPRT
REACTP	

14. Remarks: None

1. Subroutine Name: DECODE

2. Purpose: This routine will decode a format matrix and put it out in the form of full column records with no headers or trailers.

3. Equations and Procedures: Read each column into a work array and test to see if it should be uncompressed. Also keep count of the number of columns read in case there are any missing columns. A missing column indicates that all row elements are zero so regenerate the zero column. If an error occurs then call TSUM to give a tape summary of the input data set.

4. Input Arguments:

NSET	-	the data set number of the FORMAT matrix
NSETS	-	the data set number of the tape on which the decoded matrix will go
IROW	-	row dimension of input matrix
ICOL	-	column dimension of input matrix
WORK	-	work array of order IROW

5. Output Arguments: JERROR - error flag

6. Error Returns:

JERROR = 0, no error
JERROR = 1, error

7. Calling Sequence:

```
DECODE(NSET,NSETS,IROW,ICOL,WORK,JERROR)
```

8. Input Tapes: NSET

9. Output Tapes: NSETS

10. Scratch Tapes: None

11. Storage Required: Total Storage required is $3F8_{16}$ Bytes.

12. Subroutine User: GPRNT1

13. Subroutine Required:

EUTL9
TSUM

14. Remarks: None

1. Subroutine Name: REACTP

2. Purpose: This routine controls the printing of reaction.

3. Equations and Procedures: Subroutine DISPL1 is called to print out reactions for each load condition.

4. Input Arguments:

NREF	-	number of reference points
NDIR	-	number of directions
NDEG	-	number of types of degrees of freedom
NLOAD	-	number of load conditions
NMDB	-	number of degrees of freedom in a reduced system
NSETS	-	data set number of reaction matrix
LIST	-	decoding list to go from reduced degrees of freedom to total degrees of freedom
REACT	-	array containing reactions
EZERO	-	effective zero for suppression
ROW	-	row label
COLMS	-	array of column labels
KPRT	-	code denotes reaction print
NWORK	-	number of words available in working storage
WORK	-	working storage

5. Output Arguments: None

6. Error Returns: None

7. Calling Sequence:

```
REACTP(NREF,NDIR,NDEG,NLOAD,NMDB,NSETS,LIST,REACT,EZERO,
      ROW,COLMS,KPRT,NWORK,WORK)
```

8. Input Tapes: NSETS

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage Required is 322_{16} Bytes.

12. Subroutine User: GPRNT1

13. Subroutine Required: DISPL1

14. Remarks: None

1. Subroutine Name: DISPPR

2. Purpose: This routine controls the printing of the displacements.

3. Equations and Procedures: Subroutine DISPL1 is called to print out displacements for each load condition.

4. Input Arguments:

NREF	-	number of reference points
NDIR	-	number of directions
NDEG	-	number of types of degrees of freedom
NLOAD	-	number of load conditions
NMDB	-	number of degrees of freedom in reduced system
NSETS	-	data set number of displacement matrix
LIST	-	array for boundary conditions. Decoding list to go from reduced degrees of freedom to total degrees of freedom.
DISPL	-	array containing displacements
EZERO	-	effective zero for suppression
ROW	-	row label
COLMS	-	array of column labels
KPRT	-	code denoting displacement print
NWORK	-	number of words of available working storage
WORK	-	working storage

5. Output Arguments: None

6. Error Returns: None

7. Calling Sequence:

```
DISPPR(NREF,NDIX,NDEG,NLOAD,NMDB,NSETS,LIST,DISPL,EZERO,
      ROW,COLMS,KPRT,NWORK,WORK)
```

8. Input Tapes: NSETS

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is 322_{16} Bytes.

12. Subroutine User: GPRNT1

13. Subroutine Required: DISPL1

14. Remarks: None

1. Subroutine Name: EIGPPR

2. Purpose: This routine controls the printing of eigenvalues and eigenvectors.

3. Equations and Procedures: Subroutine DISPL1 is called to print out eigenvalues and eigenvector for each eigenvalue.

4. Input Arguments:

NREF	-	number of reference points
NDIR	-	number of directions
NDEG	-	number of types of degrees of freedom
NEVAL	-	number of eigenvalues
NMDB	-	length of eigenvector array
NSETS	-	data set number of eigenvector matrix
LIST	-	decoding list to go from reduced degrees of freedom to total degrees of freedom
DISPL	-	array containing eigenvector
EIGVAL	-	array containing eigenvalues
EZERO	-	effective zero for suppression
ROW	-	row label
COLMS	-	array of column labels
KPRT	-	code denoting eigenprint
NWORK	-	number of words available in working storage
WORK	-	working storage

5. Output Arguments: None

6. Error Returns: None

7. Calling Sequence:

```
EIGPPR(NREF,NDIR,NDEG,NEVAL,NMDB,NSETS,LIST,DISPL,EIGVAL,
      EZERO,ROW,COLMS,KPRT,NWORK,WORK)
```

8. Input Tapes: NSETS

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is 344_{16} Bytes.

12. Subroutine User: GPRNT1

13. Subroutine Required: DISPL1

14. Remarks: None

1. Subroutine Name: MATPRT

2. Purpose: This routine controls the printing of the USER matrix.

3. Equations and Procedures: Subroutine DISPL1 is called to print each column of the user matrix.

4. Input Arguments:

NREF	-	number of reference points
NDIR	-	number of directions
NDEG	-	number of types of degrees of freedom
NLOAD	-	number of columns
NMDB	-	length of rows
NSETS	-	data set number of USER matrices
LIST	-	decoding list to go from reduced degrees of freedom to total degrees of freedom
DISPL	-	array containing user matrices
EZERO	-	effective zero for suppression
NAME	-	name of input matrix
ROW	-	row label
COLMS	-	array containing column label
KPRT	-	code denoting user matrix print
NWORK	-	number of words available in working storage
WORK	-	working storage

5. Output Arguments: None

6. Error Returns: None

7. Calling Sequence:

```
MATPRT(NREF,NDIR,NDEG,NLOAD,NMDB,NSETS,LIST,DISPL,EZERO,
      NAME,ROW,COLMS,KPRT,NWORK,WORK)
```

8. Input Tapes: NSETS

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is 366_{16} Bytes.

12. Subroutine User: GPRNT1

13. Subroutine Required: DISPL1

14. Remarks: None

1. Subroutine Name: DISPL1

2. Purpose: To print reactions, displacements, eigenvectors, user matrices, and calculate and print eigenvalues and frequency.

3. Equations and Procedures: The value of KPRT is tested to see if the eigenvalue frequency must be calculated and to write out correct heading then the input matrix is decoded and printed out.

4. Input Arguments:

NMDB	-	number of degrees of freedom in reduced system
EZERO	-	effective zero suppression code
DISPL	-	input matrix to be printed
LIST	-	decoding list to go from reduced degrees of freedom to total degrees of freedom
NREF	-	number of reference points
NDEG	-	number of types of degrees of freedom
NLOAD	-	load condition number
ROW	-	row label
TITLE	-	column label
KPRT	-	code indicating types of print
EXTRA	-	contains name of input matrix or eigenvalues
DISP	-	working storage

5. Output Arguments: None

6. Error Returns: None

7. Calling Sequence:

```
DISPL1(NMDB,EZERO,DISPL,LIST,NREF,NDIR,NDEG,NLOAD,ROW,TITLE,
      KPRT,EXTRA,DISP)
```

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is $9C6_{16}$ Bytes.

12. Subroutine User: REACTP, DISPPR, EIGPPR, MATPRT

13. Subroutine Required: None

14. Remarks: None

1. Subroutine Name: REPLAS

2. Purpose:

Whenever B matrix element equals corresponding elements of C or whenever element of C = 0, the output matrix A will contain a direct mapping of B. When B elements are not equal to corresponding eleme ts of C, elements of A equal those non equal elements of C (excluding C= 0.0)

3. Equations and Pr. edures:

a) input matrices B and C are copied onto two scratch tapes

b) Record #I is read from input matrix B

c) Record #J is read from input matrix C

d) If record #I words equal record #J words, then the output record equals Record #I

e) If record #I words are not equal to Record #J words then the output record is equal to Record J

f) If Record J = 0 then the output record will equal Record I

4. Input Arguments:

NMOUT	-	number of output matrices
NAMOT	-	names of output matrices
IODS	-	unit numbers for output matrices
NMIN	-	number of input matrices
INPT	-	names of input matrices
INSP	-	unit numbers for input matrices
NSCR	-	number of scratch units
ISSP	-	unit numbers for scratch files
NMSCL	-	number of scalers
SC	-	scaler array (not used)
NWKR	-	length of work array
WKR	-	work storage array
IPRINT	-	print control (not used)

5. Output Arguments:

ERR - error return control

6. Error Returns:

ERR = .TRUE. if input matrix cannot be located

7. Calling Sequence:

Call REPLAS (NMOUT,NAMOT,IODS,NMIN,INPT,INSP,NSCR,ISSP,NMSCL, SC,ERR,NWKR,WKR,IPRINT)

8. Subroutine User: EXEQ

9. Subroutine Required:

EUTL3
EUTL5
EUTL6

1. Subroutine Name: HSUM

2. Purpose: A control subroutine which is used to call subroutine SMDIS and SUMSTR to compute the sum of displacement, the sum of reaction and the sum of stress.

3. Equations and Procedures:

   (a) Retreive the number of harmonic and circumferential bound constants from system matrix.

   (b) Set control variable, then call subroutine SUMDIS and subroutine SUMSTR

4. Input Arguments:

   NUMOT - the number of output matrices (3)
   NAMIO - array containing the names of output matrices
   IOSPEC - array containing the output data set number
   NUMIN - the number of input matrices (4)
   INSPEC - array containing input data set numbers
   NUMSR - the number of scratch data set
   ISSPEC - array containing scratch data set numbers
   NUMSC - the number of input scalars
   ISCALE - used by EXEO
   ERROR - error return code
   NWORK - the number of words available for work storage
   WORK - working storage array

5. Output Arguments: ERROR - Error Flag

6. Calling Sequence:

   (NUMOT,NAMIO,IOSPEC,NUMIN,NAMIN,INSPEC,NUMSR,ISSPEC,NUMSC,
   ISCALE,ERROR,NWORK,WORK)

7. Input Tapes: INSPEC

8. Output Tapes: None

9. Scratch Tapes: None

10. Subroutine User: EXEO

11. Subroutine Required: EUTL3, SUMSTR, SUMDIS

1. Subroutine Name: HDECO

2. Purpose:

   (a) To update the harmonic control loop matrix

   (b) Extract the element stiffness matrix, the element load matrix from the master element stiffness matrix

3. Equations and Procedures:

   (a) Read the harmonic number into core and decrease this number by one. If this number is equaled to zero then a null harmonic control loop matrix will be formed. There will be a harmonic loop control matrix output with the dimension of IXI.

   (b) Element stiffness matrix and the element load matrix will be extracted from the master element stiffness matrix. The extraction is dependent upon the harmonic number.

4. Input Arguments:

   NUMOT = the number of output matrices (2)
   NAMIO = array containing the names of output matrices (2)
   IOSPEC = array containing output data set numbers (2)
   NUMIN = the number of input matrices (3)
   NAMIN = array containing the names of the input matrices (3)
   INSPEC = array containing input data set numbers
   NUMSR = the number of scratch data sets (1)
   ISSPEC = array containing scratch data set numbers (1)
   NUMSC = the number of input scalars
   ISCALE = used by EXEO
   ERROR = error return code
   NWORK = the number of words available for work storage
   WORK = working storage array

5. Output Argument: Error - Error Flag

6. Calling Sequence:

   (NUMOT, NAMIO, IOSPEC, NUMIN, NAMIN, INSPEC, NUMSR, ISSPEC, NUMSC, ISCALE, ERROR, NWORK, WORK)

7. Input Tapes: INSPEC

8. Output Tapes: IOSPEC

9. Scratch Tapes: ISSPEC

10. Subroutine User: EXEC

11. Subroutine Required: EUTL3, EUTL5, EUTL6

1. Subroutine Name: SUMSTR

2. Purpose: To compute the sum of stress for each element for a given circumferential bounds, then output the results.

3. Equations and Procedures:

   (a) Compute the working storages which require retreiving all the element stress from the input matrix and retaining them in core.

   (b) Compute the sum of stress of each element.

   (c) Output the sum of stress for each element.

4. Input Arguments:

   NUMOT - the number of output matrices (2)
   NIMIO - array containing the names of the input matrices
   IOSPEC - array containing the output data set numbers
   NUMIN - the number of input matrices
   INSPEC - array containing the input data set numbers
   NUMSR - the number of scratch data set
   ISSPEC - array containing the scratch data set numbers
   NUMSC - used by EXEO
   ISCALE - used by EXEO
   IERROR - error control flag
   NWORK - the number of words available for the work storage
   WORK - working storage array
   ALLEM - number of element
   INUM - number of the working storages has allocated

5. Error Returns:

   IERROR = 21 Does not have enough working storage to be allocated to retreive all the element stress from the input matrix

   IERROR = 15 Unable to find the input matrix

6. Calling Sequence:

   (NUMOT,NAMIO,IOSPEC,NUMIN,NAMIN,INSPEC,NUMSR,ISSPEC,MUMSC, ISCALE,IERROR,NWORK,WORK,ALLEM,INUM)

7. Input Tapes: INSPEC

8. Output Tapes: IOSPEC

9. Scratch Tapes: None

10. Subroutine User: HSUM

11. Subroutine Required: EUTL3, EUTL9, EUTL5, EUTL8, EUTL6

1. Subroutine Name: SUMDIS

2. Purpose: To compute the sum of displacement and the sum of reactions for a given circumferential bounds, then output the results

3. Equations and Procedures:

   (a) Compute the working storages required to retreive all the displacement or all the reactions from the input matrix and retain them in core.

   (b) Compute the sum of displacements and the sum of reactions.

   (c) Output the sum of displacement or the sum of reaction.

4. Input Arguments:

   NUMOT - the number of output matrices (2)
   NIMIO - array containing the names of the input matrices
   IOSPEC - array containing the output data set numbers
   NUMIN - the number of input matrices
   INSPEC - array containing input data set numbers
   NUMSR - the number of scratch data set
   ISSPEC - array containing scratch data set numbers
   NUMSC - used by EXEO
   ISCALE - used by EXEO
   NWORK - the number of words available for work storage
   WORK - working storage array
   DNEGI - number of degree of freedom
   INUM - number of working storages allocated
   SUMT - sum displacements and sum reactions based on computation control variable

5. Error Returns:

   IERROR = 21 Not enough working storage to hold all the displacements or all the reactions in core

   IERROR = 15 unable to find the input matrix

6. Calling Sequence:

   (NUMOT,NAMIO,IOSPEC,NUMIN,NAMIN,INSPEC,NUMSR,ISSPEC,NUMSC, ISCALE,IERROR,NWORK,WORK,DNEGI,INUM,SUMT)

7. Input Tapes: INSPEC

8. Output Tapes: IOSPEC

9. Scratch Tapes: None

10. Subroutine User: HSUM

11. Subroutine Required: EUTL3, EUTL9, EUTL5, EUTL8, EUTL6

1. Subroutine Name: CHEQS

2. Purpose: Main execution routine Cholesky Equation Solver Abstraction Instruction

3. Equations and Procedures:

   (a) Locates input matrices and transfers them to separate scratch units. Coefficient matrix is converted to banded form (by repeated calls to AKNZ)

   (b) Triangularizes coefficient matrix by call to TCONTX place triangularized matrix as output matrix NAMOT(1,1) on unit IODS (1,1)

   (c) Allocates storage for equation solver calls BEQSX which generates solution matrix. Solution matrix placed as output matrix NAMOT (1, NMOUT) on unit IODS (1,NMOUT)

4. Input Arguments:

NMOUT	-	Number of output matrices
NAMOT	-	Names of output matrices
IODS	-	Units for output matrices
NMIN	-	Number of input matrices
INPT	-	Names of input matrices
INSP	-	Units for input matrices
NSCR	-	Number of scratch units
ISSP	-	Logical unit numbers for scratch units
NMSCL	-	Number of scalars input
SC	-	Input scalars
NWKR	-	Working storage size
WKR	-	Actual working storage
IPRINT	-	Print control

5. Output Arguments:

ERR	-	Logical variable indicating error

6. Error Returns: None

7. Calling Sequence:

```
Call CHEQS (NMOUT,NAMOT,IODS,NMIN,INPT,INSP,NSCR,ISSP,NMSCL,
            SC,ERR,NWKR,WKR, IPRINT)
```

8. Subroutine User: EXEQ - Instruction Execution Control Routine

9. Subroutine Required:

   EUTL3, EUTL9, AKNZ, TCONTX, EUTL8, BEQSX, EUTL5, EUTL6

1. Subroutine Name: AKNZ

2. Purpose: Store A - Matrix in banded column form

3. Input Arguments:
   ACOL,I3,IZR,NZEL,ICOL,N,ICND

4. Error Returns: None

5. Calling Sequence:
   Call AKNZ (ACOL,I3,IZR,NZEL,ICOL,N,ICND

6. Output Tapes: I3

7. Scratch Tapes: None

8. Subroutine User: CHEQS

9. Subroutine Required: None

1. Subroutine Name: TCONTX

2. Purpose: This routine controls tape flow for the triangularization routine.

3. Equations and Procedures:

   (1) Controls for setting up computation passes are computed in ICALC and ISTRT.

   (2) A portion of the input matrix A is read in from MTAPE.

   (3) This information is given to the routine TTRI which actually performs the triangularization for row numbers ISTRT to ICALC.

   (4) This triangularized output portion in A is stored on tape NTAPE.

   (5) Computation is repeated for each portion of the matrix until all rows are completed.

4. Input Arguments:

   N = order of system to be handled
   IZR = banding information array
   NZEL = banding information array
   A = storage array for input row of banded matrix which is read by routine
   NTOTAL = total number of words which can be considered as a "core-full"
   ATRI = intermediate storage array equals length of maximum order
   MTAPE = input tape logical number
   NTAPE = output tape logical number

5. Output Arguments:

   IERROR = error indication value
   WS = accumulative determinant

6. Error Returns: IERROR not = 0 if WS is returned from TTRI as less than zero

7. Calling Sequence: CALL TCONTR (N, IZR, NZEL, A, NTOTAL, ATRI, MTAPE, NTAPE, IERROR, WS

8. Input Tapes: MTAPE = input matrix A in banded row form. Each row equals 1 record

9. Output Tapes: NTAPE = triangularized matrix T in banded row form. Each row equals 1 record.

10. Scratch Tapes: None

1. Subroutine Name: TTRI

2. Purpose: To triangularize rows ISTRT to ICALC of a banded Matrix A.

3. Equations and Procedures:

(1) This routine triangularizes rows ISTRT to ICALC of a banded matrix A where rows 1 to ISTRT =1 of the A matrix (already triangularized) are on tape NTAPE.

(2) If ISTRT = 1, then NTAPE and work storage ATRI are not used since A is assumed to be in core.

(3) Procedure:

Using Cholesky technique, the off diagonal terms of the portion in core are triangularized. Off diagonals are then computed. Output is stored in array A.

(4) Cholesky equations:

(1) $s_{11} = (a_{11})^{\frac{1}{2}}$ (3) $s_{ii} = (a_{ii} - \sum_{l=1}^{i-1} s_{li}^2)^{\frac{1}{2}}$ $i > 1$

(2) $s_{ij} = \frac{a_{ij}}{s_{11}}$ (4) $s_{ij} = \frac{a_{ij} - \sum_{l=1}^{i-1} s_{li}\, s_{lj}}{s_{ii}}$, $j > i$

(5) $s_{ij} = 0$, $i > j$

4. Input Arguments:

ISTRT = beginning row of triangularized portion
ICALC = end row of triangularized portion
IZR = banding information array
NZEL = banding information array
NTAPE = logical tape number of input tape. NTAPE = 7
A = storage array for input A and also output array
ATRI = working storage array

5. Output Arguments: A = output array

6. Error returns: IERROR = I = row number such that WS = s_{ii} is not greater than zero.

7. Calling Sequence: CALL TTRI (ISTRT, ICALC, IZR, NZEL, NTAPE, A, ATRI, IERROR, WS)

8. Input Tapes: NTAPE = tape which contains already triangularized rows of matrix.

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: A(1), ATRI(1), NZEL(1), IZR(1)

12. Subroutine User: TCONTX

13. Subroutines Required: None

1. Subroutine Name: BEQSX

2. Purpose: To perform simultaneous equation solution for banded symmetric matrix input using Cholesky procedure.

3. Equations and Procedures:

   (1) Tape I10 which contains the assembled load columns is spaced down to the appropriate position for reading.

   (2) A loop is set up based on NL load conditions. Each load is considered separately.

   (3) The program is designed so that a "core-filled" piece is considered at one time. The procedure to handle this is set up.

   (4) Each load is read into PCOL from Tape I10 and the known displacements are placed into their appropriate position in DISPL

   (5) A call to ESCONT is made. This is the routine which actually does the computation.

   (6) Displacement equals answers are written on Tape I12.

   (7) Steps 6-9 are repeated for each load condition.

4. Input Arguments: CK1I = FORMAT tape compatible matrix name

5. Output Arguments: None

6. Error Returns: None

7. Calling Sequence: CALL BEQSX(AK,NZEL,IZR,XK,DISPL,PCOL,RSUM, NROW,I3,I7,I10,NMDB,I12, NTOTAL,NL)

8. Input Tapes: I10 = assembled loads
   7 = input triangularized banded matrix array

9. Output Tapes: I12 = displacements

10. Scratch Tapes: None

11. Storage Required: XK (2000) NROW(250)

12. Subroutine User: CHEQS

13. Subroutines Required: ESCONT

1. Subroutine Name: XCALK

2. Purpose: Solves for X where A*X = XK and A is upper triangular matrix.

3. Equations and Procedures: Determines ISTRT to ICALC of the column vector X in the matrix equations A*X = XK where A is upper triangular matrix and X and XK are column vectors

$$X(N) = \frac{XK(N)}{A_{nn}} \qquad X(I) = XK(I) - \sum_{L=I+1}^{N} A_{IL}XL \qquad I<N$$

This constitutes the second part in calculating an equation solution by Cholesky or "square root" method.

4. Input Arguments:

N = order of system
ISTRT = beginning row number of computation pass
ICALC = end row number of computation pass
IZR = number zero elements in row of reduced matrix
NZEL = cumulative total of nonzero elements from row 1 thru i - 1 of reduced matrix
A = storage array for matrix
XK = column vector array

5. Output Arguments:

X = output vector array
RSUM = updated intermediate array

6. Error Returns: None

7. Calling Sequence: CALL XCALK (N, ISTRT, ICALC, IZR, NZEL, A, XK, RSUM, X)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: A(1),XK(1),RSUM(1),X(1),IZR(1),NZEL(1)

12. Subroutine User: ESCONT

13. Subroutines Required: None

1. Subroutine Name: ESCONT

2. Purpose: Solves matrix equation A * A transpose * X = F for X

3. Equations and Procedures: The matrix equation A * A transpose X = F is solved for X by solving the two matrix equations
A*XK= F
A TRANSPOSE * X = XK
Where A is a banded lower triangular matrix and X and F are column vectors.

Procedure:

(1) Rows ISTRT to ICALC of A are read from Tape 7.

(2) A call to KCALC routine computes XK.

(3) A call to XCALK routine computes X.

(4) Steps 1, 2, and 3 are repeated for each pass.

4. Input Arguments:

N	=	order of system
NPASS	=	number of computation passes necessary
NROW	=	array for control of computation passes
IZR	=	banding information array
NZEL	=	banding information array
A	=	storage array for input matrix
F	=	storage array for input column
NTAPE	=	input tape logical number = I7
XK	=	working storage array
RSUM	=	working storage array

5. Output Arguments: X = output answer column array

6. Error Returns: None

7. Calling Sequence: CALL ESCONT (N, NPASS, NROW, IZR, NZEL, A, F, X, XK, RSUM, NTAPE)

8. Input Tapes: Tape I7 contains input triangular matrix A. A is in banded form. Each row is a separate record.

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: NROW(1),IZR(1),NZEL(1),A(1),F(1),X(1),XK(1), RSUM(1)

12. Subroutine User: BEQSX

13. Subroutines Required: KCALC
XCALK

1. Subroutine Name: KCALC

2. Purpose: Solve for XK where A*XK = F and A is a banded lower triangular matrix.

3. Equations and Procedures: Determine elements ISTRT to ICALC of the column vector XK in the matrix equations A*XK = F

$$XK(1) = \frac{F_1}{A_{11}} \qquad XK(I) = F(I) - \sum_{L=1}^{I-1} A_{LI}\, XK(L) \quad I>1$$

This constitutes the first step in equation solution by Cholesky or "square root" method.

4. Input Arguments:

ISTRT = beginning row number of computation pass
ICALC = end row number of computation pass
IZR = number zero elements in row of reduced matrix
NZEL = cumulative total of nonzero elements in rows 1 thru i - 1 of reduced matrix

A = storage array for input matrix
F = column vector array

5. Output Arguments: XK = output vector array

6. Error Returns: None

7. Calling Sequence: CALL KCALC (ISTRT, ICALC, IZR, NZEL, A, F, XK)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: A(1),XK(1),F(1),IZR(1),NZEL(1)

12. Subroutine User: ESCONT

13. Subroutines Required: None

1. Subroutine Name: ANALIC

2. Purpose: Driver routine for ANALIC Module System constants are defined in core and the storage required is calculated. If sufficient storage exists for this problem, execution continues with the actual storage allocation in routine STORIC.

3. Equations and Procedures: None

4. Input Arguments:

NUMOUT - Number of output matrices
NAMOUT - Array containing output matrix names
IODS - Array containing output matrix logical units
NUMIN - Number of input matrices
NAMIN - Array containing input matrix names
INSPEC - Array containing input matrix logical units
NUMSCR - Number of scratch data sets available
ISSPEC - Array containing scratch data set units
NUMSCL - Number of scalars
NAMSCL - Array containing scalars
NWORKR - Number of words of available working storage
W - Array of working storage

5. Output Arguments:

IERROR - Error flag

6. Error Returns:

IERROR = 21 - input matrix 2 could not be found
= 15 - error occurred somewhere in ANALIC
= 0 - no error

7. Calling Sequence:

```
Call ANALIC (NUMOUT, NAMOUT, IODS, NUMIN, NAMIN, INSPEC,
             NUMSCR, ISSPEC, NUMSCL, NAMSCL, IERROR, NWORKR,
             W)
```

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: $A2A_{16}$ bytes

12. Subroutine User: EXEC

13. Subroutines Required:

WCLOCK, EUTL3, SIZEIC, STORIC, TIMER

14. Remarks: None

1. Subroutine Name: SIZEIC

2. Purpose: To determine the total amount of storage needed in local arrays and in working storage for the problem to be solved.

3. Equations and Procedures:

<u>For Equation Solution Only</u>

Local storage must be less than 4000

$$(2*NLV+1)*NMDB \leq 4000$$

Common working storage must be less than NWORKR

$$(NMDB*(NMDB+1))/2 \leq NWORKR$$

<u>For Total Statics Problem</u>

Local storage required must be less than 4000

$$2*NMDB2+NORDM+NNOM+2*MAX(NORDM,NRSELM)+2NSYS + MAX((NORDM*(NORDM+1))/2, NORDM*NRSELM)+1 + NLV*(1+NMDB+NMDB2) \leq 4000$$

NLV is reduced to 1 to try to make problem fit. Common working storage must be less than NWORKR.

$$MAX([NR*NELEM+(NMDB2+(NMDB2+1))/2], [2*NORDM+NRSELM+13+3(((NORDM*(NORDM+1))/2 + NNOM+NELEM*NRSELM] \leq NWORKR$$

4. Input Arguments:

NMDB	- Order of the reduced system
NLV	- Number of load conditions (may be reduced on return)
NREF	- Number of reference points
NORDM	- Largest number of rows in element stiffness matrix
NNOM	- Largest number of node points for any element
NMDB2	- Number of 1's and 2's in system
NRSELM	- Maximum number of rows in element stress matrix
NELEM	- Number of elements
NSYS	- Total number of unreduced system degrees of freedom
NWORKR	- Number of words of available working storage
KALC	- Calculation control
EQUAT	- Logical equation solver only indicator
ERROR	- Error flag

5. Output Arguments: None

6. Error Returns:

   ERROR - True if problem will not fit in core

7. Calling Sequence:

```
Call SIZEIC (NMDB,NLV,NREF,NORDM,NNOM,NMDB2,NRSELM,NELEM,
             NSYS,ERROR,NWORKR,KALC,EQUAT)
```

8. Input Tapes: None

9. Output Tapes: NPOT

10. Scratch Tapes: None

11. Storage Required: $80E_{16}$ Bytes

12. Subroutine User: ANALIC

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: STORIC

2. Purpose: To define the storage for all arrays used by ANALIC.

3. Equations and Procedures:

Storage is dynamically allocated in both the local array and common working storage. Starting locations are defined in the large local and common area and passed thru the argument list to the KALCON routine.

4. Input Arguments:

NMDB - Order of reduced system
NLV - Number of load conditions
NREFV - Number of reference points
NORDM - Largest order of element stiffness
NNOM - Maximum number of nodes for any element
NRSELM - Maximum number of rows in any element stress matrix
NMDB2 - Number of 1's and 2's
REPORT - Array used to store printout information
NOUT - Scalar used with REPORT
KALC - Calculation control for equation solver
NSYS - Total number of unreduced system degrees of freedom
NMDBO - Number of bounded out points OS
INSPEC - Array containing input matrix logical units
NAMIN - Array containing input matrix names
NAMSCL - Array containing input scalars
IODS - Array containing output matrix logical units
ISSPEC - Array containing input matrix logical units
NAMOUT - Array containing output matrix names
U - Work array of order NWORKR
SC - Array containing system constants
NELEM - Number of elements
EQUAT - Equation solver only indicator
IERROR - Error flag

5. Output Arguments: None

6. Error Returns: IERROR

7. Calling Sequence:

```
Call STORIC (NMDB,NLV,NREFV,NORDM,NNOM,NRSELM,NMDB2,REPORT,
             NOUT,KALC,NSYS, NMDBO,INSPEC,NAMIN,IERROR,NAMSCL,
             IODS,ISSPEC,NAMOUT,U,SC,NELEM,EQUAT)
```

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: $47A4_{16}$ Bytes

12. Subroutine User: ANALIC

13. Subroutines Required: INPTIC, KALCON

14. Remarks: None

1. Subroutine Name: INPTIC

2. Purpose: Reads input matrices into core.

3. Equations and Procedures:

   A. Define input matrix logical unit numbers.

   B. If TR matrix is present, have total statics problem.

      1. Generate LIST by calling LISTIC routine.

      2. If PR present, read in W array
         If not present W1(I) = 0.0 and go to 5.

      3. If SUBL present, read into FCOL array.
         If not W1(I) = W(I) (Trans matrix)

      4. If SUBL present compute W1(FCOL(I)) = W(I).

      5. Read in PCOL array.

      6. Form TOTAL load column

         PCOL(IL) = W(JL) + W1(JL)

      7. Compute TRST array = [TRANS] [$LIST^{-1}$].

      8. Read column of AK into W1. Use TRST to map column of W_1 into AK stored lower half symmetric.

   C. If TR matrix is not present, just equation solve

      1. Read PR matrix into DISPL array (NL columns).

      2. Read AK matrix into core (lower half symmetric).

   D. Read values of prescribed displacements into core, if present.

4. Input Arguments:

LIST	- Array which maps reduced to unreduced D.O.F.
CONOPT	- Load scalar
PCOL	- External load column
DISPL	- Prescribed displacement column
W	- Working storage
NSYS	- Total system unreduced degrees of freedom
NMDBO	- Number of bounded out degrees of freedom
NMDB2	- Number of 1's and 2's
NL	- Number of load conditions
NAMIN	- Array containing input matrix names
INSPEC	- Array containing input matrix logical units
IERROR	- Error flag
NAMSCL	- Array containing scalars
NMDB	- Order of reduced system
W1	- Working storage
AK	- Stiffness matrix
IW	- Working storage
TRAN	- Working storage for trnasformation matrix

5. Output Arguments: None

6. Error Returns:

   IERROR - Set to i*10+1 if input matrix i cannot be found.

7. Calling Sequence:

```
Call INPTIC (LIST,CONOPT,PCOL,DISPL,W,NSYS,NMBDO,NMDB2,NL,
             NAMIN,INSPEC,IERROR,NAMSCL,NMDB,W1,AK,IW,TRAN)
```

8. Input Tapes:

   NAMIN(1,1), (1,4), (1,5), (1,6), (1,7), (1,8)

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: $133A_{16}$ bytes

12. Subroutine User: STORIC

13. Subroutines Required:

    UNITNM, LISTIC, INPTI1, EUTL3, EUTL9

14. Remarks: None

1. Subroutine Name: INPTI1

2. Purpose: Read a column of a specified input tape into core in full format.

3. Equations and Procedures:

   1. Locate matrix on specified tape by calling EUTL3.
   2. Uncompress column if necessary by calling EUTL9.

4. Input Arguments:

   NAME - Input matrix name sought
   IN - Input matrix logical unit
   WORK - Working storage
   MAXW - Number of words of working storage

5. Output Arguments:

   IMAX - Number of rows in matrix found
   JMAX - Number of columns in matrix found
   ERROR - Error flag

6. Error Returns:

   ERROR - True if matrix could not be found

7. Calling Sequence:

   Call INPTI1 (NAME,IMAX,JMAX,IN,ERROR,WORK,MAXW)

8. Input Tapes: IN

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: $2B6_{16}$ Bytes

12. Subroutines Used:

    INPTIC

13. Subroutines Required:

    EUTL3, EUTL9

14. Remarks: None

1. Subroutine Name: LISTIC

2. Purpose: Generate values for LIST array from the USER04 TR matrix.

3. Equations and Procedures:

   1. Locate TR matrix.
   2. Read values into core.
   3. Eliminate IROW values $\leq$ NMDBO.
   4. Sort the remaining terms by IVAL values.

4. Input Arguments:

IA	- Working storage of length NSYS
IB	- Working storage of length NSYS
NMDB2	- Number of 1's and 2's
NMDBO	- Number of 0's
NSYS	- Number of 0's plus 1's plus 2's
NAME	- Array containing input matrix name
IN1	- Input matrix logical unit

5. Output Arguments:

LIST	- Array which maps reduced to system D.O.F.'s
ERROR	- Error flag

6. Error Returns:

   ERROR - True if input matrix cannot be found

7. Calling Sequence:

   Call LISTIC(IA,IB,LIST,NMDB2,NMDBO,NSYS,NAME,IN1,ERROR)

8. Input Tapes: IN1

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: $41C_{16}$ Bytes

12. Subroutine Used: STORIC

13. Subroutines Required: EUTL3

14. Remarks: None

1. Subroutine Name: UNITNM

2. Purpose: To generate input or output matrix names and logical units based on only the matrices present.

3. Equations and Procedures:

Check the NAMSCL array which represents all input and output matrices. If the value of NAMSCL(I) = 1, then matrix I was present; if NAMSCL(I) = 0, the matrix was suppressed. Search for only the matrices present and define their matrix position and logical unit in arrays MATN and UNIT, respectively.

4. Input Arguments:

NAMSCL - Array containing input scalars
MAXN - Number of NAMSCL values to be searched
ISPEC - Array containing matrix logical units

5. Output Arguments:

NP - Number of matrices present
MATN - Array containing present matrix names
UNIT - Array containing present matrix logical units

6. Error Returns: None

7. Calling Sequence:

Call UNITNM(NAMSCL,MAXN,ISPEC,NP,UNIT,MATN)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: 226_{16} Bytes

12. Subroutine Used:

KALCON, INPTIC

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: KALCON

2. Purpose: To control solution of statics problem in core. Displacements, stresses, forces, and reactions are computed.

3. Equations and Procedures:

If complete STATICS problem,

(a) Read element properties.
(b) Assemble stiffness matrix.
(c) Solve for displacements.
(d) Generate stresses and forces.
(e) Generate reactions.

If equation solve only,

(a) Solve system of equations.

4. Input Arguments:

DISPL
PCOL
LIST
CONOPT
LISTEL
NODE
SIGEL
SIGELN
W
AKEL
AK
FTEL
FCOL
AF
W1
SEL
SZALEL

} Arrays which define storage locations to be used inside subroutine. Actual values need not be present when routine is called. The values will be generated inside the routine.

5. Output Arguments:

REPORT - Array used for time printout later
NOUT - Scalar used for time printout later
NL - Number of load conditions
KALC - Calculation control for equation solver
NORDM - Number of rows in largest element stiffness matrix
NAMSCL - Array containing scalars

5. Output Arguments, Continued:

IODS	- Array containing output matrix logical units
ISSPEC	- Array containing scratch matrix logical units
NAMIN	- Array containing input matrix names
NAMOUT	- Array containing output matrix names
INSPEC	- Array containing input matrix logical units
IERROR	- Error flag
SC	- Array containing system constants
EQUAT	- Equation solver indicator

6. Error Returns: None

7. Calling Sequence:

```
Call KALCON (DISPL,PCOL,LIST,CONOPT,LISTEL,NODE,SIGEL,SIGELN,
             W,AKEL,AK,FTEL,FCOL,AF,W1,SEL,SZALEL,REPORT,NOUT,
             NL,KALC,NORDM,NAMSCL,IODS,ISSPEC,NAMIN,NAMOUT,
             INSPEC,IERROR,SC,EQUAT)
```

8. Input Tapes: NAMIN

9. Output Tapes: NAMOUT

10. Scratch Tapes: ISSPEC(1,1)

11. Storage Required: $148E_{16}$ bytes

12. Subroutine User: STORIC

13. Subroutines Required:

UNITNM,EUTL1,EUTL4,WCLOCK,ASMAIC,ASMPIC,DSPCIC,DISPIC,
EUTL5,EUTL6,FARCIC,RECTIC,STRSIC

14. Remarks: None

1. Subroutine Name: ASMAIC

2. Purpose: Assembles and reduces the following matrices in-core based on the value of KODE.

KODE	Matrices Assembled and Reduced
1	Stiffness and Element Applied Loads
2	Stiffness
3	Element Applied Loads
4	Mass

3. Equations and Procedures:

The value of KODE is determined and control passes to one of the three areas to perform the assembly and reduction. Next, the appropriate element matrices and the LISTEL array are read from tape K4 for the first element. Assembly and reduction is now performed using the LISTEL array and iterating across the matrix bottom half rows and columns process is repeated for each element.

For the case where both the stiffness matrix and the element applied loads are to be assembled (KODE=1), the presence of prescribed displacements causes the routine to form the product of the prescribed displacements and the corresponding elements in the stiffness matrix and then subtract this quantity from the load column. The reason for this can be seen below. If (1) represents our problem,

$$\begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} P_1 \\ P_2 \end{bmatrix} \qquad (1)$$

where X_2 represents the prescribed displacements, we can write

$$K_{11}X_1 + K_{12}X_2 = P_1$$

and since X_2 is known, our problem reduces to

$$K_{11}X_1 = P_1 - K_{12}X_2 \quad .$$

3. Equations and Procedures, Continued:

Thus K_{11} is our assembled and reduced stiffness matrix and $P_1 - K_{12}X_2$ is our assembled and reduced element applied load column.

If just the stiffness matrix or the mass matrix is to be processed (KODE=2 or 4), the assembly and reduction is straightforward based on LISTEL and iterating across the bottom half matrix row and columns. For the assembly of the mass matrices, this routine assumes that the point loads are already assembled on the diagonal of AK when AK is input.

If only the element applied load column is desired (KODE=3), the assembly and reduction is performed using LISTEL on the column.

4. Input Arguments:

K4	- Unit number of tape which contains element matrices
NMDB	- Order of assembled and reduced system
DISPL	- Prescribed displacement column (if required)
KODE	- Input parameter which controls which matrices are assembled
NAMIN	- Array containing input matrix names
NMDBO	- Number of 0's.

5. Output Arguments:

FCOL	- Assembled and reduced element applied load column
AK	- Assembled and reduced element matrix (symmetric stored in lower half by rows)
AKEL	- Element matrix
FTEL	- Element applied load column
T1	- Time assembly and reduction started
LISTEL	- Array which maps element coordinates into system coordinates
IERROR	- Error flag

6. Error Returns:

IERROR	- Non-zero if error occurs in ASMAIC

7. Calling Sequence:

```
Call ASMAIC(K4,NMDB,DISPL,FCOL,AK,AKEL,LISTEL,FTEL, T1,
            KODE,NAMIN,IERROR,NMDBO)
```

8. Input Tapes:

   INSPEC(1,3) - Contains element matrices

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: $(DFE)_{16}$ bytes

12. Subroutine User: KALCON

13. Subroutines Required:

    RDCLOK, EUTL3

14. Remarks: None

1. Subroutine Name: ASMPIC

2. Purpose: To print the following assembled matrices based on the value of KODE. The grid point number and degree of freedom are printed for each row of the symmetric matrix or column.

KODE	Matrices Printed
1	Stiffness and Element Applied Loads
2	Stiffness
3	Element Applied Loads
4	Mass
5	Incremental Stiffness
6	Inverse of Stiffness

3. Equations and Procedures:

(a) The time since the start of the assembly and reduction or inversion is calculated and printed out.

(b) If the inverse is requested, the matrix is checked for singularity and a message is printed and control returned if a matrix is singular.

(c) Appropriate labels are printed based on the value of KODE.

(d) The matrix is printed out as a symmetric matrix by rows (except load column). The grid point number and degree of freedom, based on the bounds table input for the problem, are also printed for each row.

(e) If the element applied load column was requested, it is printed in Engineering Format.

(f) Error checks are made for the stiffness matrix. A non-positive diagonal element in the stiffness matrix causes a message to be printed out and the variable ICOUNT (initially 0), to be incremented by +1. A positive value for ICOUNT is returned as an error return.

4. Input Arguments:

W - Array of working storage
NSYS - Number of system degrees of freedom
NREF - Number of reference points
AK - Assembled and reduced element matrix (symmetric, stored in lower half by rows)
FCOL - Element applied load column (if required)

4. Input Arguments, Continued:

NMDB	- Order of assembled/reduced system
T1	- Time when assembly or inverse started
ISING	- Indicator which reflects singularity of inverse of stiffness matrix
LIST	- Array which maps coordinates from assembled reduced to unassembled/unreduced coordinates
NDIR	- Number of directions per grid point
NDEG	- Number of degrees of freedom for system
KODE	- Input parameter which controls which matrix is printed.

5. Output Arguments:

ICOUNT	- Number of non-positive diagonal elements if stiffness matrix was printed

6. Error Returns:

ICOUNT	- Positive value reflects non-positive diagonals in stiffness matrix
KODE	- Value of KODE outside interval (1,6) is error and causes return

7. Calling Sequence:

```
Call ASMPIC(AK,FCOL,NMDB,T1,ISING,ICOUNT,LIST,NDIR,NDEG,KODE,
           NREF,NSYS,W)
```

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: $9F2_{16}$ bytes

12. Subroutine User: KALCON

13. Subroutine Required:

RDCLOK,DISPIC

14. Remarks: None

1. Subroutine Name: DSPCIC

2. Purpose: To control the calculation of the displacements for the analysis package in core.

3. Equations and Procedures:

For Total STATICS Problem:

(a) The total load column is formed in DISPL.

DISPL = PCOL + CONOPT*FCOL

(b) The total load column is printed by calling DISPIC.

(c) The displacements are obtained by one of the following methods based on the value of KALC.

KALC	SOLUTION
1	Inverse
2	Gauss Elimination
3	Cholesky Triangularization
4	Gauss Wavefront

For Equation Solve Only:

(a) Compute displacements as in (c) above.

4. Input Arguments:

AK	- Reduced stiffness matrix
FCOL	- Element applied load column
PCOL	- External load columns
CONOPT	- Load scalar array
NMDB	- Order of reduced system
NL	- Number of loading conditions
NOUT	- Scalar used for printout
NREF	- Number of input node points
REPORT	- Array used for time report printout
SEL	- Stress matrix
KALC	- Calculation control
LIST	- Array which maps reduced to system
NDIR	- Number of directions/grid point
NDEG	- Number of degrees of freedom
NSYS	- Total system unreduced degrees of freedom
WORK	- Working storage of length NSYS
NAME	- Array containing output matrix name
IO	- Output matrix logical unit
KR	- Output matrix generation indicator
NAMSCL	- Array containing scalars
EQUAT	- Equation solve only indicator

5. Output Arguments:

   DISPL - Displacement columns

6. Error Return:

   KALC < 0 - error condition appropriate message will be print out

7. Calling Sequence:

```
Call DSPCIC(AK,FCOL,PCOL,CONOPT,DISPL,NMDB,NL,NOUT,NREF,
            REPORT,SEL,KALC,LIST,NDIR,NDEG,NSYS,WORK,NAME,IO,
            KR,NAMSCL,EQUAT)
```

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: $(CFA)_{16}$ bytes

12. Subroutine User: ANALIC

13. Subroutines Required:

    WCLOCK,SINVIC,ASMPIC,MSBB,GELS,MFSD,MTDS,DISPIC
    WAVEFM,DECOMP,FWD,BACK

14. Remarks: None

1. Subroutine Name: DISPIC

2. Purpose: Print the following matrices in Engineering format based on the value of KP

KP	Matrix Printed
1	Element applied load column
2	Total load column
3	Displacements

Generate displacement matrix if KR is unequal to 0.

3. Equations and Procedures:

The components are sorted using the LIST array to identify the correct degree of freedom. They are printed according to the degree of freedom

U,V,W,THETAX,THETAY,THETAZ
or
FX,FY,FZ,MX,MY,MZ

for each grid point.

If KR ≠ 0, the output displacement matrix is generated.

4. Input Arguments:

NMDB - Order of reduced system
DISPL - Column to be printed
LIST - Array which maps reduced to system d.o.F's
NREF - Number of reference points
NDIR - Number of directions
NDEG - Number of degrees of freedom
NLOAD - Load condition number
NSYS - Number of unreduced system degrees of freedom
WORK - Work storage
NAME - Name of output matrix to be written
IO - Output matrix unit number
KP - Print control
KR - Output matrix generation control

5. Output Arguments: None

6. Error Returns: None

7. Calling Sequence:

```
Call DISPIC(NMDB,DISPL,LIST,NREF,NDIR,NDEG,NLOAD,NSYS,
            WORK,NAME,IO,KP,KR)
```

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: $88E_{16}$ bytes

12. Subroutine User:

DSPCIC, ASMPIC

13. Subroutines Required:

EUTL5, EUTL6

14. Remarks: None

1. Subroutine Name: FARCIC

2. Purpose: Generates, prints element forces, and assembles forces for one load condition.

3. Equations and Procedures:

Calculates element forces for each load condition, potential energy, work, and strain energy.

Calculations:

Apparent element force: $[\$IGEL] = \sum [AKEL]*[DI\$PL]$

Net element force: $[\$IGELN] = [\$IGEL] - CONOPT * [FTEL]$

Total work of element = energy

Applied loads: $WORKEM = WORKE$

where $WORKE = \sum [DI\$PL] * [FTEL] * CONOPS$

Total work of gridpoint

Applied load: $ENE\$M = -ENE\$$

where $ENE\$ = \sum [DI\$PL]*[PCOL]$

Total work: $WORKM = -WORK$

$= WORKE + ENE\$$

(work energy + strain energy)

where $WORKE = \sum [DI\$PL]*[FTEL] * CONOPS$

and ENE$ is defined above.

Total strain energy: $\$TREN = \sum .5 * [DI\$PL]*[\$IGEL]$

Net Energy: $ENERGY = \$TREN - WORK$

where $TREN and WORK are defined above.

[AKEL] = element stiffness matrix
[DI$PL] = displacements
[FTEL] = element applied load
CONOPS = 1.0 unless a different value is inputted
[PCOL] = grid point loads

3. Equations and Procedures, Continued:

[AF] = [AF] + [SIGELN]

If KR matrix was input, the contribution [KR][DISPL] must be added into the assembled force calculations. This contribution is not added into the element force calculations.

4. Input Arguments:

NMDB	- Order of assembled and reduced system
LOAD	- Load condition
PCOL	- External load columns
DISPL	- Displacement columns
CONOPS	- Load scalar array
LIST	- Array which defines boundary conditions for assembled system
K4	- Input tape which contains element matrices
IASM	- Control which indicates if assembly is to be done
NREF	- Number of reference points
NDIR	- Number of directions
NDEG	- Number of degrees of freedom
NAMIN	- Array containing input matrix names
IERROR	- Error flag
NAM3	- Output force matrix indicator
NSYS	- Total system degrees of freedom
NMDBO	- Number of bounded out points
AK	- Reduced stiffness matrix
TRAN	- Trasnformation matrix needed if KR matrix is present
W	- Working storage
NAMSCL	- Array containing input scalars
IW	- Working storage
INSPEC	- Array containing input matrix logical units

5. Output Arguments:

SIGEL	- Apparent element forces
SIGELN	- Net element forces
AKEL	- Element stiffness matrix
NODE	- Element node point numbers
PTEL	- Element applied load columns
LISTEL	- Array which maps element nodes to system nodes
AF	- Array which contains assembled element forces
NO3	- Total length of assembled force column

6. Error Return:

IERROR	- Non-zero if an error was detected in FARCIC

7. Calling Sequence:

```
Call FARCIC(NMDB,LOAD,PCOL,DISPL,CONOPS,SIGEL,SIGELN,AKEL,
            NODE,FTEL,LIST,LISTEL,IASM,AF,NREF,NDIR,NDEG,K4,
            NAMIN,IERROR,NO3,NAM3,NSYS,NMDBO,AK,TRAN,W,NAMSCL,
            IW,INSPEC)
```

8. Input Tapes:

   INSPEC (1,3)

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: $14F6_{16}$ bytes

12. Subroutine User:

    KALCON

13. Subroutines Required:

    UNITNM,INPTI1,EUTL3,EUTL9

14. Remarks: None

1. Subroutine Name: RECTIC

2. Purpose: Compute and print reactions for every grid point for one load condition. For non-bounded points, the values supply an "inverse" check.

3. Equations and Procedures:

Reactions = . sembly of element forces - load column procedure

(a) The load column for the load condition is read into PCOL and subtracted from R, the assembled forces.

(b) Reactions are printed for the load condition.

4. Input Arguments:

NDIR	- Number of directions
NDEG	- Number of degrees of freedom in system
LOAD	- Number of the loading conditions
NMDB	- Order of assembled and reduced system
NREF	- Number of reference points in system
PCOL	- External load columns
LIST	- Array which defines boundary condition, for assembled system
R	- Assembled element force matrix
IO4	- Output reaction matrix logical unit
NSYS	- Number of system degrees of freedom

5. Output Arguments: None

6. Error Returns: None

7. Calling Sequence:

```
Call RECTIC(NDIR,NDEG,LOAD,NMDB,NREF,PCOL,LIST,R,IO4,NSYS)
```

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: None

12. Subroutine User: KALCON

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: STRSIC

2. Purpose: Controls the calculation and printing of element stresses.

3. Equations and Procedures:

   (a) The element matrices a. e read from input tape IN.

   (b) For each element and each load condition, the element stresses are calculated and printed by subroutines STRCIC and STRPIC, respectively.

4. Input Arguments:

NAMI	- Name of output matrix stress matrix
NMDBO	- Number of bounded out points
DISPL	- Calculated displacements
LOAD	- Load condition under consideration
NMDB	- Order of assembled and reduced system
CONOPT	- Array containing load scalars
IN	- Unit number of input tape which contains element matrices

5. Output Arguments:

SIGEL	- Array containing apparent element stresses
SIGELN	- Array containing net element stresses
LISTEL	- Array which maps element nodes to system nodes
SEL	- Element stress matrix
NODE	- Array containing node points of an element
SZALEL	- Element thermal stress matrix
KRSEL	- Length of column record of output stress matrices

6. Error Returns: None

7. Calling Sequence:

```
Call STRSIC(NMDB,DISPL,CONOPT,SIGEL,SIGELN,LISTEL,SEL,NODE,
           SZALEL,NAMI,IN,LOAD,KRSEL,NMDBO)
```

8. Input Tapes:

   IN - Contains element matrices

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

    $BB2_{16}$ bytes

12. Subroutine User:

    KALCON

13. Subroutines Required:

    STRCIC, STRPIC, EUTL3

14. Remarks: None

1. Subroutine Name: STRCIC

2. Purpose: To calculate element stress resultants.

3. Equations and Procedures:

Calculates the stress array $IGEL and the corrected stress array STRESN for one load condition.

$$[SIGEL] = \sum [SEL]*[DISPL]$$

$$(STRESN] = [SIGEL] - CONOPT * [SZALEL]$$

where CONOPT = 1.0 unless a different value is inputted.

4. Input Arguments:

IEL	- Input element number
IPL	- Plug number
NMDB	- Number of reduced degrees of freedom
NLOAD	- Number of load conditions printed = 1
NRSEL	- Number of rows in element stress matrix
NORD	- Number of element degrees of freedom
NNO	- Number of element node points
NAME	- Name of output stress matrix
CONOPT	- Element load scalar
LISTEL	- Array which maps system degrees of freedom to element degrees of freedom
NODES	- Array containing element node point numbers
SEL	- Element stress matrix
SZALEL	- Thermal stress matrix
DISPL	- Displacement matrix
IERROR	- Error return
NLL	- Load condition number
NMDBO	- Number of bounded out points

5. Output Arguments:

SIGEL	- Array containing apparent element stresses
STRESN	- Array containing net element stresses

6. Error Returns: None

7. Calling Sequence:

```
Call STRCIC(IEL,IPL,NMDB,NLOAD,NRSEL,NORD,NNO,NAME,CONOPT,
           LISTEL,NODES,SEL,SZALEL,STRESN,DISPL,IERROR,NLL,
           NMDBO)
```

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

    510_{16} bytes

12. Subroutine User: STRSIC

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: STRPIC

2. Purpose: Prints out element stresses which were calculated in STRCIC.

3. Equations and Procedures:

Locates the stresses that are to be printed out and prints them with respect to their stress component designations; i.e., σ_{xx}, σ_{yy}, σ_{zz}, σ_{xy}, σ_{yz}, σ_{zx}.

Element number and node points are also printed.

4. Input Arguments:

IFMT	- Format control indicator
EZERO	- Suppression print option value
NRSEL	- Number of rows in the element stress matrix
KFMT	- Format control indicator
NSC	- Format control indicator
STRESS	- Stress matrix to be printed

5. Output Arguments: None

6. Error Returns: None

7. Calling Sequence:

```
Call STRPIC(IFMT,EZERO,NRSEL,KFMT,NSC,STRESS)
```

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

$8D0_{16}$ bytes

12. Subroutine User: STRSIC

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: GELS

2. Purpose: To solve a system of simultaneous linear equations with symmetric coefficient matrix upper triangular part of which is assumed to be stored column-wise.

3. Equations and Procedures:

   Solution obtained by Gauss Elimination with pivoting in main diagonal. (See writeup in IBM Scientific Subroutine Package.)

4. Input Arguments:

R	- MXN right hand side matrix (destroyed)
A	- Upper triangular part of symmetric MXM matrix
M	- Number of equations in the system
N	- Number of right hand side vectors
EPS	- Relative tolerance used to test loss of significance
AUX	- Auxiliary storage array

5. Output Arguments:

IER	- Error return
R	- Solution of equations

6. Error Returns:

IER = 0	No error
IER = -1	No result because of zero pivot element
IER = K	Warning - possible loss of significance at step K

7. Calling Sequence:

   Call GELS(R,A,M,N,EPS,IER,AUX)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: $(8C6)_{16}$ bytes

12. Subroutine User: DSPCIC

13. Subroutine Required: None

14. Remarks: None

1. Subroutine Name: MFSD

2. Purpose: Factor a given symmetric positive definite matrix.

3. Equations and Procedures:

   Solution obtained by Cholesky square root method. The given matrix is represented as the product of two triangular matrices, where the left hand factor is the transpose of the returned right hand factor. (See IBM Scientific Subroutine Package Writeup.)

4. Input Arguments:

A	- Upper triangular part of given symmetric matrix NxN
N	- Number of columns (rows) in given matrix
EPS	- Relative tolerance for test of loss of significance

5. Output Arguments:

IER	- Error return
A	- Resultant upper triangular matrix

6. Error Returns:

   IER = 0 No error
   IER = -1 No result because some radicand is non-positive
   IER = KK Warning indicating loss of significance

7. Calling Sequence:

   Call MFSD(A,N,EPS,IER)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: $(3FC)_{16}$ bytes

12. Subroutine User: DSPCIC

13. Subroutine Required: None

14. Remarks: The routine forms the first part of the LU decomposition solution of simultaneous equations.

1. Subroutine Name: MTDS

2. Purpose: Multiply a general matrix A on the left or right by (t inverse) ((T transpose) inverse) or (inverse (transpose (T*T)). The triangular matrix T is stored in the upper half by columns.

3. Equations and Procedures:

   See IBM Scientific Subroutine Package writeup.

4. Input Arguments:

   A - Given matrix (MxN)
   M - Number of rows in A
   N - Number of columns in A
   T - Given triangular matrix stored in upper half by columns
   IOP - Variable which controls operation to be performed by routine (see IBM writeup)

5. Output Arguments:

   IER - Error code

6. Error Returns:

   IER = -1 Invalid input value for IOP or M and N
   IER = 0 Successful operation
   IER = 1 Triangular matrix is singular

7. Calling Sequence:

   Call MTDS(A,M,N,T,IOP,IER)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: $(632)_{16}$ bytes

12. Subroutine User: DSPCIC

13. Subroutines Required: None

14. Remarks: Use MFSD first and IOP = 3 to do LU decomposition.

1. Subroutine Name: SINVIC

2. Purpose: Inversion of a bottom-half symmetric matrix.

3. Equations and Procedures:

The Inversion is performed by the method of partitioning.

4. Input Arguments:

IO	- Order of symmetric matrix to be inverted
A	- Symmetric matrix stored in column form
COL	- Work array to store a column of A

5. Output Arguments:

ISING	- Error Messages
A	- Inverted matrix
T1	- Time when inversion process started

6. Error Returns:

ISING = 0 No error
ISING = 1 Singular matrix
ISING = 2 Negative main diagonals

7. Calling Sequence:

```
Call SINVIC(IO,A,ISING,COL,T1)
```

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: $5E8_{16}$ bytes

12. Subroutine User: DSPCIC

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: MSBB

2. Purpose: To evaluate the matrix product of a symmetric bottom half matrix and a rectangular matrix and store result back in rectangular matrix.

3. Equations and Procedures:

$$AN_{nm} = \sum_{e} S_{ne} * B_{em}$$

$$B_{nm} = AN_{nm}$$

4. Input Arguments:

S	- Elements of [S] matrix (symmetric)
B	- Elements of [B] matrix
N	- Number of rows in the [S], [B] and [AN] matrices (order)
M	- Number of columns in the [B] and [AN] matrices (order)
N1	- Dimension of the [B] matrix
AN	- Auxiliary storage column of length N

5. Output Arguments:

B - Matrix product

6. Error Returns: None

7. Calling Sequence:

Call MSBB(S,B,AN,N,M1,N1)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: $(326)_{16}$ bytes

12. Subroutine User: DSPCIC

13. Subroutines Required: None

14. Remarks: Performs same function as routine MSB, but stores product back into the rectangular array B.

1. Subroutine Name: WCLOCK

2. Purpose: Generates the specific item being computed and the time.

3. Equations and Procedures:

```
NOUT = NOUT + 1
REPORT(1,NOUT) = ITEM
REPORT(2,NOUT) = RDCLOK(T)
```

4. Input Arguments:

NOUT	- Cumulative number of items being performed
REPORT	- Array containing time summary information
ITEM	- Number of calculations being performed

5. Output Arguments: None

6. Error Returns: None

7. Calling Sequence:

```
Call WCLOCK(NOUT,REPORT,ITEM)
```

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: $1B8_{16}$ bytes

12. Subroutine User: KALCON

13. Subroutines Required: RDCLOK

14. Remarks: None

1. Subroutine Name: RDCLOK (Function)

2. Purpose: To convert CPU and I/O time from milliseconds to minutes.

3. Equations and Procedures:

   Call Gettim (IX)

   (Total chargeable time = CPU TIME + I/O time)$_{\text{Milliseconds}}$

   RDCLOK = TOTAL CHARGEABLE TIME/60000

4. Input Arguments:

   T - Not used

5. Output Arguments:

   RDCLOK - Total chargeable time in minutes

6. Error Returns: None

7. Calling Sequence:

   Call Y = RDCLOK(T)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: 188_{16} bytes

12. Subrutine User: WCLOCK

13. Subroutines Required: GETTIM

14. Remarks: GETTIM is a locally written IBM/360 timer function.

1. Subroutine Name: TIMER

2. Purpose: Prints a time summary of the completed analysis.

3. Equations and Procedures:

   The routine scans over the NOUT entries in the report array treating each one consecutively. It looks at the two variables for each entry, the first giving the calculation code and the second the starting time. It then prints a summary of the items, their starting, ending and execution times.

4. Input Arguments:

   NOUT - Number of codes
   REPORT - Array containing codes and starting times

5. Output Arguments: None

6. Error REturns: None

7. Calling Sequence:

   Call TIMER(NOUT,REPORT)

8. Input Tapes: None

9. Output Tapes: NPOT

10. Scratch Tapes: None

11. Storage Required: None

12. Subroutine User: ANALIC

13. Subroutines Required: None

14. Remarks: None

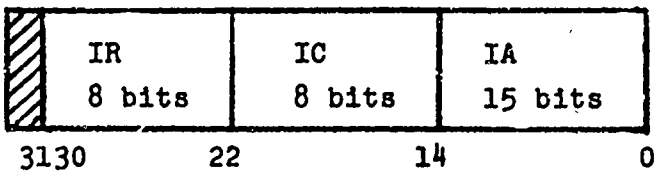
1. Subroutine Name: WAVEFM

2. Purpose: To generate the pointers and change the form of the stiffness matrix for the wavefront solution for displacements.

3. Equations and Procedures:

The three pointers which correspond to IR, IC and IA are stored in one word in the array IP. IR is the row number in the stiffness matrix. IC is the column which contains the first non-zero element in row IR. IA is the number of diagonal elements in row IR, in the IP array. All leading zero elements in each row of the stiffness matrix are eliminated. The maximum order of the system to be solved is 255, with no more than 32767 non (leading) zero elements. This limit is set by the word length of the 360.

FORMAT OF IP



4. Input Arguments:

AK - Reduced assembled stiffness matrix
NMDB - Order of AK

5. Output Arguments:

KK - Number of elements in the modified AK matrix (leading zeros eliminated)
IP - Array containing pointers

6. Error Returns: None

7. Calling Sequence:

Call WAVEFM(AK,KK,NMDB,IP)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: $(334)_{16}$ bytes

12. Subroutine User: DSPCIC

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: DECOMP

2. Purpose: Decompose a symmetric matrix for the Wavefront Solution for displacements.

3. Equations and Procedures:

Given:

$$\begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} = \begin{bmatrix} P_1 \\ P_2 \end{bmatrix}$$

decompose K_{11} as

$$K_{11} = L_{11} D^{-1} L_{11}^T$$

where L_{11} is lower triangular

D_{11} is diagonal with $d_{ii} = \ell_{ii}$

L_{11}^T is transpose of L_{11}

Elements of L_{11} are given by

$$\ell_{ij} = k_{ij} - \sum_{n=1}^{i-1} \frac{\ell_{nj}\ell_{ni}}{\ell_{nn}} \qquad (i \leq j)$$

4. Input Arguments:

KK	- Number of elements in symmetric matrix
N	- Order of matrix A
ERR	- Not used
A	- Symmetric matrix stored by rows with leading zero elements eliminated
A	- Auxiliary storage
IP	- Array containing pointers

5. Output Arguments: None

6. Error Returns: None

7. Calling Sequence:

   Call DECOMP(KK,N,ERR,A,Y,IP)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: $(894)_{16}$ bytes

12. Subroutine User: DSPCIC

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: FWD

2. Purpose: To do the forward decomposition step for the Wavefront Gauss solution.

3. Equations and Procedures:

Perform forward substitution to determine Y in

$$L_{11}Y = P_1 - K_{12}U_2$$

4. Input Arguments:

KK	- Number of elements in symmetric matrix A
N	- Order of matrix A
ERR	- Not used
A	- Symmetric matrix stored by rows with leading zero elements eliminated
DSP	- Load column
IP	- Array of pointers

5. Output Arguments: None

6. Error Returns: None

7. Calling Sequence:

```
Call FWD(KK,N,ERR,A,DSP,IP)
```

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: $(5FE)_{16}$ bytes

12. Subroutine User: DSPCIC

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: BACK

2. Purpose: To do the backward substitution step in the Wavefront Gauss solution for displacements.

3. Equations and Procedures:

   Perform backward substitution to determine U_1 from

$$Y = D^{-1}L_{11}^{T}U_1$$

4. Input Arguments:

KK	- Number of elements in symmetric matrix A
N	- Order of matrix A
ERR	- Not used
DSP	- Load column
A	- Stiffness matrix
IP	- Array of pointers

5. Output Arguments:

X	- Displacements

6. Error Returns: None

7. Calling Sequence:

   Call BACK(KK,N,ERR,DSP,A,X,IP)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: $(550)_{16}$ bytes

12. Subroutine User: DSPCIC

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: US04

2. Purpose: Control operation of the structural generative system (USER04 module)

3. Equations and Procedures: The error indicator, ERROR, is initially set to .FALSE.. Subroutine US04A is then called to control the input operations. Subroutine US04B is called to control the element matrix generation and output phases. If an error has occurred in the input phase then the call to US04B is skipped. All information received from the Format Monitor is relayed to US04A and US04B.

4. Input Arguments:

NUMOT:	Number of output matrices
NAMOUT:	Array containing output matrix names
IOSPEC:	Unit specifications for output matrices
NUMIN:	Number of input matrices
NAMIN:	Array containing input matrix names
INSPEC:	Unit specifications for input matrices
NUMSR:	Number of available scratch units
ISSPEC:	Scratch unit specifications
NUMSC:	Number of scalars
SCALAR:	Array containing scalars
NWORKR:	Number of available storages in blank common work area
WORK:	Work storage area
IPRINT:	System print control

5. Output Argument:

ERROR:	Error condition indicator

6. Error Returns: If error has occurred in US04A or US04B then ERROR will be .TRUE. upon return to the calling program.

7. Calling Sequence:

```
CALL US04 (NUMOT, NAMOUT, IOSPEC, NUMIN, NAMIN, INSPEC,
NUMSR, ISSPEC, NUMSC, SCALAR, ERROR, NWORKR, WORK, IPRINT)
```

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total storage required is $52C_{16}$ Bytes.

12. Subroutine User: SEXEQ

13. Subroutines Required:

USO4A
USO4B

14. Remarks: None

1. Subroutine Name: NTEST

2. Purpose: To determine if output matrix is to be generated by US04

3. Equations and Procedures: The first position in the output name is compared to a slash (/). If this first character is a slash then the matrix is not to be calculated. If the first character is not a slash then the matrix will be calculated and output.

4. Input Arguments: NAME - array containing output matrix name

5. Output Arguments: KØDE - control code
   if KØDE equals zero then matrix is calculated
   if KØDE equals one then matrix is not calculated

6. Error Returns: None

7. Calling Sequence: Call NTEST (NAME, KØDE)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is 156_{16} Bytes.

12. Subroutine User: US04A, US04B

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: REC1

2. Purpose: Write or read element input tape record

3. Equations and Procedures: The decision to write or read the tape record is determined by examing the input variable IOPT in the following manner:

   (1) If IOPT $\geq$ 2 then the tape record will be written
   (2) If IOPT $\leq$ 1 then the tape record will be read

4. Input Arguments: (when IOPT $\geq$ 2)

IOPT	: Read/write indicator
K	: Involved unit number
NI1	: Number of words in tape record, excluding NI1
IPL	: Element type number (plug number)
X	: "X" coordinates of element definition points
Y	: "Y" coordinates of element definition points
Z	: "Z" coordinates of element definition points
T	: Temperatures at element definition points
P	: Pressures at element definition points
NLIST	: Total degrees of freedom in element
LISTEL	: Boundary condition information list
NNO	: Number of element defining points
NODES	: Grid point numbers of element defining points
IP	: Extra element input and matrix repeat indicator
DISPEL	: Input displacements for element degrees of freedom
PCOLEL	: External loads for element degrees of freedom
LISTDL	: Not used
IG	: Maximum number of element defining points
NEL	: Element number
GPAXEL	: Grid point ax s transformation matrices for element defining points
NUMMAT	: Length of MAT array
MAT	: Array containing interpolated material properties
NUMEPS	: Length of EPSIO array
EPSIO	: Pre-strain load vector
NUMSO	: Length of SO array
SO	: Pre-stress load vector
EXTRA	: Extra element input

5. Output Arguments: (when IOPT $\leq$ 1)

   With the exception of IOPT and K, which are always input arguments, all of the above input arguments are output arguments when IOPT 1.

6. Error Returns: None

7. Calling Sequence: (IOPT, K, NIl, IPL, X, Y, Z, T, P, NLIST, LISTEL, NNO, NODES, IP, DISPEL, PCOLEL, LISTDL, IG, NEL, GPAXEL, NUMMAT, MAT, NUMEPS, EPSIO, NUMSO, SO, EXTRA)

8. Input Tapes: When IOPT ≤ 1 the input tape number is the variable K.

9. Output tapes: When IOPT ≥ 2 the output tape number is the variable K.

10. Scratch Tapes: None

11. Storage Required: Total storage required is $C14_{16}$ Bytes.

12. Subroutine User: ELEM, ELPLUG

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: LOGFLO

2. Purpose: Set logical execution controls for USER04 module

3. Equations and Procedures: APHASE, BPHASE and ERROR are initially set to .FALSE. All positions in MASTER are set to zero. If any of the first five output matrix positions are non-blank then APHASE is set to .TRUE. If any of the last seven output matrix positions is non-blank then BPHASE is set to .TRUE. MASTER is then filled by packing in the output matrix position number the requires that input section. At present there are six possible required input sections indicated in MASTER:

MASTER (1)	- System control input indicator
MASTER (2)	- Grid point coordinates input indicator
MASTER (3)	- Boundary condition input indicator
MASTER (4)	- Element definition input indicator
MASTER (5)	- Grid point loads input indicator
MASTER (6)	- Material library input indicator

4. Input Arguments:

NUMOT	: Number of output matrices
NAMOUT	: Array containing output matrix names
NUMIN	: Number of input matrices
NAMIN	: Array containing input matrix names
APHASE	: Logical variable indicating necessity to execute subroutine US04A
BPHASE	: Logical variable indicating necessity to execute subroutine US04B
NUMAST	: Length of MASTER
MASTER	: Array indicating required input sections

5. Output Arguments:

ERROR	: Logical variable indicating error condition

6. Error Returns: If output matrix position eleven is non-blank and input matrix position four is blank, then ERROR is set to .TRUE.

7. Calling Sequence:

(NUMOT, NAMOUT, NUMIN, NAMIN, APHASE, BPHASE, NUMAST, MASTER, ERROR)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total storage required is 786_{16} Bytes.

12. Subroutine User: USC4

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: USO4A

2. Purpose: Control input phase operations of structural system (USERO4 module)

3. Equations and Procedures: Input, output and scratch units supplied by the Format Monitor are assigned to their respective functions. Subroutine CONTRL is called to copy the entire structural data input onto a scratch tape, extracting structural system information in the process. From this point, subroutine INPUT controls the selection of all other subroutines which process input (see INPUT). The function of USO4A is to partition the blank common work storage area and select the proper subroutine for the following operations: If material library requests are present then subroutine FMAT is called, if report form input processing is required then subroutine REFORM is called, if generation of the loads matrix is not suppressed then subroutine FLOADS is called and finally if the broundary condition transformation matrix is not suppressed then FTR is called.

4. Input Arguments:

NUMOT:	Number of output matrices (12)
NAMOUT:	Array containing output matrix names
IOSPEC:	Unit specifications for output matrices
NUMIN:	Number of input matrices (4)
NAMIN:	Array containing input matrix names
INSPEC:	Unit specifications for input matrices
NUMSR:	Number of available scratch units
ISSPEC:	Scratch unit specifications
NUMSC:	Number of scalars (0)
SCALAR	Array containing scalars
NWORKR:	Number of available work storages in blank common area (WORK)
WORK:	Work storage area
IPRINT:	System print control
EPHASE:	Logical control variable indicating whether or not to go into element matrix generation
MASTER:	Array indicating required input sections
NUMAST:	Length of master array

5. Output Arguments:

ERROR:	Error condition indicator
KNMD:	Array containing structural system control information
KNUMD:	Array containing structural system control information
NUMK:	Length of KNUMD array

KNMD (1) - NSYS - Total number of degrees of freedom in application
KNMD (2) - NL - Number of load conditions
KNMD (3) - NMDB - Number of degrees of freedom after application of boundary conditions
KNMD (4) - NNORD- Summation of element degrees of freedom
KNMD (5) - NELEM- Number of elements
KNMD (6) - NNRSEL-Summation of element stress orders
KNMD (7) - NTD - Number of degrees of freedom per point
KNMD (8) - NRSELM-Maximum element stress order
KNMD (9) - NORDM- Maximum element degrees of freedom

5. Output Arguments: (Cont'd)

KNMD(10) - NOINRM-Maximum number of storages required for an element stiffness matrix
KNMD(11) - MAXNIl-Length of longest element record
KNMD(12) - NDIR - Number of directions per point
KNMD(13) - NDEG - Number of types of degrees of freedom
KNMD(14) - NMDBO- Number of zero boundary conditions

6. Error Returns: If at any time the number of required work storages exceeds NWORKR or a generated matrix will have a dimension greater than KONST (matrix size limitation), the appropriate message will be written, ERROR set to .TRUE. and control returned to the calling program.

7. Calling Sequence:

```
CALL, USO4A (NUMOT, NAMOUT, IOSPEC, NUMIN, NAMIN, INSPEC,
             NUMSR, ISSPEC, NUMSC, SCALAR, ERROR, NWORKR, WORK,
             IPRINT, KNMD, MASTER, NUMAST, NUMK, BPHASE)
```

8. Input Tapes:

1TAPE1 - INSPEC (1,1) - Unit containing input structure data deck
1TAPE2 - INSPEC (1,2) - Unit containing interpreted input
1TAPE3 - INSPEC (1,3) - Unit containing existing material library
1TAPE4 - INSPEC (1,4) - Unit containing input displacements

9. Output Tapes:

JTAPE1 - IOSPEC (1,1) - Unit which will contain copy of input structure data deck
JTAPE2 - IOSPEC (1,2) - Unit which will contain revised or new material library
JTAPE3 - IOSPEC (1,3) - Unit which will contain interpreted input
JTAPE4 - IOSPEC (1,4) - Unit which will contain grid point loads matrix
JTAPE5 - IOSPEC (1,5) - Unit which will contain boundary condition application transformation matrix
JTAPE6 - IOSPEC (1,6) - Unit which will contain assembly transformation matrix
JTAPE7 - IOSPEC (1,7) - Unit which will contain element stiffness matrices
JTAPE8 - IOSPEC (1,8) - Unit which will contain element load matrices
JTAPE9 - IOSPEC (1,9) - Unit which will contain element stress matrices
JTAP10 - IOSPEC (1,10) - Unit which will contain element thermal stress matrices
JTAP11 - IOSPEC (1,11) - Unit which will contain element incremental stiffness matrices
JTAP12 - IOSPEC (1,12) - Unit which will contain element mass matrices

10. Scratch Tapes:

NTAPE1 -	ISSPEC (1,1) -	External storage area for report form input preprocessor and later will contain structural control information
NTAPE2 -	ISSPEC (1,2) -	Contain temporary copy of translated input data deck and later contain generated element matrices in compact form
NTAPE3 -	ISSPEC (1,3) -	Contain temporary copy of actual input deck and later contain interpreted element input data
NTAPE4 -	ISSPEC (1,4) -	External storage area for report form input preprocessor and later contain input load conditions

11. Subroutine User: USO4

12. Subroutines Required:

CONTRL
INPUT
FMAT
REFORM
NTEST
FLOADS
FTR

13. Remarks: None

1. Subroutine Name: INDECK

2. Purpose: Translate input matrix containing a data deck into a BCD input deck

3. Equations and Procedures: The matrix is located by utilizing subroutine EUTL3. Each column of the matrix contains one input card divided into eighty rows. Each column is read in binary from the unit specified in INSPEC(1) and written on NØUT by an 80A1 format. The number of columns, as contained in the matrix header, is actually the number of cards in the data deck.

4. Input Arguments:

NAMIN	: Array containing input matrix name
INSPEC	: Array containing unit specification for input matrix
NOUT	: Logical unit reserved for output data deck
CARD	: Work storage

5. Output Arguments:

IER	: Logical variable indicating error condition

6. Error Returns: For each column of the input matrix, the compression code must be zero and the number of words must be eighty. If either condition is not satisfied then the matrix does not qualify as an input deck matrix and IER will be set to .TRUE..

7. Calling Sequence:

(NAMIN, INSPEC, NOUT, CARD, IER)

8. Input Tapes:

INSPEC(1) . unit containing input data deck matrix

9. Output Tapes:

NOUT : unit which will contain BCD data deck

10. Scratch Tapes: None

11. Storage Required: Total storage required is $56E_{16}$ Bytes.

12. Subroutine User: US04A

13. Subroutines Required: EUTL3

14. Remarks: None

1. Subroutine Name: CONTRL

2. Purpose: Generate BCD tape from system input tape data and read constants needed by US04 for dynamic storage and matrix sizes.

3. Procedure: The input data is read in BCD format of 12 words/card. A scanning of the data is made for certain card types.

   a. REPORT card - defines NBCD to be NTAPE3
   b. SYSTEM card - defines NBCD to be NTAPE2
   c. CHECK card - end of file of NBCD
   d. END card - end of file placed on NBCD
   e. SYSTEM card - NREF, NREFP, NTD, NL, NELEM are read to allocate storage

4. Input Arguments:
   NTAPE2 - tape storage number for defining NBCD
   NTAPE3 - tape storage number for defining NBCD
   NPIT - system input tape number

5. Output Arguments:

   NBCD : tape unit number on which data is stored
   NREF : number of reference points on system
   NREFP : number of reference points in grid point table
   NTD : number of degrees of freedom per point
   NL : number of grid point load conditions
   NELEM : number of elements

6. Error Returns: None

7. Calling Sequence: CALL CONTRL (NREF, NREFP, NTD, NL, NELEM, NTAPE2, NPIT, NBCD, NTAPE3)

8. Input Tapes: NPIT - Input data tape

9. Output Tapes: NBCD - Output BCD tape

10. Scratch Tapes: None

11. Storage Required: Total Storage required is $7DA_{16}$ Bytes.

12. Subroutine User: US04A

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: COPYDK

2. Purpose: Output a data deck in matrix form

3. Equations and Procedures: A matrix header is written in which the number of rows is set to eighty and the number of columns is set equal to the number of cards in the data deck. Each card of the data deck is read from NINPUT in 80A1 format and then written on the unit specified in IOSPEC(1) in a binary matrix column record containing eighty words. The process continues until an END, CHECK or $END card is encountered. Finally the matrix trailer is written and control is returned to the calling program.

4. Input Arguments:

NAMOUT	: Array containing output matrix name
IOSPEC	: Array containing unit specifications for the output matrix
CARD	: Work storage
NINPUT	: Unit containing data deck
JMAX	: Number of cards in data deck

5. Output Arguments: None

6. Error Returns: None

7. Calling Sequence:

(NAMOUT, IOSPEC, CARD, NINPUT, JMAX)

8. Input Tapes:

NINPUT : unit containing input data deck

9. Output Tapes:

IOSPEC(1): unit which will contain output data deck matrix

10. Scratch Tapes: None

11. Storage Required: Total storage required is $4C4_{16}$ Bytes.

12. Subroutine User: US04A

13. Subroutines Required:

EUTL5
EUTL6

14. Remarks: None

1. Subroutine Name: INPUT

2. Purpose: Process directly or control processing of all structural input data

3. Equations and Procedures: The input variable IN designates the Fortran logical unit number containing a direct label card input deck. If the input deck was actually direct it was copied onto IN by subroutine CONTRL. If report form input was used then the report form input preprocessor placed the generated direct label card input deck on IN.

   The logic in INPUT is to read a label card and branch to the appropriate section to process the indicated data. The available label sections and the action taken upon encountering each is indicated in the following list.

Input Section Label	Action Taken
TITLE	Title cards are read and printed on system output unit in INPUT
PRINT	No used, the data card is flushed through
NRL	Processed directly in INPUT, data eventually stored on scratch tape (NTAPE1)
GRID	Processed directly in INPUT, data eventually stored on scratch tape (NTAPE3)
BOUND	Processed by direct call to subroutine BOUND, data stored on scratch tapes (NTAPE1 and NTAPE3)
ELEM	Processed by direct call to subroutine ELEM, data stored on scratch tape (NTAPE3)
LOADS	Processed by direct call to subroutine FGRLDS data stored on scratch tape (NTAPE4)
END	Processed directly in INPUT, terminates input processing
TRANS	Processed directly in INPUT, data eventually stored on scratch tape (NTAPE3)
GRAXES	Processed by direct call to subroutine FRED, data eventually stored on scratch tape (NTAPE3)
MATER	Processed by setting input/output variable ITRACE equal to number of requests and returning to USO4A where ITRACE will be tested causing subroutine FMAT to be called; after the MATER section is processed USO4A will again call INPUT.

TZERO:	Processed directly in INPUT,eventually stored on scratch tape (NTAPE3)
CHECK:	Processed directly in INPUT, terminates input processing for a case but does not execute data
REPORT:	Processed by setting input/output variable IN to the value of NTAPE2 and returning to US04A where IN will be tested causing subroutine REFORM to be called; after report form input processing is completed US04A will again call INPUT.
SYSTEM:	Processed directly in INPUT
ELPR 3000	Element pressures processed directly in INPUT, data eventually stored on scratch tape (NTAPE3)
ELTE 4000	Element temperatures processed directly in INPUT, data eventually stored on scratch tape (NTAPE3)
HARM:	Processed by direct call to subroutine HARGEN
TEM:	Processed by direct call to subroutine HARGEN
SDC:	Processed directly in INPUT, data eventually used by subroutine HARGEN

4. Input Arguments:

NTAPE1:	Scratch unit number
NTAPE2:	Scratch unit number
NTAPE3:	Scratch unit number
NTAPE4:	Scratch unit number
ITAPE1:	Existing material library unit number
JTAPE1:	Revised or new material library unit number
NREFP1:	Not used
NSYS:	Total degrees of freedom in application (adjustable dimension)
IN:	Data deck unit number
IPRINT:	System print control
NPIT1:	Scratch input control for report form input
ITRACE:	Material library residence indicator
NAMIN:	Existing material library matrix name
INSPEC:	Existing material library unit number
NAMOUT:	Revised or new material library name
IOSPEC:	Revised or new material library unit number
NRF:	Number of total reference points in application (must be equal to highest point number)
X,Y,Z:	Storage allocated for coordinate data
T:	Storage allocated for grid point temperatures
P:	Storage allocated for grid point pressures
TGRA:	Storage allocated for grid point axes transformation matrices
IZR:	Not used
LIST:	Storage allocated for boundary conditions
DISPL:	Storage allocated for input displacements
LNOD:	Not used
NZEL:	Not used

PCOL: Storage allocated for grid point loads
ELPRM: Array of element pressure modal values
ELTPM: Array of element temperature modal values
NUMIN: Number of input matrices
NUMOT: Number of output matrices
WORK: Work storage
NWORKR: Maximum work storage available
NIN: Number of work storages used
IHONT: Array containing harmonic data

5. Output Arguments:

ICALC: Execution indicator

if END Card read, ICALC is set to 1 and USO4A will relinquish control to USO4B for matrix generation

if CHECK card read, ICALC is set to zero and subroutine USO4A will set controls to return to Format Monitor (execution of data is suppressed)

ITRACE: Material request indicator

if ITRACE is not equal to zero upon exit from INPUT then USO4A will call FMAT

IN: Report form input preprocessor indicator

if IN is equal to NTAPE2 upon exit from INPUT then USO4A will call REFORM

ICONT: Array indicating processed input sections

6. Error Returns: If any errors are detected then INPUT will set ERROR to .TRUE. and return.

7. Calling Sequence:

CALL INPUT (X, Y, Z, T, P, TGRA, IZR, LIST, DISPL, LNOD, NZEL, PCOL, ITRACE, ICALC, NTAPE1, NTAPE2, NTAPE3, NTAPE4, ITAPE1, JTAPE1, NREFP1, NSYS, IN, IPRINT, NMD, NPIT1, ERROR, NAMIN, INSPEC, NAMOUT, IOSPEC, NRF, NUMIN, NUMOT, ICONT, ELPRM, ELTPM, WORK, NWORKR, NIN, IHONT)

8. Input Tape:

ITAPE1 - Contains existing material library
JTAPE1 - Contains revised or new material library

9. Output Tapes: None

10. Scratch Tapes

NTAPE1 - Temporary storage for structure control information including system orders, boundary conditions and system print operations

NTAPE2 - Scratch unit used when rewriting NTAPE3 for grid point axes data storage

NTAPE3 - Storage for interpreted element input

NTAPE4 - Storage for input grid point load conditions

11. Subroutine User: US04A

12. Subroutines Required:

BOUND
ELEM
FGRLDS
FRED
REC1
HARGEN

13. Remarks: None

1. Subroutine Name: FRED

2. Purpose: To compute transformation matrices when input for GRAXES is encountered.

3. Equations and Procedures:

$$\begin{Bmatrix} u^1 \\ v^1 \\ w^1 \end{Bmatrix} = [T] \begin{Bmatrix} u \\ v \\ w \end{Bmatrix}$$

where (1) u, v, w, are the displacements in the global x, y, z system

(2) u', v', w' are the displacements in the new x', y', z' system

(3) $[T]$ contains the direction cosines

4. Input Arguments:

X :X coordinates of plane defined by 3 pts.
Y :Y coordinates of plane defined by 3 pts.
Z :Z coordinates of plane defined by 3 pts.
KID :See Remarks
L :Point 1 of Plane
M :Point 2 of Plane
N :Point 3 of Plane

5. Output Arguments: TRANSC - transformation matrix $[T]$

6. Error Returns:

(1) If points 1 and 2 have same coordinates, no plane defined.
(2) If point 3 lies on the line connecting points 1 and 2, there is no plane defined.

7. Calling Sequence:

CALL FRED (X, Y, Z, TRANSC, KID, L, M, N)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: X(1), Y(1), Z(1), TRANSC (3,3)

12. Subroutine User: INPUT

13. Subroutines Required: None

14. Remarks:

1. Since 3 points define a plane, KID may be

   (a) 0 when the 1st 2 points define the x' axis
   (b) 1 when the 1st 2 points define the y' axis
   (c) 2 when the 1st 2 points define the z' axis

The direction cosines are first computed for points 1 and 2 defining the x' axis. If KID is $\neq$ 0, then the direction cosines are rearranged to give the respective notation described above.

2. In spite of error returns indicated, analysis does not terminate.

1. Subroutine Name: BOUND

2. Purpose: Read and process boundary condition data and input displacement data

3. Equations and Procedures: The boundary conditions are read for each point input and the data is stored in the array LIST to be later written on scratch tape NTAPE1 by subroutine INPUT. Omitted points are constrained for all degrees of freedom. Only unconstrained degrees of freedom are stored in LIST, giving LIST a length equal to the actual degrees of freedom for which solution will be obtained (NMDB). For each degree of freedom for which a solution is desired, its appropriate total system degree of freedom location, which is NTD*(IN-1)+L, where NTD is the number of degrees of freedom per point, IN is the point number and L is the subject degree of freedom for that point number, is placed in the next available position in LIST. The same procedure is followed for input displacements, which are stored in DISPL.

4. Input Arguments:

IVEC	-	Not used
NDIR,NDEG	-	Product equals NTD, number of degrees of freedom per point
NREF	-	Total number of points referenced in application
NREF4	-	Number of points for which boundary conditions have been input
IN	-	Input unit containing boundary condition data
NSYS	-	Total number of degrees of freedom in application

5. Output Arguments:

NMDB	-	Number of degrees of freedom for which solutions are desired
NMDB2	-	Number of degrees of freedom for which displacements have been input
LIST	-	Array containing degree of freedom numbers for which solutions are to be obtained and displacements have been input
DISPL	-	Array containing input displacements

6. Error Returns: None

7. Calling Sequence:

CALL BOUND (IVEC, NDIR, NDEG, NREF, NMDB, NMDB2, LIST, DISPL, NREF4, IN, NSYS)

8. Input Tape:

IN - Unit containing boundary condition and input displacement data

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total storage required is $7FA_{16}$ Bytes.

12. Subroutine User: INPUT

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: ELEM

2. Purpose: Process element input data (input section ELEM)

3. Equations and Procedures: Processing of element input data begins by reading the element definition input for an element and checking the values for errors and inconsistencies. Error messages for subroutine ELEM are exhibited in Appendix III. The information read is then printed on the system output unit. If no errors have been detected then the element definition input is merged with the required system input. Specifically, the following operations are performed for each element to assimilate the required information for generation of element matrices:

   the coordinates, temperatures and pressures are extracted and stored for each of the element definition node points;

   the grid point axes transformation matrices are initialized as identity matrices and stored for each of the element definition node points;

   the interpolation temperature for material properties is read or calculated dependent upon input, the material library is searched to locate the requested material, the interpolation is performed and the results stored;

   the element generation print control is stored;

   the boundary conditions for the degrees of freedom referenced by the element defining node points are extracted from the system boundary condition list and stored;

   the input displacements, if any, for the degrees of freedom referenced by the element defining node points are extracted from the system input displacement list and stored;

   the pre-strains and pre-stresses, if input, are read and stored;

   the extra element input data, if any, is read and stored and finally, subroutine REC1 is called to place all of the above interpretated element data on scratch tape NTAPE3 (see REC1).

4. Input Arguments:

NELEM:	Number of elements
X,Y,Z:	Arrays containing coordinates of system grid points
T,P:	Arrays containing temperatures and pressures respectively for system grid points
IVEC:	Not used
LIST:	Array containing boundary condition information for system grid points
NMDB2:	Number of entries in array LIST
NDIR,NDEG:	Product equals number of degrees of freedom per grid point
IG:	Maximum number of element defining points possible for an element
NMDB:	Number of system degrees of freedom for which solutions are desired
DISPL:	Array containing input displacements
LNGD:	Not used
GPAXEL:	Work storage reserved for grid point axes transformation matrices
NUTAPE:	Logical variable indicating that new or revised material library has been generated
TZERO:	Base temperature for application
NUMSEQ:	Material library sequence number
XEL,YEL, ZEL:	Work storage reserved for extracting coordinates for element definition node points
TEL,PEL:	Work storage reserved for extracting temperatures and pressures for element definition node points
LISTEL:	Work storage for extracting boundary condition information for element definition node points
NODES:	Array containing element definition node point numbers
DISPEL:	Work storage reserved for extracting input displacements for element definition node points
PCOLEL:	Not used
MAT:	Work storage reserved for interpolated material properties, element print control, mass density and TZERO
EPSIO:	Work storage reserved for pre-strain load vector
SO:	Work storage reserved for pre-stress load vector
EXTRA:	Work storage area reserved for extra element input
IN:	Element data input unit number
NREFP:	Number of input system grid points
ITAPE1:	Existing material library unit number
JTAPE:	Not used

JTAPE1: New or revised material library unit number
NTAPE3: Scratch unit number
NAMIN: Name of existing material library
INSPEC: Same as ITAPE1
NAMOUT: Name of new or revised material library
IOSPEC: Same as JTAPE1

5. Output Arguments:

IFLAG: Error indicator
NNORD: Summation of element degrees of freedom
NNRSEL: Summation of element stress orders
NORDM: Maximum element degrees of freedom for this application
NOINKM: Maximum number of storages for element stiffness matrix for this application
NRSELM: Maximum element stress order for this application

6. Error Returns: If an error is encountered then IFLAG is set to minus one and control is returned to the calling program.

7. Calling Sequence:

CALL ELEM (NELEM, X, Y, Z, T, P, IVEC, LIST, NMDB2, NDIR, NDEG, IG, NMDB, DISPL, LNOD, GPAXEL, NUTAPE, TZERO, IFLAG, NUMSEQ, XEL, YEL. ZEL, TEL, PEL, LISTEL, NODES, DISPEL, PCOLEL, MAT, EPSIO, SO, EXTRA, IN, NREFP, NNORD, NNRSEL, NORDM, NOINKM, NRSELM, ITAPE1, JTAPE, JTAPE1, NTAPE3, NAMIN, INSPEC, NAMOUT, IOSPEC)

8. Input Tape:

IN - Contains element input data

9. Output Tapes:

NTAPE3 - Contains interpreted element input

10. Scratch Tapes: None

11. Storage Required: Total storage required is $4D78_{16}$ Bytes.

12. Subroutine User: INPUT

13. Subroutines Required:

MATCH
EUTL3
LAG
REC1

14. Remarks: In calculating the interpolated material properties, if the requested material and the interpolation temperature of the present element being processed are the same as the previous element then the results calculated for the previous element are used and no searching or interpolation is done; if the requested material is in core but the interpolation temperature is different then just the searching is eliminated.

1. Subroutine Name: MATCH

2. Purpose: Compare a material number and its interpolation temperature to the material number and interpolation temperature last referenced in order to determine if a search of the material library tape and/or interpolation is necessary.

3. Equations and Procedures: The material number, TAG1, is compared to the material number now residing in core, NSAVE1. If they do not match, then they are tested again to see if they differ only by an asterisk in the first position. If they still do not match then control is returned to the calling program at the statement following the CALL MATCH statement. If a match was obtained while testing for an asterisk then STAR is set to TRUE. Once a match has been obtained for the material number, the following procedure is followed:

   If IELEM equals one then control is returned to the statement number replacing the first asterisk since interpolation must be done for the first element. If IELEM is not one then a check is made to see if a search of the material library was in progress to find this material number. If this is the case then control is returned to the calling program at the statement number replacing the first asterisk since this material table has just been place in core and interpolation will be necessary. If a search was not in progress then TEMP is compared to SAVTEM. If they are equal then interpolation of the material table has already been calculated and control is returned to the calling program at the statement number replacing the second asterisk. If TEMP does not equal SAVTEM then control returns through the first asterisk in order to perform the interpolation.

4. Input Arguments:

TAG1	: Material number desired
NSAVE1	: Material number now residing in core
TEMP	: Interpolation temperature desired
SAVTEM	: Last interpolation temperature processed
NDIFF	: Constant used to determine if asterisk is present in material number
IELEM	: Element number
SEARCH	: Logical variable indicating if a search of the material library is in progress
,	: Non-standard returns to calling program (See 7. Calling Sequence)

5. Output Arguments:

   STAR : Logical variable indicating presence of asterisk in material number.

6. Error Returns: None

7. Calling Sequence: CALL MATCH (TAG1, NSAVE1, TEMP, SAVTEM, NDIFF, STAR, IELEM, SEARCH, *,*)

   Where the asterisks are statement numbers, preceded by a dollar sign ($), that MATCH will return control to in the calling program. Control will pass to the statement number replacing the first asterisk if TAG1 matches NSAVE1 but TEMP does not match SAVTEM (i.e. the material is the same but the interpolation temperatures differ). Control will pass to the statement number replacing the second asterisk if TAG1 matches NSAVE1 and TEMP matches SAVTEM (i.e. the material is the same as the last material referenced and the interpolation temperatures are also the same). If TAG1 does not match NSAVE1 then control is returned to the calling program at the statement following the CALL MATCH statement.

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is $2FE_{16}$ Bytes.

12. Subroutine User: ELEM

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: LAG

2. Purpose: Linear interpolation routine for material properties

3. Equations and Procedures:

$$ZAPX = \frac{X(I)\,Y(I-1) - X(I-1)\,Y(I) + P(Y(I) - Y(I-1))}{X(I) - X(I-1)}$$

4. Input: P - temperature at which material properties will be interpolated
   K - number of pairs of coordinates
   X - X coordinate
   Y - Y coordinate

5. Output; ZAPX - value of the material property being interpolated

6. Error Returns: None

7. Calling Sequence: CALL LAG (P, ZAPX, K, X, Y)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage: Total Storage required is $2F2_{16}$ Bytes.

12. Subroutine User: ELEM

13. Subroutines Required: None

14. Remarks: If there is only one X-Y pair, ZAPX will be set equal to Y.

1. Subroutine Name: FGRLDS

2. Purpose: Read and print grid point loads data

3. Equations and Procedures: System input is read from NTAPE4 and includes LIST which is an array containing row numbers of degrees of freedom which are to be retained in the reduced load column. Grid point loads are read for each input point and printed. If grid point axis transformations are present, this transformation is applied. The assembled PCØL is stored on tape NTAPE4, followed by the reduced PCØL. This process is repeated for each load condition.

4. Input Arguments:

NL :Number of grid point load conditions
TGRA :Grid point axes transformation matrices
NØGPA :Number of grid point axes transformations
LIST :Reduction array
IT :System input tape number
NTAPE1 :Input tape number
NTAPE4 :Output tape number

5. Output Arguments: PCØL - Loads Column

6. Error Returns: None

7. Calling Sequence:

CALL FGRLDS (NL, TGRA, NØGPA, LIST, IT, PCØL, NTAPE1, NTAPE4, NSYS)

8. Input Tapes: NTAPE1: Record 1 - not used

Record 2 - NMDB1, NMDB, LIST

9. Output Tapes: NTAPE4: Record 1 - N1, NMDB1, NMDB

Record 2 - PCOL (assembled)

Record 3 - PCOL (reduced)

Repeat Record 2 and 3 for each load condition

10. Scratch Tapes: None

11. Storage Required:

LIST (NSYS)
ELØAD (12)
PCOL (NSYS)
COL (3)
ISAVE (3)
TGRA (3, 3, NREFP)

12. Subroutine User: INPUT

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: FMAT

2. Purpose:

   (a) Generate material library tape
   (b) Update material library tape
   (c) Print material library information

3. Equations and Procedures:

   Subroutine FMAT operates in three distinct phases.

   First, a test is made on NM. If NM is positive, then it assumed that this is an update run and the original material library is read into PRØPER from ITAPE1. Each table in the library is placed in PRØPER in a block of length NTØT, where NTØT is computed as the necessary storage needed. If NM is zero, it is considered an error condition and a message is printed and control is returned to the calling program. If NM is negative, then it is assumed that this is a generation of a new material library tape and the section which reads the original material library tape is skipped.

   The second phase consists of processing the requests. The requests are controlled by an input code read into location D. The legal input codes are:

   (1) I : add or revise isotropic material
   (2) Ø : add or revise orthotropic material table
   (3) PI : add or revise plastic isotropic material table
   (4) PØ : add or revise plastic orthotropic material table
   (5) P : add plastic section to existing material table
   (6) ØUT : delete material table with correct lock code
   (7) ALL : print entire material library
   (8) ØEE : print material table
   (9) ØUM : print summary of material library
   (10) /*/ : print lock code for material table
   (11) ZAP : delete material table regardless of lock code

If NM was negative, then the only allowable codes are I,Ø,PI and PØ and the requests are processed and placed into the array PRØPER starting from the beginning and ending at NTØL. If NM was positive, then the material number is checked against the materials in PROPER to see if it already existed in the original library. If no match is obtained, then the material is added at the next open block in PRØPER and NTØL is updated accordingly. If a match occurred, then the revised table will be placed in same position as the original table. If the locations for the material is greater or lesser than before, the remaining contents of PROPER, i.e. those tables after the one in question, are shifted down or up respectively. If the request is of the type that will alter or delete the original table, then the lock code (TAG2) must match the lock code of the original table, otherwise an error condition is encountered and control returns to the calling program. Once it has been decided where the table is to be placed, then the table is read into PRØPER by material temperature points and plastic temperature points. The material properties are as follows:

E - Young's Modulus
ν - Poisson's Ratio
α - Coefficients of Thermal Expansion
G - Rigidity Modulus

For an input code of I or PI only E, ν , and α are read and G is computed from $E/2(1+\nu)$ for each material temperature point. For an input code of Ø or PØ, then E_x, E_y, E_z, ν_{xy}, ν_{yz}, ν_{zx}, α_x, α_y, α_z, G_{xy}, G_{yz} and G_{zx} are read for each material temperature point. If the input code contains a P, then for each plastic temperature point the following data is read:

N - exponent of stress-strain function assumption
K - scalar of stress-strain function assumption
X - nondimensionalizing factor for Van Mises yield criteria
Y - " "
Z - " "
R - " "
S - " "
T - " "

The procedure for input codes 6-11 is as follows:

ØUT - If the material is not located in PRØPER, then a message is printed to this effect and the request is ignored. If the material is located and the lock codes do not match, then a message is printed and the request ignored. If the material is located and the lock codes match, then the deletion occurs when the remaining contents of PRØPER are merely shifted up over the deleted material.

ALL - A flag (WRTALL) is set for phase three and control passes to the next request.

SEE - If the material is not located, a message is printed and the request ignored. If the material is located, the table is printed and control passes to the next request.

SUM - All the tables in PRØPER are scanned and the following information is printed for each table:

Material Number (TAG1)
Material Identification (MIDENT)
Analysis Capability (derived from I,Ø, PI,PØ)
Number of Material Temperature Points (NP1)
Number of Plastic Temperature Points (NP2)
Temperature Range of Material Table
Temperature Range of Plastic Table

/*/ - If the material is located, the lock code is printed. If the material is not located, the request is ignored.

ZAP - If the material is not located, the request is ignored. If the material is located, it is deleted regardless of lock code.

Phase two ends when all of the requests have been processed.

Phase three consists of writing the new or updated material library on JTAPE1 and printing the entire tape if it has been requested. Writing of the tape and a print of the entire material library, if requested, are done in a parallel processing manner; i.e., a table is written on tape and then printed, if requested. Either process may be done separately or together depending upon the requests received. Finally, if a tape has been written, a summary is printed.

4. Input Arguments:

NM	: Number of Requests
MATTAP	: Code Controlling Selection of Input and Output Tapes
IN	: Input Tape Unit
TABMAT	: Material Properties Work Storage Area
TABPLA	: Plastic Properties Work Storage Area
PRØPER	: Material Library Work Storage Area
NWØRK	: Number of Available Work Storages
ITAPE1	: Input Material Library Tape Unit
JTAPE1	: Output Material Library Tape Unit
NAMØUT	: Array Containing Output Material Library Name
NAMIN	: Array Containing Input Material Library Name

5. Output Arguments:

MATTAP	: Code signifying error condition has been encountered, if MATTAP $\geq$ 0, then no error has been encountered, if MATTAP $<$ 0, then error condition exists.

6. Error Returns:

	Message	Action Taken
(1)	Value of Young's Modulus (E) $\leq$ 1.0	RETURN
(2)	Value of Poisson's Ratio $<$ -1.0 or $>$ 1.0	RETURN
(3)	Value of thermal expansion coefficient (α) $<$ -1.0 or $>$ 1.0	RETURN
(4)	Value of Rigidity Modulus (G) $\leq$ 1.0	RETURN
(5)	Value of mass density is negative	RETURN
(6)	Lock codes do not match for revision	RETURN
(7)	Lock codes do not match for deletion	IGNORE REQUEST
(8)	Capacity of material library exceeded	RETURN
(9)	Number of material or plastic temperature points $>$ 9	RETURN
(10)	Attempt to delete nonexistent material	IGNORE REQUEST
(11)	Attempt to input plastic data only for nonexistent material	IGNORE REQUEST
(12)	Unrecognizable input code	RETURN
(13)	Request to print nonexistent material	IGNORE REQUEST
(14)	Number of requests is zero	RETURN

7. Calling Sequence:

```
Call FMAT (NM, MATTAP, IN, TABMAT, TABPLA, PRØPER, NWØRK,
           ITAPE1, JTAPE1, NAMØUT, NAMIN)
```

8. Input Tapes
9. Output Tapes

Input and output tapes are identical with respect to information contained and record format. Records are as follows from the matrix header to the matrix trailer:

Format Matrix Header Record

Record number 1 - ICØL, KØDE, IWØRDS, NUMTAB, NUMSEQ

Record numbers 2 to NUMTAB+1 - ICØL, KØDE, IWØRDS, NTØT, D, TAG1, TAG2, NP1, NP2, DENSTY, MIDENT, ((TABMAT (I,J), J=1, NMAT), I=1, NP1), ((TABPLA(I,J), J=1, NPLA), I=1,NP2)

Format Matrix Trailer Record

where
- ICOL : Dummy Variable
- KØDE : Dummy Variable
- IWØRDS : Number of Words Remaining in Record
- NUMTAB : Number of Material Tables in Library
- NUMSEQ : Sequence Number of Library
- NTØT : Total Number of Words in the Specific Table
- D : Input Code
- TAG1 : Material Number
- TAG2 : Lock Code
- NP1 : Number of Material Temperature Points
- NP2 : Number of Plastic Temperature Points
- DENSTY : Mass Density
- MIDENT : Material Identification (Short Description or Name)
- TABMAT : Material Properties Table
- NMAT : Number of Material Properties per Temperature Point + 1
- TABPLA : Plastic Properties Table
- NPLA : Number of Plastic Properties per Temperature Point + 1

10. Scratch Tapes: None

11. Storage Required:

    CØM(10), MIDENT(4), G(16), LEADER(20), TAG1(6), NFIX1A(6), FL1A(6)
    Total Storage required is $4EB8_{16}$ Bytes.

12. Subroutine User: USØ4A

13. Subroutines Required: SHIFT

14. Remarks:

    Whenever new or updated material tape is written, all changes and/or additions and a summary of the output tape are printed.

1. Subroutine Name: SHIFT

2. Purpose: Given a one-dimensional array, this routine can relocate a block of data, within the array.

3. Equations and Procedures: The routine computes the size of the block to be shifted. It checks the direction of shift, and initializes the shift constants, finally performing the shift.

4. Input Arguments:

PROPER	: Array in which shifting is to occur
IFROM	: Initial subscript of block to be shifted
ITO	: Final subscript of block to be shifted
ISIZE	: Size of shift
NDIR	: Direction of shift

5. Output Arguments:

IERROR : Error return

6. Error Returns: If the size of the block to be shifted is computed to be negative (IFROM . ITO) IERROR is set equal to 1 (one).

7. Calling Sequence:

SHIFT (PROPER, IFROM, ITO, ISIZE, NDIR, IERROR)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total storage required is $2B6_{16}$ Bytes.

12. Subroutine User: FMAT

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: HARGEN

2. Purpose:

Compute the working storages which are required to form harmonic numbers and harmonic coefficients.

3. Equations and Procedures:

(a) Read the number of element from the tape NTAPE3 and number of harmonic required from the tape NPIT.

(b) (b) Compute the working storages required.

(c) Call subroutine HARGE1

4. Input Arguments:

XEL:	used by REC1
YEL:	used by REC1
ZEL:	used by REC1
TEL:	used by REC1
PEL:	used by REC1
NORD:	used by REC1
LISTEL:	used by REC1
NNO:	used by REC1
NODES:	used by REC1
IP:	used by REC1
DISPEL:	used by REC1
PCOLEL:	used by REC1
LISTDL:	used by REC1
IG:	used by REC1
NEL:	used by REC1
GPAXEL:	used by REC1
NUMMAT:	used by REC1
MAT:	used by REC1
NUMEPS:	used by REC1
EPSION:	used by REC1
NUMSO:	used by REC1
SO:	used by REC1
EXTRA:	used by REC1
WORK:	working storage array
NWORK:	the number of words available for the working storage
NIN:	number of the working storage allocated
IHONT:	used by HARGE1
NTAPE3:	input tape
NTAPE4:	used by HARGE1
IN:	input tape
HPTC:	used by HARGE1
HARP:	used by HARGE1
HART:	used by HARGE1
ERROR:	error return code

5. Output Arguments:

   If error = TRUE on return. It means it does not have enough core storage to compute the harmonic number and the harmonic coefficient.

6. Calling Sequence:

   (XEL,YEL,ZEL,TEL,PEL,NORD,LISTEL,NNO,NODES,IP,DISPEL,PCOLEL,LISTDL,
   IG,NEL,GPAXEL,NUMMAT,MAT,NUMEPS,EPSION,NUMSO,SO,EXTRA,WORK,NWORKR,
   NIN,IHONT,NTAPE3,NTAPE4,IN,HPTC,HARP,HART,ERROR)

7. Input Tapes: NTAPE3, IN

8. Output Tapes: None

9. Scratch Tapes: None

10. Subroutine User: Input

11. Subroutine Required: HARGE1

1. Subroutine Name: HARMA

2. Purpose: Subroutine HARMA is used to compute the harmonic number and the harmonic coefficient from fourier series.

3. Equations and Procedures:

The general formula for fourier series is

$$F(X) = \frac{Ao}{2} + \sum_{N=1} \left(A_n \text{COS} \frac{N\pi}{L} X + B_n \text{SIN} \frac{N\pi X}{L}\right), \text{ Where}$$

$$A_n = \frac{1}{L} \sum_{-L}^{L} (+(X) \text{ COS } \frac{N \pi X}{L}) \quad N = 0, 1, 2 \qquad (1)$$

$$B_n = \frac{1}{L} \sum_{-L}^{L} (+ (X) \text{ SIN } \frac{N \pi X}{L}), N = 1, 2 \qquad (2)$$

For a given value of +(X) and N, the harmonic coefficients An and Bn will be computed by using formula (1) and (2). The harmonic number is formed by taking the average value between -L and L.

4. Input Arguments:

Y - Y array contains pressure loads or thermal loads
Fc - On return array Fc contain the fourier coefficients
KH - Fourier number
NPT - Number of the pressure loads or the thermal loads
C - COS value
S - SIN value
KMAX - Maximum fourier number to be generated
CONV - Array conv contains the convergent factors

5. Output Arguments:

Fc - Array Fc contain the fourier coefficients
KH - Fourier number

6. Error Returns: None

7. Calling Sequence:

(Y,A,KHN,NH,NPTC,C,S,KMAX,CON)

1. Subroutine Name: HARGE 1

2. Purpose: Read input data for subroutine HARMA. Output the harmonic numbers and the harmonic coefficients.

3. Equations and Procedures:

   (a) Retrieve the number of elements from the input tape
   (b) Read the necessary harmonic generated input data
   (c) Call subroutine HARMA to generate the harmonic number and the harmonic coefficients for each element in this analysis.
   (d) Add the harmonic number and the harmonic coefficients onto Ntape 3. Ntape 3 contains element informations.

4. Input Arguments:

XEL	-Used by REC1
YEL	-Used by REC1
ZEL	-Used by REC1
TEL	-Used by REC1
PEL	-Used by REC1
NORD	-Used by REC1
LISTEL	-Used by REC1
NNO	-Used by REC1
NODES	-Used by REC1
IP	-Used by REC1
DISPEL	-Used by REC1
PCOLEL	-Used by REC1
LISTDL	-Used by REC1
IG	-Used by REC1
NEL	-Used by REC1
GPAXEL	-Used by REC1
NUMMAT	-Used by REC1
MAT	-Used by REC1
NUMEPS	-Used by REC1
EPSION	-Used by REC1
NUMSO	-Used by REC1
SO	-Used by REC1
EXTRA	-Used by REC1
Work	-Work Storage Array
NWOKR	-Number of work storages
NIN	-Number of work storage allocated
IHONT	-Harmonic information array
NTAPE3	-Input tape contains element informations
NTAPE4	-Scratch tape
IN	-Card input unit
HPTC	-Control for the pressure load or the thermal load
HARP	-Control for the pressure load
HART	-Control for the thermal load

5. Output Arguments:

   IHONT - This array contains the harmonic informations

6. Error Returns: None

7. Calling Sequence:

   (XEL, YEL, ZEL, TEL, PEL, NORD, LISTEL, NNO, NODES, IP, DISPEL, PCOLEL, LISTDL, IG, NEL, GPAXEL, NUMMAL, MAT, NUMEPS, EPSION, NUMSO, SO, EXTRA, WORK, NWOKR, NIN, IHONT, NTAPE3, NTAPE4, IN, HPTC, HARP, HART)

8. Input Tapes: NTAPE3

9. Output Tapes: NTAPE3

10. Scratch Tapes: NTAPE4

11. Storage Required:

12. Subroutine User: Input

13. Subroutine Required: HARMA, REC1

1. Subroutine Name: REFORM

2. Purpose: Control generation of BCD input tape from Report Form Input Sheets.

3. Equations & Procedures: Storage is allocated for all variables needed by PHASE1 and PHASE2 combined. All valid input section names are stored by a data statement in array NAMES. Temporary tape storage for input sections which must be merged are assigned to scratch tapes NTAPE1 and NTAPE2. Subroutine PHASE1 is entered to read and store all data. Subroutine PHASE2 is entered to merge and output on INTAPE the data that was read in PHASE1. If a dump has been requested then the contents of INTAPE are printed on the system output unit. Control is then returned to the calling program.

4. Input Arguments:

INTAPE:	Tape unit number on which BCD input data is to be generated
NTAPE1,NTAPE2:	Scratch tape unit numbers
IN:	Input tape unit number
NRFP,NSS,NRF:	Adjustable dimension variables
COORD:	Storage area reserved for grid point coordinates.
T:	Storage area reserve for grid point temperatures.
P:	Storage area reserved for grid point pressures.
IBOUND:	Storage area reserved for grid point boundary conditions
IODISP:	Output unit of prescribed displacement matrix
NAMDIS:	Name of prescribed displacement matrix

5. Output Arguments:

ELPRM;	Modal values from element pressure input section.
ELTPM	Modal values from element temperature input section.
DINFO:	Array containing dynamics information for analysis
ERROR:	Logical variable indicating error condition.

6. Error Returns: If an error has occurred in PHASE1 or PHASE2 then ERROR is set to .TRUE..

7. Calling Sequence:

CALL REFORM (INTAPE, NTAPE1, NTAPE2, IN, NRFP, NSS, NRF, COORD, T, P, IBOUND, ERROR, IODISP, NAMDIS, DINFO, ELPRM, ELTPM)

8. Input Tapes:

IN - Scratch tape containing card images of data deck

9. Output Tapes:

INTAPE - BCD tape containing sorted data generated for subroutine INPUT

10. Scratch Tapes:

NTAPE1 - Temporary storage for grid point axes input, initial displacement input and element definition input.

NTAPE2 - Temporary storage area for grid point loads input, prescribed displacement input and special element input

11. Subroutine User: US04A

12. Subroutines Required:

PHASE1
PHASE2

13. Remarks: None

1. Subroutine Name: PHASE1

2. Purpose: Read, sort and store temporarily, all report form input data.

3. Equations & Procedures: First, all core storage areas are initialized with either blanks or zeroes. The following core storage areas are initialized with blanks: IBOUND, COORD, BM, LM, INM, PRM, EM and ERRMOD. The following core storage areas are initialized with zeroes: P, T, MEMORY, TM and PM.

   Reading of input is controlled entirely by label cards for each input section. Correlation between label codes and input sections is as follows:

Code	Input Section
TITLE	Title cards
COORD	Grid point coordinates
TEMP	Grid point temperatures
PRESS	Grid point pressures
BOUND	Grid point boundary conditions
MATER	Material library requests
LOADS	Grid point external loads
GRAXES	Grid point axes (matrices generated)
TRANS	Grid point axes (matrices input)
INITA	Grid point initial displacements
PRDISP	Grid point prescribed displacements
ELEM	Element definition data
EXTERN	Special element data
INPUT	Master input control
PRINT	Print controls
CALC	Calculation controls
END	End card
CHECK	Check card
SYSTEM	System control information
STST	Element stress-strain data
ELPR	Element Pressure data
ELTP	Element Temperature data

   After initialization, the data may be read from IN. The only restriction placed upon order of input sections is that SYSTEM may only be preceded by TITLE, MATER and/or INPUT.

   The procedure for a typical input section is as follows:

   (1) Subroutine LATCH is called to determine the identity of the input section.

(2) Control s transferred to the corresponding section of PHASE1 that will read and store the data. This step is accomplished either directly in PHASE1 itself or by a call to FORMIN.

(3) Data storing for a section terminates upon reading of a section label card which differs from the section being read.

Upon reading a CHECK or END card, PHASE1 returns control to the calling program.

4. Input Arguments:

NAMES:	Array containing valid input section labels
INTAPE:	Tape unit number on which BCD input data is to be generated
LOCATE:	Array containing tape unit numbers locating temporary tape storage for input sections. For each entry in NAMES there is a corresponding entry in LOCATE pointing to a temporary storage area. If the entry in LOCATE is a zero then storage is in core. If the entry is non-zero then storage is on the tape number indicated.
NUMCAL:	Number of possible solution techniques
NUMNAM:	Number of valid input section labels
ICASE:	Case number
NDIR:	Number of directions per grid point
NEND:	Last word of every input section placed on tape
IN:	Input tape unit number
NRFP:	Adjustable dimensions for COORD, T and P
NRF:	Adjustable dimension for IBOUND
DINFO:	Dynamics information

5. Output Arguments:

COORD:	Array containing grid point coordinates
T:	Array containing grid point temperatures
P:	Array containing grid point pressures
MEMORY:	Array containing indicators which record input sections that have been encountered during processing of data

IBOUND:	Array containing grid point boundary conditions
TM:	Array containing grid point temperature modal values
PM:	Array containing grid point pressure modal values
BM:	Array containing grid point boundary condition modal values
SM:	Array containing grid point load modal values for each load condition
INM:	Grid point initial displacement modal values
PRM:	Grid point prescribed displacement modal values
EM:	Special element input modal values
NLOAD:	Array containing number of points in each load condition
NINITA:	Array containing numbers of points in each initial displacement condition
NPRDIS:	Array containing number of points in each prescribed displacement condition
ICALC:	Array containing solution procedures desired
NREF:	Number of system referenced grid points
NREFP:	Number of input grid points
NTD:	Number of degrees of freedom per grid point
NL:	Number of load conditions
NID:	Number of initial displacement conditions
NPD:	Number of prescribed displacement conditions
NAXES:	Number of grid point axes systems
NELEM:	Number of elements
NM:	Number of requests of the material library tape
NREF4:	Number of input boundary condition grid points
TZERO:	System reference temperature
NREF4C:	Number of boundary condition points read by PHASE1
NREFPC:	Number of grid points read by PHASE1
NELEMC:	Number of elements read by PHASE1

NGRAXC:	Number of grid point axes systems read by PHASE1
NTRANC:	Number of grid point axes transformation matrices read by PHASE1
ERROR:	Error indicator
DUMPT:	Debug dump indicator
ELPRM	Array of element pressure modal values
ELTPM	Array of element temperature modal values
NPRKT	Number of entries in element pressure section
NTPKT	Number of entries in element temperature section
NPDL	Number of prescribed displacement conditions

6. Error Returns:

Message:	Action Taken:
Unexpected blank label card encountered.	Flush to next recognizable label card and insert check card.
No option has been selected for request number xxx of material library.	Flush to next recognizable label card and insert check card.
More than one option has been selected for request number xxx of material library.	Retain first selection encountered.
Maximum number of load conditions allowed is 100. This problem contains xxx.	Flush to next recognizable label card and insert check card.
Load condition xxx sub-label is incorrect. Program cannot distinguish between load conditions.	Flush to next recognizable label card and insert check card.
Illegal modal card encountered. Card will be ignored.	Self-explanatory
Due to previously encountered error condition this section is being skipped. Program will flush data deck until next recognizable section is encountered.	Self-explanatory
Unrecognizable input section.	Flush to next recognizable label card and insert check card.

Due to above error message Self-explanatory
this section will be omit-
ted and check card inser-
ted.

7. Calling Sequence:

CALL PHASE1 (COORD, T, P, MEMORY, IBOUND, NAMES, TM, PM, BM, SM, INM, PRM, EM, NLOAD, NINITA, NPRDIS, ICALC, NREF, NREFP, NTD, NL, NID, NPD, NAXES, NELEM, NM, NREF4, TZERO, INTAPE, LOCATE, NUMCAL, NUMNAM, ICASE, NDIR, NEND, NREF4C, NREFPC, NELEMC, NGRAXC, NTRANC, IN, NRFP, NRF, ERROR, DUMPT, DINFO, NPDC, ELPRM, ELTPM, NPRKT, NTPKT)

8. Input Tapes:

IN - BCD tape containing card images of data deck

9. Output Tapes:

NTAPE1 - Temporary storage for grid point axes input, initial displacement input, and element definition input

NTAPE2 - Temporary storage for grid point loads input, prescribed displacement input and special element input

INTAPE - TITLE, MATER, PRINT sections are output if they were present.

10. Scratch Tapes: None

11. Subroutine User: REFORM

12. Subroutines Required:

LATCH
FORMIN

13. Remarks: None

1. Subroutine Name: LATCH

2. Purpose: Compare a six character name to the recognizable list of input section names for Report Form Input.

3. Equations and Procedures: The six character name LABEL is compared to each of the legal input section names (array NAMES). If a match is found then LEADER is set to the position number in NAMES which contained the matching name. If no match is found then LEADER is set equal to one plus the number of legal section names.

4. Input Arguments:

   LABEL - name to be matched
   NUMNAM - number of valid input section names
   NAMES - array containing valid input section names

5. Output Arguments:

   LEADER - position number in NAMES of input section name which matches LABEL

   If no match was found then LEADER is set equal to NUMNAM + 1

6. Error Returns: None

7. Calling Sequence: CALL LATCH (LABEL, LEADER, NUMNAM, NAMES)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is $1BA_{16}$ Bytes.

12. Subroutine User: PHASE1

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: FORMIN

2. Purpose: Read and store on tape or in core all table form input to Phase One of Report Form Input Preprocessor.

3. Equations and Procedures: The decision to store data on tape or in core is determined by examining the input variable NTAPE. If NTAPE is less than or equal to zero then the data is stored in core, otherwise the data is stored on the unit specified by NTAPE. Any modal values read are always stored in core.

4. Input Arguments:

LEADER	: Index number referring to input section being processed
MEMORY	: Not used
NAMES	: Array containing legal input section labels
NTAPE	: Storage indicator, if≠0 then NTAPE contains unit number for external storage
AMODAL	: Storage reserved for modal values read, if any
MODAL	: Modal card label
NUMBER	: Number of input values to be read per card
REPEAT	: Logical variable indicating legality of repeat option
FMT1-5	: Input formats
MSG1-3	: Error message formats
WARN	: Error message warning flag
FATAL	: Error message fatal flag
NCARD	: Number of input cards per table entry
CORE	: Core storage area if data is to remain in core
NR, NC	: Adjustable dimensions of CORE
LABSUB	: Sub-label for multiple condition input sections
IN	: Unit number containing input data

5. Output Arguments:

LABEL	: Input section label encountered which was different from input section label now being processed
KOUNT	: Number of input table entries read
NERROR	: Error indicator
NCOND	: Condition number for encountered sub-label
SCALAR	: Constant for encountered sub-label

6. Error Returns: Error conditions are indicated in NERROR as follows:

   If NERROR equals zero, then no error has occurred
   If NERROR is less than zero, then a sub-label has been encountered
   If NERROR is greater than zero, then a fatal error has occurred and an appropriate message will be printed

7. Calling Sequence: Call FORMIN

   (LEADER, MEMORY, NAMES, LABEL, KOUNT, NTAPE, AMODAL, MODAL, NUMBER, REPEAT, FMT1, FMT2, FMT3, FMT4, FMT5, MSG1, MSG2, MSG3, WARN, FATAL, NERROR, NCARD, CORE, NR, NC, LABSUB, NCOND, SCALAR, IN)

8. Input Tape: IN contains input data

9. Output Tape: If NTAPE is greater than zero then it will contain the stored input, otherwise there is no output tape.

10. Scratch Tapes: None

11. Storage Required: Total storage required is $C74_{16}$ Bytes.

12. Subroutine User: PHASE1

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: PHASE2

2. Purpose: Merge, order and output form input data stored by PHASE1.

3. Equations and Procedures: The input sections stored by PHASE1 are detected by examining the array MEMORY. The exact procedure is to check the MEMORY array in the order required for output and if the MEMORY value for that section is greater than zero then output that section's stored data; otherwise continue to the next section. The order in which the stored input sections are output, if present, and the sections that they are to be merged with is as follows:

Input Section	Generated from Report Form Input Sections
NREF	SYSTEM
TZERO	SYSTEM
GRID	COORD, TEMP, PRESS
BOUNDS	BOUND, CALC, INITA, PRDISP
ELEM	ELEM, EXTERN
TRANS	TRANS
GRAXES	GRAXES
LOADS	LOADS
END	END
CHECK	CHECK

4. Input Arguments:

COORD	: Array containing system grid point coordinates
T	: Array containing grid point temperatures
P	: Array containing grid point pressures
MEMORY	: Array indicating report from input sections read
IBOUND	: Array containing grid point boundary conditions
NAMES	: Array containing legal report form input section names
TM	: Array containing modal values for grid point temperatures
PM	: Array containing modal values for grid point pressures
BM	: Array containing modal values for grid point boundary conditions
SM	: Array containing modal values for grid point load conditions
INM	: Array containing modal values for initially displaced grid points

PRM	: Array containing modal values for prescribed displaced grid points
EM	: Array containing modal values for special element input
NLOAD	: Number of loaded grid points per load condition
NINITA	: Number of initial displacement conditions
NPRDIS	: Number of prescribed displacement conditions
ICALC	: Array containing solution codes
NREF	: Number of grid points in system
NREFP	: Number of input grid points
NTD	: Number of degrees of freedom per grid point
NL	: Number of load conditions
NID	: Number of initially displaced grid points
NPD	: Number of prescribed displaced grid points
NAXES	: Number of grid point axes transformation systems
NELEM	: Number of elements
NM	: Number of requests of material library
NREF4	: Number of input boundary condition points
TZERO	: System base temperature
INTAPE	: Unit on which processed output is to be written
LOCATE	: Array indication storage location of input sections
NUMCAL	: Number of solution codes
NUMNAM	: Number of legal report form input section labels
ICASE	: Not used
NDIR	: Number of directions per grid point
NEND	: Not used
NREF4C	: Number of input boundary condition points actually read
NREFPC	: Number of input grid points actually read
NELEMC	: Number of input elements actually read

NGRAXC	: Number of input grid point axes systems actually read (transformation matrices generated)
NTRANC	: Number of input grid point axes systems actually read (transformation matrices input)
IN	: Not used
NRFP	: Not used
NRF	: Adjustable dimension for COORD, T, P, and IBOUND
NPRKT	: Number of entries in element pressure section
NTPKT	: Number of entries in element temperature section
IODISR	: Output unit number for prescribed displacement matrix
NAMDIS	: Name of prescribed displacement matrix
NPDC	: Number of prescribed displacement conditions

5. Output Argument

DINFO	: Array containing dynamics information for eigenvalue analysis
ERROR	: Error indicator

6. Error Returns: Error messages are indicated in Appendix. If an error occurs logical variable ERROR is set to TRUE and control is returned to the calling program.

7. Calling Sequence: Call PHASE2

(COORD, T, P, MEMORY, IBOUND, NAMES, TM, PM, BM, SM, INM, PRM, EM, NLOAD, NINITA, NPRDIS, ICALC, NREF, NREFP, NTD, NL, NID, NPD, NAXES, NELEM, NM, NREF4, TZERO, INTAPE, LOCATE, NUMCAL, NUMNAM, ICASE, NDIR, NEND, NREF4C, NREFPC, NELEMC, NGRAXC, NTRANC, IN, NRFP, NRF, ERROR, DINFO, IODISP, NAMDIS, NPDC, NPRKT, NTPKT)

8. Input Tapes: The array LOCATE contains the unit number, if any, on which data was stored by subroutine PHASE1.

9. Output Tape: INTAPE contains processed output.

10. Scratch Tapes: None

11. Subroutine User: REFORM

12. Subroutine Required: OPEN

13. Remarks: None

1. Subroutine Name: PDISP

2. Purpose: Generate prescribed displacement matrix if required.

3. Equations and Procedures:

   A. Check if matrix name is suppressed, if it is then return (no matrix is output).
   B. Check if NPDC≠1 and NPDC<NL print error message and return.
   C. Use EUTL5 to write matrix header.
   D. If MODAL array is blank insert zeros, if not insert MODAL values into displacement column.
   E. Loop on number of grid points for which values were given, inserting them into the displacement column.
   F. Compress column each time, using EUTL8, and write it out.
   G. If column compresses to zero skip write out.
   H. Do (D) to (G) for each prescribed displacement condition.
   I. At end use EUTL6 to write matrix trailer.

4. Input Arguments:

NREF	-	number of system grid points
NTD	-	number of degrees of freedom/point (NDEG*NDIR)
NL	-	number of external load conditions input
PRM	-	array of modal values/condition
NPROIS	-	number of input points/condition
IODISP	-	output logical unit number of matrix
NAMDIS	-	name of output matrix array (7 elements long)
NPDC	-	number of prescribed displacement conditions input
DISP	-	(array area used by IBOUND array used in PHASE2- now used to store displacement column)
KTAPE	-	tape logical unit number used for displacement input

5. Output Arguments: ERROR - logical variable true if error return is used.

6. Error Returns: If NPDC≠1 and NPDC<NL.

7. Calling Sequence:

```
Call PDISP(NREF,NTD,NL,PRM,NPRDIS,NPDC,IODISP,NAMDIS,DISP,
          KTAPE,ERROR)
```

8. Input Tapes: KTAPE - See Item 4.

9. Output Tapes: NPOT - standard print out unit; IODISP - See Item 4

10. Scratch Tapes: None

11. Storage Required: Total Storage required is $7DE_{16}$ Bytes.

12. Subroutine User: PHAS22

13. Subroutine Required:

EUTL5
EUTL6
EUTL8

14. Remarks: None

1. Subroutine Name: OPEN

2. Purpose: Select a unit and then locate the requested input section on that unit

3. Equations and Procedures: The correct unit number is extracted from the array LOCATE. The unit is then searched for the requested input section. Searching starts from the present position of the unit and allows the end of the unit's extent to be reached twice before the search is abandoned.

4. Input Arguments:

LEADER	: Identification number of input section being processed
NAMES	: Array containing valid labels
LOCATE	: Array containing corresponding logical units for valid labels
*	: Non-standard return for error condition

5. Output Arguments:

NTAPE	: Unit containing requested input section

6. Error Returns: If the requested input section is not located on the selected unit the non-standard return is used.

7. Calling Sequence: Call OPEN

   (LEADER, NAMES, LOCATE, NTAPE, $XXXXX) where XXXXX is the statement number to which control is returned in the calling program if an error occurs.

8. Input Tapes: The array LOCATE contains the logical unit numbers which may be input tapes.

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total storage required is $24A_{16}$ Bytes.

12. Subroutine User: PHASE2

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: CHEK

2. Purpose: Perform input/output cross-checking for USER04 module

3. Equations and Procedures: The required input sections for the selected output matrices are indicated in the array MASTER (see subroutine LOGFLO). The actual input sections processed are indicated in the array ICONT. The logical array, GO, is set according to the information in MASTER as compared with ICONT. If an output matrix requires an input section that is not present then a message is printed giving the matrix name and corresponding position in the GO array is set to .FALSE.

4. Input Arguments:

NAMOUT	: Array containing output matrix names
NUMOT	: Number of output matrices
NAMIN	: Array containing input matrix names
NUMIN	: Number of input matrices
MASTER	: Array indicating required input sections
NUMAST	: Length of MASTER
ICONT	: Array indicating processed input sections
NCONT	: Length of ICONT

5. Output Arguments:

GO	: Array indicating input requirements have been satisfied, one position for each possible output matrix

6. Error Returns: None

7. Calling Sequence:

(NAMOUT, NUMOT, NAMIN, NUMIN, MASTER, NUMAST, ICONT, NCONT, GO)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total storage required is $59A_{16}$ Bytes.

12. Subroutine User: USO4A

13. Subroutines Required: NTEST

14. Remarks: None

1. Subroutine Name: OUTINT

2. Purpose: Output interpreted input data as a matrix

3. Equations and Procedures: After processing the input data deck, all necessary information is stored in three areas. System control information is stored in the array KNMD and in the first two records on scratch unit NTAPE1. Element generation data is stored on scratch unit NTAPE3. All of this data is output as a matrix, the first column containing KNMD, the second and third columns containing the first two records from NTAPE, the fourth column containing two words (number of elements, NELEM, and grid point axes indicator) and the last 2*NELEM columns containing the input element generation data.

4. Input Arguments:

NAMOUT	: Array containing output matrix name
IOSPEC	: Array containing unit specifications for output matrix
NTAPE1	: Unit containing system control information
NTAPE3	: Unit containing element generation data
KNMD	: Array containing system control information
NUMK	: Length of KNMD
IWORK	: Work storage area

5. Output Arguments: None

6. Error Returns: None

7. Calling Sequence:

(NAMOUT, IOSPEC, NTAPE1, NTAPE3, KNMD, NUMK, IWORK)

8. Input Tapes:

NTAPE1	: Unit containing system control information
NTAPE3	: Unit containing element generation data

9. Output Tapes:

IOSPEC(1): Unit which will contain interpreted input data matrix

10. Scratch Tapes: None

11. Storage Required: Total storage required is $6DE_{16}$ Bytes.

12. Subroutine User: US04A

13. Subroutine Required:

EUTL5
EUTL6

14. Remarks: None

1. Subroutine Name: FLOADS

2. Purpose: To generate a matrix of external grid point loads which is acceptable to Format.

3. Equations and Procedures: A grid point load matrix, PCOL, is read from NTAPE4 for each load condition. It is then converted into compressed format and stored on tape IØSPEC.

   The matrix dimensions are NSYS x NL, where NSYS is the size of the total assembled load column and NL is the number of grid point load conditions.

4. Input Arguments

   NSYS - Size of total assembled load column
   NAMOUT- Array containing output matrix name for load matrix
   IOSPEC- Output tape unit number for loads matrix
   NTAPE4- Input tape unit number containing loads matrix
   PCOL - Core storage area for loads matrix

5. Output Arguments: None

6. Error Returns: None

7. Calling Sequence: CALL FLØADS (NSYS, NAMØUT, IOSPEC, NTAPE4, PCØL)

8. Input Tapes: NTAPE4

9. Output Tapes: IOSPEC

10. Scratch Tapes: None

11. Storage Required: Total Storage required is $34A_{16}$ Bytes.

12. Subroutine User - USO4A

13. Subroutines Required - None

14. Remarks: None

1. Subroutine Name: FTR

2. Purpose: To generate a matrix which will transform another matrix from full system coordinates to "reduced" system, i.e. boundary condition constrained.

3. Equations and Procedures: The matrix TR is of order NMDB X NSYS such that if J = LIST (I), then the element TR (I, J) = 1.0. LIST contains the row numbers of the full system which are to be retained in the reduced matrix. Only fixed bounds are reduced out as indicated by KODE = 0 in input data bounds.

   Each column is generated and stored on tape as defined by FORMAT. Each column record consists of: J, 1, 2, 1.0, where J = LIST (i).

4. Input Arguments: NMDB - order of reduced matrix
   NSYS - order of full system
   NAMØUT - matrix name of TR
   IØSPEC - matrix output tape for TR

5. Output Arguments: None

6. Error Returns: None

7. Calling Sequence:

```
CALL FTR (NSYS, LIST, NTAPE1, NAMØUT, IØSPEC)
```

8. Input Tape: NTAPE1
   Record #1 COM1 (not required)
   Record #2 NMDB1, NMDB, (LIST (I), I=1, NMDB)

9. Output Tapes: IOSPEC - Format Output Tape Number

10. Scratch Tapes: None

11. Storage Required: Total Storage required is 348_{16} Bytes.

12. Subroutine User: US04A

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: TSYS

2. Purpose: To output as a format matrix system constants needed outside of the USERO4 module.

3. Equations and Procedures: The array NMD containing system constants generated in the input phase of the USERO4 module is passed to subroutine TSYS by subroutine USERO4A. These constants are then converted to floating point variables and outputs a matrix with 2 columns to the format system.

   The constants that are output are as follows in their respective order:

NDIR	-	number of directions
NDEG	-	number of types of degrees of freedom
NREF	-	Highest reference node in element connections
NMDB	-	the order of the reduced system = NMDB1+NMDB2
NMDB0	-	the number of zero boundary conditions
NMDB1	-	The number of ones
NMDB2	-	the number of twos
NMDB01	-	the number of zeros plus ones
NMDB12	-	the number of ones plus twos
NTYPE	-	code for element degrees of freedom NTYPE = 0 for 3 types of D.O.F. NTYPE = 1 for 1 or 2 types of D.O.F.
NSYS	-	the total number of system degrees of freedom equals NDIR*NDEG*NREF
NELEM	-	the number of elements in the analyses
NL	-	the number of external load conditions in the analysis

4. Input Arguments:

NMD	-	array of system constants
NAMOUT	-	Array containing the name of the format matrix
NSET	-	logical unit number matrix is to be written on
NREF	-	highest reference node in element connections
DINFØ	-	array containing dynamics information for eigen value analysis
DINFØ(1)	-	the number of eigen values requested
DINFØ(2)	-	the convergence criteria
DINFØ(3)	-	the maximum number of iterations
DINFØ(4)	-	control for debug print
DINFØ(5)	-	first normalizing element for print
DINFØ(6)	-	second normalizing element for print
DINFØ(7)	-	control for guess vector iterations start

5. Output Arguments:

   IHONT - contains harmonic data

6. Error Returns: None

7. Calling Sequence: TSYS(NMD,NAMOUT,NSET,NREF,IHONT)

8. Input Tapes: None

9. Output Tapes: NSET

10. Scratch Tapes: None

11. Subroutine User: US04A

12. Subroutine Required: EUTL5,EUTL6

13. Remarks: Note that these constants have been converted to floating point numbers

1. Subroutine Name: US04B

2. Purpose: Control Phase Two and Phase Three operations (element matrix generation and element matrix output, respectively).

3. Equations and Procedures: System control information is extracted from the array KNMD. Scratch units are assigned from the array ISSPEC. If input displacements are present then subroutine DEFLEX is called to record the input displacements on scratch unit NTAPE4. If the interpreted input matrix position is non-blank then subroutine ININT is called to generate input tapes NTAPE1 and NTAPE3. Subroutine FELEM is called to control the generation of the element matrices. And, finally, subroutine OUTMAT is called to place the generated matrices into the Format System.

4. Input Arguments:

NUMOT	: Number of output matrices
NAMOUT	: Array containing names of output matrices
IOSPEC	: Array containing unit specifications for output matrices
NUMIN	: Number of input matrices
NAMIN	: Array containing names of input matrices
INSPEC	: Array containing unit specification for input matrices
NUMSR	: Number of available scratch units
ISSPEC	: Array containing scratch unit specifications
NUMSC	: Number of scalars
SCALAR	: Array containing scalars
NWORKR	: Number of available storages in work area
WORK	: Work area
IPRINT	: System print control
KNMD	: System control information
MASTER	: Array containing input/output cross-checking codes
NUMAST	: Length of MASTER
NUMK	: Length of KNMD

5. Output Arguments:

ERROR	: Logical variable indicating error condition

6. Error Return: If an error is detected in element matrix generation or in element matrix output then ERROR is set to TRUE and control is returned to the calling program.

7. Calling Sequence: Call US04B

(NUMOT, NAMOUT, IOSPEC, NUMIN, NAMIN, INSPEC, NUMSR, ISSPEC, NUMSC, SCALAR, ERROR, NWORKR, WORK, IPRINT, KNMD, MASTER, NUMAST, NUMK)

8. Input Tapes:

NTAPE1 : Contains system control information
NTAPE3 : Contains interpreted element input

9. Output Tapes:

IOSPEC(1,6)	: Reserved for assembly transformation matrix
IOSPEC(1,7)	: Reserved for element stiffness matrices
IOSPEC(1,8)	: Reserved for element load matrices
IOSPEC(1,9)	: Reserved for element stress matrices
IOSPEC(1,10)	: Reserved for element thermal stress matrices
IOSPEC(1,11)	: Reserved for element incremental stiffness matrices
IOSPEC(1,12)	: Reserved for element mass matrices

10. Scratch Tape:

NTAPE2	: Contains element generated matrices in compact form
NTAPE4	: Contains input displacements, if present

11. Storage Required: Total storage required is $7D8_{16}$ Bytes.

12. Subroutine User: US04

13. Subroutines Required:

NTEST
ININT
DEFLEX
FELEM
OUTMAT

14. Remarks: None

1. Subroutine Name: ININT

2. Purpose: Restore data from interpreted input matrix

3. Equations and Procedures: Subroutine EUTL3 is called to locate the input matrix. The first column of the matrix contains system control information and is read into KNMD. Columns two and three contain further system information and are recorded as the first two records on NTAPE1. Column four and all succeeding columns contain element generation input data and are recorded on NTAPE3.

4. Input Arguments:

NAMIN	: Array containing input matrix name
INSPEC	: Array containing unit specifications for input matrix
NTAPE1	: Unit reserved for system control information
NTAPE3	: Unit reserved for element generation input data
KNMD	: Array reserved for system control information
IWORK	: Work storage area
NUMK	: Length of KNMD

5. Output Arguments:

IER	: Logical variable indicating error condition

6. Error Returns: If the input matrix cannot be located, or a word count error occurs for columns one or four, or the matrix trailer record is encountered unexpectedly, then IER is set to .TRUE..

7. Calling Sequence:

(NAMIN, INSPEC, NTAPE1, NTAPE3, KNMD, IWORK, NUMK, IER)

8. Input Tapes:

INSPEC(1) : Unit containing interpreted input matrix

9. Output Tapes:

NTAPE1	: Unit reserved for system control information
NTAPE3	: Unit reserved for element generation input data

10. Scratch Tapes: None

11. Storage Required: Total storage required is 912_{16} Bytes.

12. Subroutine User: US04B

13. Subroutines Required: EUTL3

14. Remarks: None

1. Subroutine Name: DEFLEX

2. Purpose: Sort input displacement matrix into separate element input sections

3. Equations and Procedures: The input displacements for the system are read into the IWORK array and restored at the end of the IWORK array. For each element, the following procedure is invoked: the element generation input data is read from scratch unit NTAPE3; the array containing the element definition points is extracted; the input displacements corresponding to these points are selected from the system input displacements and written on scratch unit NTAPE4.

4. Input Arguments:

NSYS	: Total degrees of freedom in system
NAMIN	: Array containing input matrix name
INSPEC	: Array containing unit specifications for input matrix
NTAPE3	: Unit containing element generation input
NTAPE4	: Unit reserved for element input displacements
IWORK	: Work storage area
NWORK	: Length of IWORK
MAXNI1	: Maximum length of record on NTAPE3

5. Output Arguments:

IER	: Logical variable indicating error condition

6. Error Returns: If the input matrix cannot be found, or its dimensions are not NSYS by one or IWORK does not contain sufficient storage locations then IER is set to .TRUE..

7. Calling Sequence:

```
(NSYS, NAMIN, INSPEC, NTAPE3, NTAPE4, IWORK, NWORKR,
 MAXNI1, IER)
```

8. Input Tapes:

NTAPE3	: Unit containing element generation input data
INSPEC(1)	: Unit containing system input displacement matrix

9. Output Tapes:

NTAPE4	: Unit reserved for element input displacements

10. Scratch Tapes: None

11. Storage Required: Total storage required is $AD4_{16}$ Bytes.

12. Subroutine User: US04B

13. Subroutines Required:

    EUTL3
    EUTL9

14. Remarks: None

1. Subroutine Name: FELEM

2. Purpose: Set element matrix generation controls and initiate matrix generation.

3. Equations and Procedures: Logical unit definitions are assigned to their structural system functions. An array, IWORK, is reserved for storage of generation controls and system information. The generation controls are determined by examining the output matrix names and the system information is retrieved from unit NTAPE1. Subroutine SQUISH is called to compute matrix suppression controls. The number of elements is read from unit NTAPE3 and subroutine ELPLUG, which selects the correct element type, is called for each element.

4. Input Arguments:

KP	: Not used
NTAPE1	: Logical unit containing system control information
NTAPE2	: Logical unit reserved for generated element matrices
NTAPE3	: Logical unit containing interpreted element input
NORDM	: Maximum element degrees of freedom
NRSELM	: Maximum element stress order
NOINKM	: Maximum storage required for element stiffness matrix
NIAM	: Maximum storage required for element matrix record on NTAPE2
NTAPE4	: Logical unit containing input displacements, if present

5. Output Arguments:

ERROR	: Logical variable indicating error condition

6. Error Returns: If an error occurs in generation of element matrices then ERROR is set to .TRUE. and control is returned to the calling program.

7. Calling Sequence: Call FELEM

(KP, NTAPE1, NTAPE2, NTAPE3, NORDM, NRSELM, NOINKM, NIAM, ERROR, NTAPE4)

8. Input Tapes:

NTAPE1	: Contains system control information
NTAPE3	: Contains interpreted element input

9. Output Tapes:

   NTAPE2 : Reserved for compact storage of element generated matrices

10. Scratch Tapes: None

11. Storage Required: Total storage required is $5D8_{16}$ Bytes.

12. Subroutine User: US04B

13. Subroutines Required: ELPLUG, SQUISH

14. Remarks: None

1. Subroutine Name: SQUISH

2. Purpose: Set matrix suppression codes for element generation phase

3. Equations and Procedures: The indicators are initially set to zero, signifying suppression is desired. Subroutine NTEST is called to examine the output matrix names for suppression selections. For each non-suppressed matrix position encountered the corresponding indicator is reset to one.

4. Input Arguments:

NAMOUT	: Array containing matrix names
NUMOT	: Number of output matrices

5. Output Arguments:

KK	: Suppression indicator for element stiffness matrices
KF	: Suppression indicator for element load matrices
KS	: Suppression indicator for element stress matrices
KN	: Suppression indicator for element incremental stiffness matrices
KM	: Suppression indicator for element mass matrices
KDS	: Suppression indicator for element structural damping matrices
KDV	: Suppression indicator for element viscous damping matrices
KTS	: Suppression indicator for element thermal stress matrices

6. Error Returns: None

7. Calling Sequence:

(NAMOUT, KK, KF, KS, KN, KM, KDS, KDV, KTS, NUMOT)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total storage required is $3DE_{16}$ Bytes.

12. Subroutine User: FELEM

13. Subroutines Required: NTEST

14. Remarks: None

1. Subroutine Name: ELPLUG

2. Purpose: Select proper element type to generate requested element matrices.

3. Equations and Procedures: Subroutine REC1 is called to obtain the interpreted element input. If input displacements were present then the values are retrieved from unit NTAPE4. Included in the interpreted element input is the element type code number (plug number). From this data the proper plug subroutine is called and the requested element matrices are generated. If the plug number is five, six or fourteen the grid point axes transformations are then applied. If the plug number was one, two or seven then grid point axes transformations were applied inside the plug. Subroutines REC3 and REC4 are called to write as external units element control data and the generated element matrices, respectively. Finally, if an element matrix print has been requested then subroutine ELPRT is called to perform the printing.

   There is only one exception to the above procedure. If the option to repeat element matrices has been selected (IP = -2), then the plug subroutine is bypassed and element matrices from the previous element are written again by REC3 and REC4.

4. Input Arguments: The input arguments contained in JWORK are:

JWORK(1)-IEL	: Element generation sequence number (IEL = 1,2,3, . . . , NELEM)
JWORK(2)-ITAPE	: Indicator controlling writing of matrices on external unit
JWORK(3)-KK	: Element stiffness matrix suppression control
JWORK(4)-KF	: Element load matrix suppression control
JWORK(5)-KS	: Element stress matrix suppression control
JWORK(6)-KM	: Element mass matrix suppression control
JWORK(7)-KDS	: Not used
JWORK(8)-KDV	: Not used
JWORK(9)-KN	: Element incremental stiffness matrix suppression control
JWORK(11)-NMDB	: Not used
JWORK(12)-NDIR	: Number of directions per grid point
JWORK(13)-NDEG	: Number of solution degrees of freedom per grid point
JWORK(14)-ICONT	: Grid point axes transformation indicator

JWORK(15)-NTAPE2 : Unit number reserved for generated element matrices
JWORK(16)-NTAPE3 : Unit number containing interpreted element input
JWORK(18)-ILP : Internal element type code
JWORK(19)-IPL : Input element type code
JWORK(20)-NTAPE4 : Unit number containing input displacement, if present
JWORK(21)-INDISP : Variable indicating presence of input displacements

Other input arguments are:

NUMOT : Number of output matrices
NAMOUT : Array containing output matrices names

5. Output Arguments: Input and output arguments are contained in the array JWORK. The output arguments contained in JWORK are:

JWORK(10)-NORD : Element degrees of freedom
JWORK(17)-NIAM : Maximum number of storages required to write a record on unit NTAPE2
JWORK(20)-NERR : Returning error code,
if NERR is zero then no error has occurred,
if NERR is one then element type code number is incorrect,
if NERR is two then the number of element defining points is incorrect,
if NERR is three then the special element input is incorrect, and
if NERR is four then the number of element degrees of freedom is incorrect.

6. Error Returns: If NERROR is not zero upon return from ELPLUG, then an error has occurred.

7. Calling Sequence: Call ELPLUG (JWORK, NUMOT, NAMOUT)

8. Input Tape:

NTAPE3 : Unit containing interpreted element input

9. Output Tape:

NTAPE2 : Unit reserved for generated element matrices

10. Scratch Tapes: None

11. Subroutine User: FELEM

12. Subroutines Required:

REC1
PLUG1
PLUG2
PLUG5
PLUG6
PLUG7
PLUG14
AXTRA3
AXTRA2
AXTRA1
REC3
REC4
ELPRT
HTC0
PLUG25
PLUG10
PLUG13
PLUG23
PLUG20
PLUG26

13. Remarks: Storage for the generated element matrices and work areas required by ELPLUG is allocated by equivalencing into the blank common work area starting at location 1001 and extending to location 6000. Work storage for the various element types is allocated by equivalending into the blank common work area at location 6001.

1. Subroutine Name: REC3

2. Purpose: Write or read element control information tape records.

3. Equations and Procedures: The decision to read or write the record is determined by examining the input variable IOPT in the following manner:

   if IOPT $\leq$ 1 the record is read
   if IOPT $\geq$ 2 the record is written

4. Input Arguments: (if IOPT$\geq$2)

IOPT:	Read/write indicator
K:	Fortran logical unit number
NI3:	Number of words in record (excluding NI3)
JEL:	Element number
IPL:	Element type code number (plug number)
NLIST:	Element order (number of degrees of freedom per point * number of points)
LISTEL:	Vector containing boundary condition information for element
NIA:	Not used (set equal to one)
IAKEL:	Not used

5. Output Arguments: (if IOPT 1)

   Given the proper value of IOPT, all of the above input arguments will be output arguments with the exception of IOPT and K, which are always input arguments.

6. Error Returns: None

7. Calling Sequence:

   CALL REC3 (IOPT, K, NI3, JEL, IPL, NLIST, LISTEL, NIA, AKEL)

8. Input Tape: If IOPT$\leq$1, then K is an input tape.

9. Output Tape: If IOPT$\geq$2, then K is an output tape.

10. Scratch Tape: None

11. Storage Required: Total storage required is 368_{16} Bytes.

12. Subroutine User: ELPLUG

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: REC4

2. Purpose: Read or write generated element matrices records.

3. Equations and Procedures: The decision to read or write the record is determined by examining the input variable IOPT in the following manner:

   if IOPT$\leq$1 then a record is read
   if IOPT$\geq$2 then a record is written

4. Input Arguments: (when IOPT$\geq$2)

IOPT:	Read/write indicator
K:	Fortran logical unit number
NOINK:	Number of storages required for stiffness and incremental stiffness matrices
AKELT:	Element stiffness matrix
NORD:	Number of storages required for element loads matrix
FTEL:	Element loads matrix
NNO:	Number of element defining points (node points)
NODES:	Grid point numbers defining element
NSEL:	Number of storages required for element stress matrix
NRSEL:	Number of rows in element stress and thermal stress matrices, also number of storages required for element thermal stress matrix
SEL:	Element stress matrix
SZALEL:	Element thermal stress matrix
ANEL:	Element incremental stiffness matrix
FNEL:	Not used
NMASS:	Number of storages required for element mass matrix
AMASS:	Element mass matrix
NDMPV:	Number of storages required for element viscous damping matrix
DAMPV:	Element viscous damping matrix
NDMPS:	Number of storages required for element structural damping matrix
DAMPS:	Element structural damping matrix

5. Output Arguments: (when IOPT $\leq$1)

   NI4 - number of words contained in record (excluding NI4)
   All of the above input arguments are output arguments given the correct value of IOPT except for IOPT and K which are always input arguments.

6. Error Returns: None

7. Calling Sequence:

   CALL REC4 (IOPT, K, NI4, NOINK, AKELT, NORD, FTEL, NNO, NODES, NSEL, NRSEL, SEL, SZALEL, ANEL, FNEL, NMASS, AMASS, NDMPV, DAMPV, NDMPS, DAMPS)

8. Input Tape: If IOPT $\leq$1 then K is an input unit.

9. Output Tape: If IOPT $\geq$ 2 then K is an output unit.

10. Scratch Tapes: None

11. Storage Required: Total storage required is 850_{16} Bytes.

12. Subroutine User: ELPLUG

13. Subroutines Required: None

14. Remarks: None

1. Subroutine name: MINV

2. Purpose: Invert a matrix.

3. Equations and Procedures: The standard Gauss-Jordan Method is used in which the inverted matrix is stored back on itself.

4. Input Arguments:

   A: Matrix to be inverted
   N: Order of matrix
   D: Determinant of matrix
   L: Work vector of length N
   M: Work vector of length N

5. Output Arguments: A - Contains the inverted matrix

6. Error Returns: If D = 0, matrix is singular.

7. Calling Sequence: CALL MINV (A, N, D, L. M)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is 818_{16} Bytes.

12. Subroutine User: TRAIC, NEWFT, PLUG1, PTBM, PTBF, MATPR, NEWFT1

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: AXTRA2

2. Purpose: Apply grid point axes transformation by post-multiplication using either the actual transformation matrix or its transpose.

3. Equations and Procedures:

$$[M_{OUT}] = [M_{IN}][\Gamma_{GPA}] \quad \text{or} \quad [M_{OUT}] = [M_{IN}][\Gamma_{GPA}]^T$$

where: $[M_{IN}]$ is the input element matrix,

$[\Gamma_{GPA}]$ is the element grid point axes transformation matrix,

$[M_{OUT}]$ is the output transformed element matrix,

$[M_{OUT}]$ is stored in the same location as $[M_{IN}]$, therefore, the input element matrix is lost once the multiplication has been effected. Advantage is taken, during multiplication, of the fact that $[\Gamma_{GPA}]$ is structured as a set of (3 x 3) or (2 x 2) matrices with main diagonal positions lying on the main diagonal of $[\Gamma_{GPA}]$.

4. Input Arguments:

GPAXEL : Element grid point axes transformation matrix, $[\Gamma_{GPA}]$
SEL : Input element matrix $[M_{IN}]$
NROW : Number of rows in SEL
NNO : Number of element node points
NDEG : Number of degrees of freedom
NDIR : Number of directions
IPL : Element plug number
ITRAN : Control code, if ITRAN = 0, then $[M_{OUT}] = [M_{IN}][\Gamma_{GPA}]^T$
if ITRAN = 1, then $[M_{OUT}] = [M_{IN}][\Gamma_{GPA}]$

5. Output Arguments:

SEL : Output transformed element matrix, $[M_{OUT}]$

6. Error Returns: None

7. Calling Sequence:

CALL AXTRA2 (GPAXEL, SEL, NROW, NNO, NDEG, NDIR, IPL, ITRAN)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is $4D6_{16}$ Bytes.

    ROW (3)
    ISAVE (3)

12. Subroutine User: ELPLUG, PLUG7, PLUG2, CK22, CK11

13. Subroutine Required: None

14. Remarks: The output matrix is stored in the input matrix storage. Grid point axes transformation is not applied to the rotation terms at the mid-points of the quadrilateral thin shell and the triangular thin shell elements.

1. Subroutine Name: MAB

2. Purpose: To evaluate the matrix product A * B = AN

3. Equations & Procedures:

$$AN_{nm} = \sum_{J} A_{nj} * B_{jm}$$

4. Input Arguments:

A: Elements of [A] matrix
B: Elements of [B] matrix
N: Number of rows in [A] matrix
L: Number of columns/rows in [A] [B] matrix
M: Number of columns in [B] matrix
N1,M1: Dimension of [A] matrix
N2,M2: Dimension of [B] matrix

5. Output Arguments:

AN: The matrix product

6. Error Returns: None

7. Calling Sequence: CALL MAB (A,B,AN,N,L,M,N1,M1,N2,M2)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is $2F6_{16}$ Bytes.

A(1)
B(1)
AN(1)

12. Subroutine User: Used by many subroutines within the MAGIC program

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: MSB

2. Purpose: To evaluate the matrix product of a symmetric-bottom half matrix and a rectangular matrix

3. Equations & Procedures:

$$AN_{nm} = \sum_{e} S_{ne} * B_{em}$$

4. Input Arguments:

S: Elements of [S] matrix (symmetric)
B: Elements of [B] matrix
N: Number of rows in the [S], [B] and [AN] matrices (order)
M: Number of columns in the [B] and [AN] matrices (order)
N1 and M1: Dimensions of the [B] and [AN] matrices

5. Output Arguments: AN: Matrix product

6. Error Returns: None

7. Calling Sequence: CALL MSB (S,B,AN,N,M,N1,M1)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is 312_{16} Bytes.

B(1)
S(1)
AN(1)

12. Subroutine User: Used by various subroutines within the MAGIC Program.

13. Subroutines Required: None

14. Remarks: [S] is of the form

$$\begin{bmatrix} S_{1\,1} & & & \\ S_{2\,1} & S_{2\,2} & & \\ \vdots & \vdots & \vdots & \\ S_{N1\,1} & S_{N12} & \cdots & S_{N1\,N1} \end{bmatrix}$$

1. Subroutine Name: BCB

2. Purpose: To evaluate the triple product of the transpose of a matrix A, a symmetric matrix S and the A matrix.

3. Equations and Procedures:

$$AN_{mm} = \sum_{n} \sum_{n} A^{T}_{mn} * S_{nn} * A_{nm}$$ (See remark 1)

4. Input Arguments:

A: The elements of the [A] matrix
SYM: The elements of the [S] matrix (symmetric-bottom half)
ND,MD: Dimensions of a matrix
N,M: Order of A matrix
N1: Number of rows to be deleted in multiplication
SCAL: Scalar quantity
IASSY: (see remark 2)

5. Output Arguments:

AN: Elements of the matrix AN which is the final product

6. Error Returns: None

7. Calling Sequence:

CALL BCB (A, SYM, AN, ND, MD, N, M, N1, SCAL, IASSY)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is 542_{16} Bytes.

12. Subroutine User: Various routines within MAGIC

13. Subroutines Required: None

14. Remarks:

1. In the summations, the n's must be replaced by dummy subscripts, running from 1 to n. The dummy must be used (ie. $\sum_{j=1}^{n}$ $\sum_{r=1}^{n}$) to ensure proper summing.

2. IASSY controls the summation procedure.

   If IASSY = 1, AN will be the sum of the calculated AN and all previous calculations of AN.
   If IASSY = 0, AN will be the triple product for this calculation.

1. Subroutine Name: MATB

2. Purpose: Subroutine to evaluate the matrix product of A transpose and B.

3. Equations and Procedures:

$$AN_{nm} = \sum_{e} A^{T}_{en} * B_{em}$$

where

A^{T}_{en} is the transpose of A_{ne}.

4. Input Arguments:

A: elements of [A] matrix
B: elements of [B] matrix
N: number of rows in [A] matrix (order)
L: number of columns in [A] matrix (order)
M: number of rows in [B] matrix (order)
N1,M1: dimension of [A] matrix
N2,M2: dimension of [B] matrix

5. Output Arguments:

AN: elements of matrix product

6. Error Returns: None

7. Calling Sequence: CALL MATB (A, B, AN, N, L, M, N1, M1, N2, M2)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is 306_{16} Bytes.

12. Subroutine User: Various subroutines in MAGIC

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: SYMPRT

2. Purpose: To print a symmetric matrix as output

3. Equations and Procedures: Not Applicable

4. Input Arguments:

   SYM: Elements of the symmetric matrix
   N1: Matrix identification number
   N2: Dimension of matrix

5. Output Arguments: None

6. Error Returns: None

7. Calling Sequence: CALL SYMPRT (SYM, N1, N2)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is 270_{16} Bytes.

12. Subroutine User: Various subroutines in MAGIC System

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: LØC

2. Purpose: Compute a vector subscript for an element in a matrix of specified storage mode

3. Equations and Procedures: The routine determines the type of matrix and computes the subscript accordingly.

4. Input Arguments:

I:	Row number of element
J:	Column number of element
N:	Number of rows in matrix
M:	Number of columns in matrix
MS:	Storage mode of matrix

0 General
1 Symmetric (Upper Half)
2 Diagonal

5. Output Arguments: IR - Resultant vector subscript

6. Error Returns: None

7. Calling Sequence: CALL LØC (I, J, IR, N, M, MS)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is 208_{16} Bytes.

12. Subroutine User: MPRD, TPRD, AXTRA3

13. Subroutines Required: None

14: Remarks: None

1. Subroutine Name: ELTEST

2. Purpose: Check on input variables (plug number, number of nodes, order of matrix', for a specific element.

3. Equations & Procedures: Logical "IF" statement is used to check equivalence of variables with predefined program constants.

4. Input: IPL & IPL1 - plug number & check constant
NNO & NNO1 - number of nodes & check constant
NORD & NORD1 - order of matrix & check constant

5. Output: NERR (error return)

6. Error Returns: NERR = 0 No error
NERR = 1 Plug number incorrect
NERR = 2 Number of nodes incorrect
NERR = 4 Order of matrix incorrect

7. Calling Sequence: CALL ELTEST (IPL, IPL1, NNO, NNO1, IP, IP1, NORD, NORD1, NERR)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage: Total storage required is $27A_{16}$ Bytes.

12. Subroutine User: All plugs

13. Subroutines required: None

14. Remarks: None

1. Subroutine Name: HTCO

2. Purpose: The functions of this subroutine is to rearrange the harmonic number and the harmonic coefficients for Plug 25 to generate the element stiffness matrix, the pressure load and the thermal load.

3. Equations and Procedures:

   a) Check the load control variable to determine type of loading for analysis

   b) Rearrange the harmonic number and the harmonic coefficients.

4. Input Arguments:

   IRT - Load control variable
   NL - Harmonic number for the pressure load
   NLL - Number of the harmonic for this analysis
   JAY - Harmonic numbers array for the pressure load
   PRZO - Harmonic coefficients array for the pressure load
   NLT - Harmonic number for the thermal load
   JAY - Harmonic number array for the thermal load
   PRZT - Harmonic coefficients array for the thermal load
   JA - Harmonic number constant for the pressure load
   PR - Harmonic coefficient constant for the pressure load in radial direction

   PZ - Harmonic coefficient constant for the pressure load in the axial direction

   ST - Harmonic number constant for the thermal load
   PRT - Harmonic coefficient constant for the thermal load in the radial direction

   IAI - Harmonic loop control value

5. Output Arguments:

   JA - Harmonic number constant for pressure load
   PR - Harmonic coefficient constant for pressure load in radial direction
   PZ - Harmonic coefficient constant for the pressure load in the axial direction

   JT - Harmonic number constant for the thermal load
   PRT - Harmonic coefficient constant for the thermal load in the axial direction

   PZT - Harmonic coefficient constant for the thermal load in the radial direction

6. Error Returns: None

7. Calling Sequence:
(IRT,NL,NLL,JAY,PRZO,NLT,JAT,PRZT,IIRT,JA,PR,PZ,JT,PRT,PZT,IAI)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

12. Subroutine User: Elplug

13. Subroutine Required: None

1. Subroutine Name: PLUG1

2. Purpose: To formulate the element matrices for a quadrilateral plate

3. Equations and Procedures: The following sequence of operations are necessary in order to obtain the element matrices. Equations are found in Volume I.

   A. The material and geometric properties are obtained from MAT and EXTRA respectively.

   B. From the Appendix of reference 1, the corner points defining the element are redefined to local oblique system by TRAOBQ. Provision is made to also account for different material axis orientation (due to orthotropy) or for a specific input stress direction.

   C. The following operations are performed as formulated in the appropriate equations:

      (1) Call NEWFT to form matrices necessary for thermal loadings,

      (2) Call CDELPQ to determine integrals of each zone of the quadrilateral,

      (3) The material property matrix dependent upon the stress-strain input of EXTRA (4) is coded as EM,

      (4) The strain, stress and displacement transformations are coded as TES, TESS and TW respectively,

      (5) Compute $[EG] = [TE\$]^T [EM] [TE\$]$,

      (6) Store transpose of [TE$$] into [T$AVE], [T$AVE] is then stored back into [TE$$] and inverted,

      (7) If print option equals -1, call P1PRTA for print of intermediate computations,

      (8) Initialize the thermal load, pressure, thermal stress, stress and mass matrices to zero.

   D. Membrane computations are performed in the following manner:

      1. Call CK11 to formulate the [K21S] element stiffness matrix in global system,

      2. Formulate the transformation from local to global system by forming the product [TAOM] [TOGM] [TGM] = [TMS],

3. Equations and Procedures: Continued

(3) If mass matrix is requested then
   a. Call CMMASS to form the membrane mass matrix in local systems (CMM),
   b. The mass matrix is then transformed to global systems as $[AMASS] = [TGSM]^T [CMM] [TGSM]$.

(4) If stress and/or force matrices are requested then
   a. Call C$TM to formulate the membrane stress matrix [S],
   b. Call CFMTS to formulate the membrane thermal force and stress matrices.

(5) If print controls equal -1, call PRT1 to print out intermediate matrices.

E. Flexural computations are then performed in the following manner:

(1) Call CK22 to add the flexural contributions to the stiffness matrix [K21S],

(2) Apply transformation to global system by performing $[TFM] = [TGAMB][TOGB][TGRB]$,

(3) If stress and/or force matrices are requested then
   a. If input pressure not equal to 0, call CFP to formulate the pressure matrix,
   b. The flexural contributions to the stress matrix are formulated by calling C$TF,
   c. If flexural input temperature not equal to zero, calls CFFTS to formulate the thermal force and stress matrices.

(4) If mass is requested then
   a. Call CFMASS to form the membrane mass matrix in local system [CMF],
   b. The mass matrix is transformed to global system as $[AMASS] = [TGFS]^T [CMF] [TGFS]$

(5) Again if the print option is -1, intermediate element computation printout is obtained from PRT1.

4. Input Arguments:

IPL	: Plug number
NNO	: Number of nodes (8)
XC,YC,ZC	: Coordinates of element node points
TEL	: Temperature array of element node points
PEL	: Pressures at element node points
NN	: Number of nodes
NL	: Node point numbers
KK,KN	: Control for computation of matrices (see remarks)
GPAXEL	: Grid point axes transformations
MAT	: Array containing material properties
EXTRA	: Array containing geometric properties

5. Output Arguments:

K21S	: Stiffness matrix
FTEL	: Element force matrix
S	: Stress matrix
SZALEL	: Thermal stress matrix
AMASS	: Mass matrix for dynamic analysis

6. Error Returns:

a. Standard error returns by ELPLUG (NERR)
b. Sin α = 0 indicates coordinate input data error

7. Calling Sequence:

CALL PLUG1 (IPL, NNO, XC, YC, ZC, TEL, PEL, QS, IP, NORD, NERR, NOINK, K21S, AN1, FTEL, S, SZALEL, AMASS, DAMPV, DAMPS, NRSEL, NN, NL, NMASS, NDMPV, NDMPS, NSEL, KK, KF, K8, KTS, KM, KDS, KDV, KN, JUSEL, EPSLON, SIGZER, MAT, EXTRA, GPAXEL, NDIR, NDEG, ICONT)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total storage required is 2916_{16} Bytes.

12. Subroutine User: ELPLUG

13. Subroutines Required:

ELTEST	CSTM
NEWFT	CFMTS
CDELPQ	PRTL
MINV	CK22
BCB	CFP
CK11	CSTF
MABC	CFFTS
CMMASS	CFMASS

14. Remarks:

The following is a list of control indicators for PLUG1. For all indicators shown a value of one will cause the operation to be performed and a value of zero will cause the operation to be skipped.

LT1	-	compute membrane contributions
LT2	-	compute flexural contributions
KK	-	compute element stiffness matrix
KF	-	compute element force matrix (thermal and/or pressure)
K8	-	compute element stress matrix
KTS	-	compute element thermal stress matrix
KM	-	compute element mass matrix
KDS	-	not used
KDV	-	not used
KN	-	compute element incremental stiffness matrix

1. Subroutine Name: CC21

2. Purpose: To assemble a submatrix into an assembled matrix

3. Equations and Procedures: None

4. Input Arguments:

   K : Control on positioning of elements for assembly
   NI : Constants from PLUG1
   C : elements of input matrix

5. Output Arguments:

   C21 - elements of the expanded matrix

6. Error Returns: None

7. Calling Sequence: CALL (K, NI, C, C21)

8. Input Tapes: None

9. Output: None

10. Scratch Tapes: None

11. Storage Required: NI(8,10), C(1), C21(105) and total storage is $(145)_{10}$

12. Subroutine User: CK11

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: MABC

2. Purpose: To evaluate the triple product of
[AN] = [A] [B] [C]

3. Equations and Procedures:

   a. Each row of the [A] matrix is multiplied by the corresponding column of the [B] matrix and stored in the [AM] matrix by column.

   b. Then each row of the [AM] matrix is multiplied by the corresponding column of the [C] matrix and the final product stored in the [AN] matrix by column.

4. Input Arguments:

A:	elements of [A] matrix
B:	elements of [B] matrix
C:	elements of [C] matrix
AM:	working storage
N:	number of rows in [A] matrix (order)
L:	number of rows in [B] matrix (order)
K:	number of rows in [C] matrix (order)
M:	number of columns in [C] matrix(order)
N1, M1:	dimension of [A] matrix
N2, M2:	dimension of [A] matrix
N3, M3:	dimension of [C] matrix

5. Output Arguments:

AN:	Elements of triple product matrix

6. Error Returns: None

7. Calling Sequence:

(A, B, C, AN, AM, N, L, K, M, N1, M1, N2, M2, N3, M3)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is $40A_{16}$ Bytes.

12. Subroutine User: general subroutine used by many other subroutines

13. Subroutines Required: None

14. Remarks: Standard matrix multiplication routine; but caution must be exercised when the dimensions and orders of input and output matrices are different

1. Subroutine Name: NEWFT

2. Purpose: Generate membrane and flexural thermal loads for quadrilateral thin shell in local coordinates

3. Equations and Procedures:

$$[BCT] = [F]^{-1} [CT]$$
$$[BMT] = [BCT] \{TEMM\}$$
$$[BFT] = [BCT] \{TEMF\}$$

where [F] and [CT] are geometric matrices of local coordinates

$\{TEMM\}$ = $\{TEL\ (I,1)\}$ membrane temperatures

$\{TEMF\}$ = $\{TEL\ (I,2)\}$ flexural temperatures

4. Input Arguments:

DELTM	: Average membrane temperature
DELTF	: Average flexure temperature
TEL	: Temperature array of element
R1B	: Local X coordinate of node 1
R2B	: " Y " of node 2
R3B	: " X " of node 3
R4B	: " Y " of node 4
IPRINT	: Print option
TZ	: Initial membrane temperature

5. Output Arguments:

BMT	: Membrane thermal load in local coordinates
BFT	: Flexural thermal load in local coordinates

6. Error Returns: None

7. Calling Sequence:

(DELTM, DELTF, TEL, R1B, R2B, R3B, R4B, BMT, BFT, IPRINT, TZ)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

F(3,3), BCT(3,4), CT(3,4), BMT(4,1), BFT(4,1), TEMM(4), TEMF (4), TEL(12,2), R1B(1), R2B(1), R3B(1), R4B(1)

Total Storage is (227_{10}).

12. Subroutine User: PLUG1

13. Subroutines required: MTRV, MAB

14. Remarks:

a. If print option equals -1, intermediate computations are printed out.

b. The membrane or flexural contribution is by passed if the respective thickness is 0.

1. Subroutine Name: CDELPQ

2. Purpose: To compute the integrals from equations in documentation for PLUG1 in Volume I.

3. Equations and Procedures:

$DELPQ^{j} = Cx_{j}\ Y_{j}$ where $p = 0,1,2,3,4$
$q = 0,1,2,3,4$
$j = 1,2,3,4$
C = constant

4. Input Arguments:

AJ - x distance from centroid to respective node point
BJ - y distance from centroid to respective node point

5. Output Arguments:

DELPQ - table of integrals for the 4 zones of the quadrilateral

6. Error Returns: None

7. Calling Sequence:

Call CDELPQ (AJ, BJ, DELPQ)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

DELPQ (4,5,5)
AJ (4)
BJ (4)
Total Storage is $(241)_{10}$.

12. Subroutine User: PLUG1

13. Subroutines Required: CHDEL1

14. Remarks: None

1. Subroutine Name: CHDEL1
2. Purpose: To rearrange the integrals generated by CDELPQ
3. Equations and Procedures: None
4. Input Arguments: DELPQ - integrals generated by CDELPQ
5. Output Arguments: DELPQ - rearranged integrals
6. Error Returns: None
7. Calling Sequence: CALL CHDELD1 (DELPQ)
8. Input Tapes: None
9. Output Tapes: None
10. Scratch Tapes: None
11. Storage Required: DELPQ (4,5,5)

    Total Storage is $(70)_{10}$.
12. Subroutine User: CDELPQ
13. Subroutines Required: None
14. Remarks: None

1. Subroutine Name: P1PRTA

2. Purpose: Print variables generated by PLUG1, if IPRINT equals -1.

3. Equations and Procedures: Not applicable

4. Input Arguments:

EX, EY	: Youngs modulus in X and Y directions respectively
MUXY	: Poisson's Ratio
GXY	: Shear modulus
GAMMA	: Material angle
ALPHAX, ALPHAY	: Thermal coefficients of expansion in X and Y directions
TF, TM	: Flexural and membrane thickness
RZB	: Vector normal to plane of quadrilateral element
R24	: deviation of local coordinates between points 2 and 4 of the quadrilateral
LAMDA	: Coefficient of normal vector so that element lies in a plane
R24BP	: Sum of the inplane vector and normal vector
THETA	: Angle for calculating centroid of element
E	: Column vector colinear with local geometric X, Y and Z system
TPRIME	: Transformation matrix
NL	: Node point numbers
SINAL, CØSAL	: Sine and cosine of oblique coordinate system
SINA, CØSA	: Sine and cosine for stress angles
SING, CØSG	: Sine and cosine of material angle
EM	: Coefficient matrix utilizing Hook's Law
ALPHM	: Matrix containing coefficients of thermal expansion
CØØRDL	: local coordinates
DELPQ	: table of integrals for the 4 zones of the quadrilateral
ALPHG	: Dummy
EG	: E matrix transformed
TES	: Strain transformation matrix
TW	: Displacement function transformation matrix

5. Output Arguments: None

6. Error Returns: None

7. Calling Sequence:

CALL P1PRTA (EX, EY, MUXY, GXY, GAMMA, ALPHAX, ALPHAY, TF, TM, RZB, R24B, LAMDA, R24BP, RØB, THETA, E, TPRIME, NL, ØINAL, CØSAL, SINA, CØSA, SING, CØSG, EM, ALPHM, CØØRDI, DELPQ, ALPHG, EG, TES, TW)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is $D38_{16}$ Bytes.

12. Subroutine User: PLUG1

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: CK11

2. Purpose: To generate the membrane stiffness for PLUG1, quadrilateral thin shell element

3. Equations and Procedures: The following sequence of operations takes place to formulate the membrane stiffness matrix:

(1) Call CT11 to formulate the membrane displacement coordinate transformation as TAO.
(2) Call MATI6O to invert the above matrix.
(3) Call CTOGM to form the transformation from oblique to geometric coordinates as TOGM.
(4) Generate the transformation matrix from geometric to reference system coordinates (TGRM) by calling CTGRM.
(5) If grid point axes transformations to another system other than global are to formulated, call AXTRA2 to generate the new TGRM matrix.
(6) Generate the displacement function transformation as TU.
(7) Call BCB to form the product

$$[TU]^T [EG] [TU] = [EO]$$

This matrix is then multiplied by the constant T x SINA and renamed the JPQ matrix.
(8) Generate the membrane stiffness (C matrix) by calling CC1. The C matrix is then expanded by CC21 and C21.
(9) The transformation matrix TA0 is expanded as TAOM.
(10) Call BCB to form the following products:

(a) $[K11O] = [TAOM]^T [C11] [TAOM]$
(b) $[K11G] = [TOGM]^T [K11O] [TOGM]$
(c) $[K21S] = [TGRM]^T [K11G] [TGRM]$

The final product, [K21S], is the desired membrane stiffness matrix.

4. Input Arguments:

NDIR	: Number of directions of movement for each grid point, control needed for AXTRA2
NDEG	: Number of degrees of freedom for each grid point, control needed for AXTRA2
ICONT	: Control set equal to 1 if grid point axes transformations are required from input data
GPAXEL	: The grid point axis transformation matrix
NNC	: Number of grid points (8) describing the element
NL	: Array containing the grid point numbers
EEZ	: Input on element data card for eccentricity
AJ, BJ	: Local X and Y coordinates of the element

SINA, COSA : Sine and cosine of the angle defined by the diagonals of the element between grid points 1 and 2
TPRIME : Transformation matrix
IPRINT : Print option
T : Membrane thickness
LT1 : Control set equal to 1 when membrane thickness is not zero
EG : Material properties matrix
DELPQ : Table of integrals
NI : Array for assembly purposes

5. Output Arguments:

K21S : Membrane stiffness matrix
EO : Material properties matrix
TU, TAO, TAOM, TOGM, TGRM : Transformation matrices defined in item 3 above
K11O, K11G, C11, JPQ, C21 : Intermediate matrices formed and defined in item 3 above.

6. Error Returns: None

7. Calling Sequence:

```
CALL CK11, (K21S, NDIR, NDEG, ICONT, GPAXEL, NMO, NL, EEZ,
     AJ, BJ, SINA, COSA, TPRIME, IPRINT, T, NI, LT1, EG,
     DELPQ, TAO, TAOM, TOGM, TGRM, K11O, K11G, C11, JPQ,
     C21, TU, EO, TF$, TMS, C)
```

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

NL (8), GPAXEL (3, 3, 12), AJ (1), BJ (1), TAO (8, 8), TPRIME (3,3), EG (10), EO (10), C (55), C21 (105), NI (8, 10), TU (3, 4), K11O (136), K11G (210), C11 (105), TAOM (16, 16), TOGM (16, 20), TGRM (20, 48) JPQ (10), TGRA (16, 48), DELPQ (4, 5, 5), TFS (16, 48), TMS (16, 48)
Total Storage is $(464)_{10}$

12. Subroutine User: PLUG1

13. Subroutines Required:

CT11	CTOGM	AXTRA2	CC1
MATI60	CTGRM	BCB	CC21

14 Remarks: None

1. Subroutine Name: CT11

2. Purpose: To formulate the membrane displacement coordinate transformation as [TAØ]

3. Equations and Procedures: The formulation is given in the documentation for PLUG1 in Volume I.

4. Input Arguments:

   AJ : Local X coordinates
   BJ : Local Y coordinates
   IPRINT : Print indicator

5. Output Arguments:

   TAØ : Transformation matrix

6. Error Returns: None

7. Calling Sequence: CALL CT11 (AJ, BJ, TAØ, IPRINT)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: AJ (1), BJ (1), TAØ (8,8)
    Total Storage is $(227)_{10}$.

12. Subroutine User: CK11

13. Subroutines Required: None

14. Remarks: If IPRINT equals -1, the TAØ matrix is printed.

1. Subroutine Name: MATI60

2. Purpose: Invert the TAØ matrix

3. Equations and Procedures: None

4. Input Arguments: N - order of matrix to be inverted
   A - to be inverted

5. Output Arguments: ISING - error messages
   DETR - value of determinant
   A - contains elements of the inverted matrix

6. Error Returns: ISING = 0 No error
   ISING = 1 Singular matrix

7. Calling Sequence:

   Call MATI60(N, A, ISING, DETR)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage: Total Storage required is 644_{16} Bytes.

12. Subroutine User: CK11

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: CTØGM

2. Purpose: To formulate the transformation matrix from oblique to geometric coordinates

3. Equations and Procedures: See writeup for PLUG1

4. Input Arguments:

   CØSA : Cosine and sine of the angle defined by the
   SINA diagonals of the element between grid points 1 and 2

5. Output Arguments:

   TØGM : Transformation matrix

6. Error Returns: None

7. Calling Sequence:

   CALL CTØGM (CØSA, $INA, TØGM)

8. Input tapes: None

9. Output tapes: None

10. Scratch Tapes: None

11. Storage Required:

    TØGM (16,20)
    Total Storage $(67)_{10}$

12. Subroutine User: CK11

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: CTGRM

2. Purpose: Formulate the transformation from geometric to reference system coordinates

3. Equations and Procedures: See writeup of PLUG1.

4. Input Arguments:

   NL : Node point numbers
   EEZ : Eccentricity factor
   TRIME : Transformation matrix to be expanded

5. Output Arguments:

   TGRM : Transformation matrix

6. Error Returns: None

7. Calling Sequence:

   CALL CTGRM (NL, EEZ, TPRIME, TGRM)

8. Input tapes: None

9. Output tapes: None

10. Scratch tapes: None

11. Storage Required:

    NL (1), TPRIME (3,3), TGRM (20,48),
    Total Storage is $(275)_{10}$

12. Subroutine User: CK11

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: CC1

2. Purpose: Generate the bottom half of the membrane contribution to the element stiffness matrix for the quadrilateral element

3. Equations and Procedures: Contained in documentation for quadrilateral element in Volume I.

4. Input Arguments:

   KI : Control for appropriate computation
   JPQ : Matrix containing material properties
   DELPQ : Table of integrals

5. Output Arguments:

   C : Membrane contribution to stiffness matrix

6. Error Returns: None

7. Calling Sequence: CALL CC1 (KI, JPQ, DELPQ, C)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

    DELPQ (4,5,5), C (55), JPQ (10)
    Total Storage is $(666)_{10}$.

12. Subroutine User: CK11

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: CMMASS

2. Purpose: Generate the membrane contribution to the mass matrix in local coordinates

3. Equations and Procedures: Contained in documentation for the quadrilateral element in Volume I

4. Input Arguments:

T	: Membrane thickness
DOO	: Area of each zone of quadrilateral
SINA	: Sine of angle defined by points 1 and 2 and the diagonal of the quadrilateral
DENS	: Density of the plate material

5. Output Arguments:

AMS : Membrane mass contribution

6. Error Returns: None

7. Calling Sequence: CALL CMMASS (T, DOO, SINA, DENS, AMS)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is $2CA_{16}$ Bytes.

12. Subroutine User: PLUG1

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: CSTM

2. Purpose: Evaluate the membrane stress matrix in local coordinates for the quarilateral element

3. Equations and Procedures: The following sequence of operations is performed:

   (1) Call CDM to formulate the membrane displacement derivative matrix as [DFM].
   (2) Call MAB to form [AM4] = [DFM] [TMS]
   (3) Call MAB to form [AM5] = [TU] [AM4]
   (4) Call MSB to form [AM6] = [EG] [AM5]
   (5) Call MAB to form [AM5] = [TES] [AM6]
   (6) Multiply [AM5] by the thickness and store in appropriate location of the stress matrix.

4. Input Arguments:

   R1B, R2B, R3B, R4B : Local coordinates
   TU : Displacement function transformation
   EG : Material properties matrix
   TES : Strain displacement matrix
   T : Membrane thickness
   TMS : Transformation matrix to system coordinates

5. Output Arguments:

   S : Stress matrix in system coordinates.

6. Error Returns: None

7. Calling Sequence:

```
CALL C$TM (R1B, R2B, R3B, R4B, TU, EG, TE$, T, S, TFS,
     TMS, DFM, AM4, AM5, AM6)
```

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

AM4 (4, 48), DFM (4,16), TU (3,4), EG (10), TES (3,3), S (40,48), AM5 (3,48), AM6 (3,48), TFS (16,48), TMS (16,48), Total Storage is $(176)_{10}$.

12. Subroutine User: PLUG1

13. Subroutines Required: MAB, MSB

14. Remarks: None

1. Subroutine Name: CDM

2. Purpose: To evaluate membrane displacement derivative matrix for the 4 zones of the quadrilateral

3. Equations and Procedures:

   See Writeup on PLUG 1 for equations

4. Input Arguments:

   IZ - constant for zone to be evaluated

   R1B, R2B, R3B, R4B - local coordinates of element

5. Output Arguments:

   LFM - membrane displacement displacement matrix

6. Error returns: None

7. Calling Sequence:

   Call CDM (IZ, R1B, R2B, R3B, R4B, DFM)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch tapes: None

11. Storage required:

    R1B (1), R2B (1), R3B (1), R4B (1), DFM (4, 16)
    Total Storage is $(257)_{10}$

12. Subroutine User: CSTM

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: CFMTS

2. Purpose: To evaluate the membrane thermal load and thermal stress matrices

3. Equations and Procedures:

(1) The thermal load is computed as follows:

$$\{AM1\} = [EM]\{ALPHM\}$$
$$\{AM2\} = [TES]^T\{AM1\}$$
$$\{IT\} = [TU]^T\{AM2\}$$
$$\{IT\} = T|SINA|\{IT\}, \text{ then}$$

Call CFM to formulate the thermal load FPB then

$$\{FT\} = [TMS]^T\{FPB\}$$

(2) The thermal stress matrix is computed as follows:

$$\{AM2\} = DELTM(T)\{AM2\}$$
$$\{SZLM\} = [TESS]\{AM2\}$$

The SZLM array is assembled into SZALEL.

4. Input Arguments:

EM	: Material properties matrix
ALPHM	: Coefficients of thermal expansion
TES	: Strain transformation matrix
TU	: Displacement function transformation
T	: Membrane plate thickness
SINA	: Sine of angle determined by the intersection of diagonals and grid points 1 and 2
DELPQ	: Table of integralsfor the 4 zones of the quadrilateral
BMT	: Transformation matrix
DELTM	: Membrane temperature
TESS	: Stress transformation
TMS	: Transformation to global system
WK1	: Array containing DELPQ

5. Output Arguments:

SZALEL	: Thermal stress matrix
FT	: Thermal load matrix
AM4, AM7, FPB	: Working arrays

6. Error Results: None

7. Calling Sequence: (EM, ALPHM, TES, TU, T, SINA, DELPQ, BMT, DELTM, TESS, SZALEL, FT, TFS, TMS, FPB, AM4, AM7, WK1)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage: EM (1), ALPHM (1), TES (3,3), TU (3,4), FPB (16), IT (4), DELPQ (4,5,5), FT (1), AM1 (3), AM2 (3), AM4 (4,48), AN7 (2,48), ØZLM (3), ØZALEL (1), TEØØ (3,3), TFØ (16,48), TMØ (16,48), WK1 (100)
Total Storages is $(195)_{10}$.

12. Subroutine User: PLUG1

13. Subroutines Required: MAB, MATB, CFMF

14. Remarks: None

1. Subroutine Name: CFMV

2. Purpose: To generate the membrane thermal load matrix in local coordinates

3. Equations and Procedures: Formulations are given in the documentation on the quadrilateral element in Volume I.

4. Input Arguments:

   DELC, DELPQ : Table of integrals for 4 zones of quadrilateral
   IT : Thermal vector
   BMT : Transformation matrix

5. Output Arguments:

   FPB1 : Thermal vector

6. Error Returns: None

7. Calling Sequence:

   CALL CFMV (DELC, FPB1, IT, BMT, DELPQ)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage: DELC (4,5,5), FPB1 (16), IT (4), BMT (4), FPB (16), DELPQ (4,5,5)
    Total Storage is $(310)_{10}$.

12. Subroutine User: CFMTS

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: PRT1

2. Purpose: If IPRINT equals -1, intermediate matrices generated are printed out

3. Equations and Procedures: not applicable

4. Input Argument:

   LT - Control on either membrane or flexural output

   TU, TAØ, TGAMB, TØGBM, TGRBM - transformation matrices

   FP, FT, CM, EO, IJPQ, C21, K21Ø, K21G - intermediate element matrices

   KK - Control for dynamics print

5. Output Arguments: None

6. Error Returns: None

7. Calling Sequence:

   CALL PRT (LT, TU, EO, IJPQ, C21, K21Ø, K21G, TAØ, TGAMB, TØGBM, TGRBM, KM, CM, FP, FT)

8. Input tapes: None

9. Output tapes: None

10. Scratch tapes: None

11. Storage: EO(10), IJPQ(10), C21(105), K21Ø(136). K21G(210(, TAØ(8,8), TU(3,4), TGAMB(16,16), TØGBM(16,20), TGRBM(20,48), CM(1), FP(1), ET(1)

    Total Storage is $(538)_{10}$

12. Subroutine User: PLUG 1

13. Subroutine Required: SYMPRT

14. Remarks: Matrices above defined in other writeups.

1. Subroutine Name: CK22

2. Purpose: Formulate the flexural stiffness matrix in local coordinates.

3. Equation and Procedures:

   (1) The following operations take place to formulate the transfer motion matrices

      (a) Calls CTGB to evaluate the transformation to geometric coordinates as TGAMB
      (b) Inverts TGAMB
      (c) Calls CTØGB to formulate the transformation from oblique to geometric coordinates as TØGB
      (d) Calls CTGRB to formulate the transformation from geometric to reference system coordinates as TGRBM
      (e) If grid point axes transformations are used, call AXTRA2 to revise the flexural transformation TGRB.

   (2) The flexural stiffness is then obtained by:

      (a) Formulating the rigidity as IPQ
      (b) Evaluating the [C] matrix for each zone by calling CC2
      (c) Assembling the [C] matrix for each zone into C21 by calling CC21.
      (d) Forming the following products:

$$[K22\emptyset] = [TGAMB]^T [C22][TGAMB]$$
$$[K22G] = [T\emptyset GB]^T [K22\emptyset][T\emptyset GB]$$
$$[K21\$] = [TGRB]^T [K22G][TGRB]$$

      Where [K21$] is the desired flexural stiffness-matrix

4. Input Arguments:

K21$	: Input Stiffness matrix from membrane contribution
IA$$Y	: control to add membrane plus flexural stiffness
NL	: Node points of element
NDIR	: Number of directions
NDEG	: Number of degrees of movement
ICØNT	: Control on grid point axis transformations
GPAXEL	: Grid point axis transformations
NNO	: Number of node points being transformed
AJ;BJ	: Local coordinates

TMS
TF$
AMATT
TRAØBQ
TGN
TPRIME
: Transformation matrices

$INA
COSA
: Sine and cosine of angle defined by intersection of diagonals and points 1 and 2

LT2 : Control on flexural computation
EG : Modified materials property matrix
T : Flexural plate thickness
NI : Array for assembly purposes
DELPQ : Table of integrals for 4 zones of quadrilateral

5. Output Arguments

K21$: Flexural contribution to stiffness matrix
TGAMB
TØGB
TGRB
TGRBM
: Transformation matrices

C
EO
K220
K226
C22
IPQ
C21
: Intermediate matrices

6. Error Returns: None

7. Calling Sequence:

```
CALL  CK22 (K21S, IASSY, NL, NDIR, NDEG, ICØNT, GPAXEL,
      NNO, AJ, BJ, AMAT, TRAØBQ, $INA, CØSA, TGN, TPRIME,
      LT2, TW, EG, T, NI, DELPQ, TGAMB, TØGB, TGRB,
      K22Ø, K22G, C22, IPQ, C21, TGRMB, EO, TF$, TMS, C)
```

8. Input tapes: None

9. Output tapes: None

10. Scratch tapes: None

11. Storage:

AJ (1), BJ(1), AMAT (3,4), TRAØBQ (3,3), TGN (4,2,2), TPRIME (3,3), TW (3,3), EG (10), EO (10), NI (8, 10), DELPQ (4,5,5), TGAMB (16,16), C (28). C21 (105), TGRBM (20, 48) K220 (136), K22G (210), C22 (105), TØGB (16,20), TGRB (20, 48), IPQ (10), TF$ (16, 48) TM$ (16, 48)

Total Storage is $(269)_{10}$

12. Subroutine user: PLUG1

13. Subroutines required are:

CTGB, MATI70, CTØGB, CTGRB, AXTRA2, BCB, CC2, CC21.

14. Remarks:

All formulations are given in the report for the quadrilateral thin shell element.

1. Subroutine Name: CTGB

2. Purpose: To formulate the flexural transformation matrix from local to geometric coordinates.

3. Equations and Procedures:

   (1) The TGB matrix is formulated from local coordinates
   (2) Using elements from AMAT, the lengths of the sides of each zone are computed and assembled in the TGN matrix
   (3) The TØN matrix is evaluated for the 4 zones by first storing [TRAØBQ] into [TØG] and then solving [TØN] = [TØC] [TGN]
   (4) The TGB matrix is then evaluated for the 4 zones as [TGB] = [TØN]{WX} + [TØN] {WY}
   Where {WX} and {WY} are arrays of local coordinate values for the respective zones.

4. Input Variables:

   AJ, BJ : Local coordinates
   AMAT : Transformation to local coordinates
   TRAØBQ : Transformation from local to oblique coordinates

5. Output Variables

   TGAMB, TGB, TGN : Transformation matrices

6. Error Returns: None

7. Calling Sequence:

   CALL CTGB (AJ, BJ, AMAT, TRAØBQ, TGAMB, TGB, TGN)

8. Input tapes: None

9. Output tapes: None

10. Scratch tapes: None

11. Storage:

    TØN (4,2,2), TGB (16, 16), TOG (2,2), XD (4), YD (4), WX (16), WY (16), L (4), TGAMB (16, 16), AJ (1), BJ (1), AMAT (3,4), TGN (4,2,2), TRAØBQ (3,3)
    Total storage is $(681)_{10}$

12. Subroutine User: CK22

13. Subroutines Called: None

14. Remarks:

All formulations are given in the report on the quadrilateral thin shell element.

1. Subroutine Name: MATI70

2. Purpose: To invert the [TGAMB] matrix

3. Equations: standard inverse technique where inverted matrix is stored back on top of itself.

4. Input Arguments:

   N - order of matrix = 16
   A - matrix to be inverted

5. Output Arguments

   A - inverted matrix
   I$ING - error return
   DETR - value of determinant

6. Error Return:

   IF I$ING = 1, singular matrix

7. Calling Sequence: CALL MATI70 (N, A, I$ING, DETR)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage:

    Total Storage required is 664_{16} Bytes.

12. Subroutine User: CK22

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: CTØGB

2. Purpose: formulate the flexural transformation matrix from oblique to geometric coordinates

3. Equations and procedures: The formulation is given in the report on the quadrilateral plate .

4. Input Arguments:

   SINA, CØSA - sine and cosine of the angle defined by the diagonals and points 1 and 2
   TGN - Transformation matrix

5. Output Arguments:

   TØGB - the required transformation matrix

6. Error Returns: None

7. Calling Sequence:

   CALL CTØGB (SINA, CØSA, TGN, TØGB)

8. Input Tapes: None

9. Output Tapes: None.

10. Scratch Tapes: None

11. Storage: TGN(4,2,2), TØGB(16,20)
    Total Storage is $(78)_{10}$

12. Subroutine User: CK22

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: CTGRB

2. Purpose: formulate the flexural transformation matrix from geometric to reference system coordinates.

3. Equations and Procedures:

   (1) Elements of the TPRIME matrix are first assembled into their respective positions

   (2) If any midpoints are suppressed, the contribution of the midpoints is redistributed to the respective corner points

4. Input Arguments:

   NL - node point numbers
   TGN - transformation matrix for midpoints
   TPRIME - transformation matrix to local coordinates

5. Output Arguments:

   TGRB, TGRBM - transformation from geometric to reference system coordinates

6. Error Returns: None

7. Calling Sequence:

   CALL CTGRB(NL, TGN, TPRIME, TGRBM, TGRB)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage: NL(1), TGN(4,2,2), AI(4), BI(4), TPRIME(3,3)
    TGRB(20,48), TGRBM(20,48)
    Total storage is $(345)_{10}$

12. Subroutine User: CK22

13. Subroutines Required: None

14. Remarks: Formulation is given in report on Quadrilateral Plate

1. Subroutine Name: CC2

2. Purpose: Form for the 4 zones of the quadrilateral the flexural contributions to an intermediate matrix C.

3. Equations and Procedures: The formulation is given in report on quadrilateral thin shell element

4. Input Arguments:

   K : Control on zone contribution
   IPQ : Rigidity matrix
   DELPQ : Table of integrals for the 4 zones of the quadrilateral

5. Output Arguments:

   C : Elements of the intermediate matrix

6. Error Returns: None

7. Calling Sequence:

   CALL CC2 (K, IPQ, DELPQ, C)

8. Input tapes: None

9. Output tapes: None

10. Scratch Tapes: None

11. Storage:

    Total Storage required is $5B6_{16}$ Bytes.

12. Subroutine User: CK22

13. Subroutine Required: None

14. Remarks: None

1. Subroutine Name: CFP

2. Purpose: Formulate the pressure load for the quadrilateral plate in reference system coordinates

3. Equations and Procedures:

   Call CFPB to generate the pressure load vector in reference system coordinates as FPB as defined by

$$\{FP\} = [TF\$]^T \{FPB\} .$$

4. Input Arguments:

   DELPQ : Table of integrals for the 4 zones of the quadrilateral
   P : Pressures at node points
   SINA : Sine of angle defined by intersection of diagonals and points 1 and 2 of the element
   TFB : Flexural transformation matrix

5. Output Arguments:

   FP : Pressure load vector in reference system coordinates
   FPB : Pressure load in local system.

6. Error Returns: None

7. Calling Sequence:

   Call CFP (DELPQ, P, ØINA, FP, TFS, TMS, FPB)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage:

    DELPQ (4,5,5), FP (4,8), FPB (16), TFB (16,48), TMS (16,48)
    Total Storage is $(57)_{10}$.

12. Subroutine User: CK22

13. Subroutine Required: CFPB, MATB

14. Remarks: The formulation is given in the documentation on the quadrilateral element in Volume I.

1. Subroutine Name: CFPB

2. Purpose: Formulate the pressure load in local coordinates for the quadrilateral thin shell element.

3. Equations and Procedures: Formulation is given in the report on the quadrilateral thin shell element.

4. Input Arguments:

   DELPQ : Table of intergrals for the 4 zones of the element
   P : Pressure value.
   SINA : Sine of angle defined by intersection of diagonals and points 1 and 2 of the element

5. Output Arguments:

   FPB : Pressure load in local coordinates

6. Error Returns: None

7. Calling Sequence:

   CALL CFPB (DELPQ, P, SINA, FPB)

8. Input tapes: None

9. Output tapes: None

10. Scratch tapes: None

11. Storage:

    DELPQ (4,5,5), FPB (16)
    Total Storage is $(227)_{10}$.

12. Subroutine User: CFP

13. Subroutines required: None

14. Remarks: None

1. Subroutine Name: CSTF

2. Purpose: To evaluate the flexural contribution to the stress matrix in reference system coordinates for the quadrilateral element.

3. Equations and Procedures:

(1) Call CDF to evaluate the membrane displacement derivative matrix DFM.
(2) Perform the following operations:

(a) [AM5] = [DFM] [TFS]
(b) [AM6] = [TW] [AM5]
(c) [AM5] = [EG] [AM6]
(d) [AM6] = [TE$] [AM5]

The AM6 matrix is then assembled into the stress matrix S.

(3) Call CDX to evaluate the flexural derivatives matrix DFM.
(4) After generating the G matrix, perform the following:

(a) [AM5] = [DFM] [TF$]
(b) [AM6] = [TW] [AM5]
(c) [AM5] = [EG] [AM6]
(d) [AM8] = [G] [AM5]

(5) Evaluate another G matrix and call CDFX and CDFY to formulate the flexural derivate matrix DFM.

(6) Perform the following operations:

(a) [AM5] = [DFM] [TR$]
(b) [AM6] = [TW] [AM5]
(c) [AM5] = [EG] [AM6]
(d) [AM7] = [G] [AM5]

The AM7 and AM8 matrices are then assembled into the stress matrix S.

4. Input Arguments:

T : Flexural thickness
TW, TE, TF$, TM$: Transformation matrices
EG : Material properties matrix

R1B)
R2B)
R3B) :Local coordinates
R4B)

CØSA, :Sine and cosine of angle defined by the intersection
SINA of the diagonals and points 1 and 2 of the element

5. Output Arguments:

S :Stress matrix

DFM)
AM5)
AM6) :Intermediate matrices
AM7)
AM8)

6. Error Returns: None

7. Calling Sequence:

```
CALL  C$TF (T, TW, EG, TES, R1B, R2B. R3B, R4B, CØSA, SINA,
      S, TFS, TMS, DFM, AM5, AM6, AM7, AM8)
```

8. Input tapes: None

9. Output tapes: None

10. Scratch Tapes: None

11. Storage Required:

R1B (3), R2B (3), R3B (3), R4B (3), DFM (4, 16), TW (3,3), EG (10), TES (3,3), S (40, 48), AM5 (3, 48), AM6 (3, 48), AM7 (2, 48), AM8 (2, 48), G (2, 3), TF$ (16, 48), TMS (16, 48).
Total Storage is $(446)_{10}$.

12. Subroutine User: PLUG1

13. Subroutines Required:

MAB, MSB, CDF, CDFX, CDFY

14. Remarks:

The formulations are given in the documentation on the quadrilateral element.

1. Subroutine Name: CDF

2. Purpose: To evaluate the flexure derivative matrices for the 4 zones of the quadrilateral element

3. Equations and Procedures: Formulation is given in the documentation on the quadrilateral element.

4. Input Arguments:

   IZ : Control on zone computation
   R1B, R2B, R3B, R4B : Local coordinates

5. Output Arguments:

   DFM : Flexural derivative matrix

6. Error Returns: None

7. Calling Sequence:

   CALL CDF (IZ, R1B, R2B, R3B, R4B, DFM)

8. Input tapes: None

9. Output tapes: None

10. Scratch tapes: None

11. Storage Required:

    R1B (1), R2B (1), R3B (1), R4B (1), DFM (4, 16)
    Total storage is $(271)_{10}$.

12. Subroutine User: CSTF

13. Subroutines required: None

14. Remarks: None

1. Subroutine Name: CDFX

2. Purpose: To evaluate the partial derivatives with respect to x of the flexural displacement matrix for the 4 zones of the quadrilateral element

3. Equations and Procedures: Formulation is given in the documentation on the quadrilateral element.

4. Input Arguments:

ITE : Control on constant (T1)
IZ : Control on zone computation
CØSA, SINA : Sine and cosine of angle defined by the intersection of the diagonals and points 1 and 2 of the element.

5. Output Arguments:

DFM : Flexural derivative matrix

6. Error returns: None

7. Calling Sequence:

CALL CDFX (ITE, CØSA, SINA, IZ, DFM)

8. Input tapes: None

9. Output tapes: None

10. Scratch tapes: None

11. Storage Required: DFM (4, 16)
Total Storage is $(176)_{10}$.

12. Subroutine User: CSTF

13. Subroutines required: None

14. Remarks: None

1. Subroutine Name: CDFY

2. Purpose: To evaluate the partial derivatives with respect to y of the flexural displacement derivative matrix for the 4 zones of the quadrilateral element

3. Equations and Procedures: Formulation is given in the documentation on the quadrilateral element

4. Input Arguments:

   IZ : Control on zone computation
   SINA : Sine of angle defined by the intersection of the diagonals and points 1 and 2 of the element

5. Output Arguments:

   DFM : Flexural derivative matrix

6. Error Returns: None

7. Calling Sequence:

   CALL CDFY (IZ, SINA, DFM)

8. Input tapes: None

9. Output tapes: None

10. Scratch tapes: None

11. Storage Required: Total Storage required is $2E8_{16}$ Bytes.

12. Subroutine User: CSTF

13. Subroutines required: None

14. Remarks: None

1. Subroutine Name: CFFTS

2. Purpose: To evaluate the flexural contribution to the thermal load and stress matrices for the quadrilateral element.

3. Equation and Procedures:

(1) The thermal stress is obtained by:

(a) $\{AM1\} = [EM]\{ALPHM\}$

(b) $\{AM2\} = [TES]^T \{AM1\}$

(c) $\{JT\} = [TW]^T \{AM2\}$

(d) $\{AM2\} = C4 \{AM2\}$ where C4 is a flexural constant

(e) $\{SZLF\} = [TESS] \{AM2\}$

SZLF is assembled into the thermal stress matrix SZALEL.

(2) The thermal load is obtained by:

(a) Define a flexural constant C3,

(b) $\{JT\} = C3 \times \{JT\}$,

(c) Call CFFV to formulate the thermal load in local system coordinates as $\{FPB\}$,

(d) Transform the thermal load to reference system coordinates as $[AM3] = [TFS]^T \{FPB\}$, $\{AM3\}$ is assembled into the thermal load matrix FT.

4. Input Arguments:

EM	: Material properties matrix
ALPHM	: Thermal coefficient matrix
TMS, TE$, TW, TE$$, BMT, TF$	: Transformation matrices
DELTF	: Flexural temperature
T	: Flexural thickness
SINA	: Sine of angle defined by intersection of diagonals and points 1 and 2 of the element
DELPQ	: Table of integrals for the 4 zones

5. Output Arguments:

SZALEL	: Thermal stress matrix
FT	: Thermal load matrix
FPB, AM3, WK1	: Intermediate matrices

6. Error Returns: None

7. Calling Sequence:

CALL CFFTS (EM, ALPHM, TES, TW, DELTF, T, TESS. SINA, DELPQ, BMT, SZALEL, FT, TFS, TMS, FPB, AM3, WK1)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

EM (10), ALPHM (3), TES (3,3), TW (3,3), DELPQ (4,5,5), FT (48), FPB (16), JT (3), BMT (3,1), SZLF (3), SZALEL (1), TESS (3,3), AM3 (48), AM1 (3), AM2 (3), TFS (16, 48), TMS (16, 48), WK1 (100)
Total Storage is $(222)_{10}$.

12. Subroutine User: PLUG1

13. Subroutines Required: MAB, CFFV, MATB

14. Remarks: Formulation is given in documentation on the quadrilateral element in Volume I.

1. Subroutine Name: CFFV

2. Purpose: To evaluate the flexural thermal load matrix in local system coordinates for the quadrilateral element

3. Equations and Procedures: Formulation is given in the report on the quadrilateral element in Volume I

4. Input Arguments:

   DELC, DELPQ :Table of integrals for the 4 zones of the quadrilateral
   JT :Flexural rigidity
   BMT :Transformation matrix

5. Output Arguments:

   FPB1 :Flexural load matrix in local coordinates

6. Error Results: None

7. Calling Sequence:

   CALL CFFV (DELC, FPB1, JT, BMT, DELPQ)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

    DELC (4,5,5), DELPQ (4,5,5) FPB1 (16), JT (4), BMT (4), FPB (16)
    Total Storage is $(365)_{10}$.

12. Subroutine User: CFFTS

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: CFMASS

2. Purpose: Evaluate the flexural mass matrix in local system coordinates for the quadrilateral thin shell element

3. Equations and Procedures: Formulation is given in report on the quadrilateral thin shell element.

4. Input Arguments:

T	: Flexural thickness
DØØ	: Area array of the 4 zones of the quadrilateral
SINA	: Sine of angle defined by intersection of the diagonals and points 1 and 2 of the element
DENS	: Density of element material

5. Output Arguments:

AMS	: Elements of the mass matrix in local coordinate system

6. Error Returned: None

7. Calling Sequence:

```
CALL   CFMA$$ (T, DØØ, SINA, DENS, AMS)
```

8. Input tapes: None

9. Output tapes: None

10. Scratch tapes: None

11. Storage Required:

Total Storage required is $23A_{16}$ Bytes.

12. Subroutine User: PLUG1

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: PLUG7

2. Purpcse: To formulate the element matrices for a frame element

3. Equations and Procedures: The following sequence of operations take place:

(1) Plug constants are set and checked against plug input
(2) Data is processed for:
    (a) grid points
    (b) element data such as area, inertia, etc.
(3) The length for the element and the direction cosines are determined and stored in TPRIME.
(4) Call CTS and CTCQ to formulate transformation matrices TS and TCQ. The eccentity of the element is taken into account by calling CECC and modifing the TS matrix.
(5) The transformation to systems coordinates is performed as [TCQS] = [TCQ] [TS] and if grid point axes transformations are necessary, [TCQS] is modified.
(6) The matrix [KS] is evaluated and then pre and post multiplied by [TCQS] to form the stiffness matrix as [K$EL].
(7) Dependent on the type of analysis, the incremental and mass matrices may be computed.
(8) The thermal load is set equal to zero.
(9) The stiffness matrix is rearranged into the stress matrix and the thermal stress matrix set equal to zero.
(10) If the print option is not equal to 0 calls P7PRT to print out intermediate computations.

4. Input Arguments:

IPL : Plug number
NN, NNO : Number of nodes
XC, YC, ZC : Element coordinates
TEL : Temperature array
PEL : Pressure array
QS : Initial displacements
NØRD : Order of stiffness matriy
NERR : Error return
KK, KF, K8, KM, ET, KVM, KN : Controls on element matrices to be computed

EPSIØ, SØ : Prestress and prestrain values
MAT : Material properties array
EXTRA : Element geometric properties
GPAXEL : Grid point axis transformations
NDIR, NDEG : Number of directions and degree control for grid point axis transformation
ICØNT : Control on grid point axis

5. Output Arguments:

KSEL : Stiffness matrix
GT : Gradient
FTEL : Thermal load matrix
SEL : Stress matrix
SZALEL : Thermal stress matrix
AMASS : Mass Matrix
DAMPV, DAMPS : Viscous and Structural Damping Matrices
NRSEL : Number of rows in stress matrix
NL : Node point numbers
NØINK, NMASS, NDMPV, NDMP$, N$EL : Number of elements in the stiffness, mass, viscous damping, structural damping and stress matrices

6. Error Returns: If third node point is not present, then exit. Standard tests on plug constants.

7. Calling Sequence:

```
CALL PLUG7 (IPL, NNO, XC, YC, ZC, TEL, PEL, QS, IP, NØRD,
     NERR, NØINK, K$EL, AN1, FTEL, SEL, SZALEL, AMASS,
     DAMPV, DAMPS, NRSEL, NN, NL, NMASS, NDMPV, NDMPS,
     NSEL, KK, KF, K8, KM, ET, KVM, KN, IUSEL, EPSIO, SO,
     MAT, EXTRA, GPAXEL, NDIR, NDEG, ICØNT)
```

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total storage required is $16A6_{16}$ Bytes.

12. Subroutine User: ELPLUG

13. Subroutines Required: ELTEST, CTCQ, MAB, AXTRA2, CECC, BCB, MSB, MATB, P7PRT, CTS, INCRE

14. Remarks: Formulations are given in report on Frame Element.

1. Subroutine Name: INCRE

2. Purpose: To evaluate the incremental matrix for the frame element.

3. Equations and Procedures: Formulation is given in report on Frame Element.

4. Input Arguments:

CØN1, CØN2	: Constants set equal to 1.0
L, J1	: Physical properties of element
C	: Input displacement matrix
TCQS	: Transformation matrix

5. Output Arguments:

AN1	: Element incremental stiffness matrix transformed to reference system coordinates.
AI, CI, N, AN2, AN3	: Intermediate matrices

6. Error Returns: None

7. Calling Sequence:

```
CALL INCRE (CØN1, CØN2, L, J1, AN1, AN2, C, TCQS, N, AN3,
     AI, CI)
```

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

AN1 (171), AN2 (78), N (78), AN3 (78), AI (3,5), C (1), TCQS (12,12), CI (18)

12. Subroutine User: PLUG7

13. Subroutines Required: BCB

14. Remarks: None

1. Subroutine Name: P7PRT

2. Purpose: To print out intermediate computations and matrices from the frame element

3. Equations and Procedures: Not applicable.

4. Input Arguments:

NERR	: Error test
GRI, GR4, GRT	: Gradient terms
PHI1, PHI4	: Energy terms
AMMAS	: Mass Matrix
EX, G, A, AJ1, J1, L, AIY, AIZ, EEI	: Material and geometric properties
ET	: Control on element matrix computation
RN, R1, R2, R3, RM	: Intermediate computations
TPRIME, TCQ, T$, TCQS	: Transformation matrices
AN1, AN2	: Incremental matrices
KS	: Stiffness matrix
C	: Intermediate displacement matrix
IPRINT	: Print option

5. Output Arguments: Not applicable

6. Error Returns: If node point 3 equal to zero, then exit.

7. Calling Sequence:

```
CALL  P7PRT (NERR, GR1, GR4, PHI1, PHI4, AMMAS, G, A, AJ1,
      L, AIY, AIZ, RN, R1, R2, R3, AJ, TPRIME, KS, TCQ,
      TS, TCQS, C, QS, AN2, AN1, RM, EX, EE1, PRINT, AN1
      ET)
```

8. Input tapes: None

9. Output tapes: None

10. Scratch tapes: None

11. Storage Required:

GR (1), GR4 (1), AMMAS (1), RN (1), RM (1), R1 (1), R2 (1), R3 (1), AJ (1), TPRIME (3,3), KS (1), TCQ (12,12), TS (12,12), C (1), QS (1), AN2 (1), AN1 (1), TCQS (12,12), GRT (1). Total storage is $(687)_{10}$.

12. Subroutine Used: PLUG7

13. Subroutines Required: SYMPRT

14. Remarks: None

1. Subroutine Name: CTS

2. Purpose: To evaluate the transformation matrix from local to referenced system coordinates for the frame element

3. Equations and Procedures: Formulation is given in documentation on frame element.

4. Input Arguments:

   TPRIME : Local coordinates transformation matrix

5. Output Arguments:

   TS : Required transformation matrix

6. Error Returns: None

7. Calling Sequence: CALL CTS (EE1, EE2, TS, TPRIME)

8. Input tapes: None

9. Output tapes: None

10. Scratch tapes: None

11. Storage Required: TS (12,12), TPRIME (3,3)
    Total Storage is $(105)_{10}$.

12. Subroutine User: PLUG7

13. Subroutines Required: None

14. Remarks: EE1, EE2 - Dummy arguments

1. Subroutine Name: CTCQ

2. Purpose: To formulate the transformation matrix to local system coordinates for the frame element

3. Equations and Procedures: Formulations are given in documentation on Frame Element.

4. Input Arguments:

   TCQ : Elements of input transformation
   L, L2, L3, L4, L5 : Length, Length squared, etc.

5. Output Arguments:

   TCQ : Required transformation matrix

6. Error Returns: None

7. Calling Sequence: CALL CTCQ (TCQ, L, L2, L3, L4, L5)

8. Input tapes: None

9. Output tapes: None

10. Scratch tapes: None

11. Storage required: Total Storage required is $2FC_{16}$ Bytes.

12. Subroutine User: PLUG7

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: CECC

2. Purpose: To compute modifications to the transformation matrix to account for eccentricity for the frame element

3. Computations and Procedures: Formulation is given in documentation on Frame Element.

4. Input Arguments:

   EE1, EE2: Eccentricity matrices
   TS : Transformation matrix to be modified

5. Output Arguments:

   TS : Modified transformation matrix

6. Error Returns: None

7. Calling Sequence: CALL CECC (EE1, EE2, TS)

8. Input tapes: None

9. Output tapes: None

10. Scratch tapes: None

11. Storage required: TS (12,12), EE1 (3), EE2 (3)
    Total Storage is $(146)_{10}$

12. Subroutine User: PLUG7

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: MPRD

2. Purpose: Multiply two matrices to form a resultant matrix

3. Equations and Procedures:

$$[R] = [A] \quad [B]$$

4. Input Arguments

A:	First input matrix	
B:	Second input matrix	
N:	Number of rows in A matrix	
L:	Number of columns in B	
MSA:	Control on storage mode of A	See Remarks
MSB:	Control on storage mode of B	

5. Output Arguments: R - Resultant matrix

6. Error Returns: None

7. Calling Sequence: CALL MPRD (A, B, R, N, M, MSA, MSB, L)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is $3EA_{16}$ Bytes.

12. Subroutine User: Utility subroutine

13. Subroutines Required: LØC

14. Remarks:

Storage Control of A and B matrix (MSA and MSB)

0 - General
1 - Symmetric (upper half)
2 - Diagonal

1. Subroutine Name: TPRD

2. Purpose: Transpose a matrix and postmultiply by another to form a resultant matrix.

3. Equations and Procedures

$$[R] = [A]^T [B]$$

A is not actually transposed. Instead, elements in matrix A are taken column-wise rather than row-wise for multiplication by B.

4. Input Arguments

A:	First input matrix	
B:	Second input matrix	
N:	Number of rows in A and B	
M:	Number of columns in A and rows in R	
L:	Number of columns in B and rows in R	
MSA:	Control of storage mode of A	See Remarks
MSB:	Control of storage mode of B	

5. Output Arguments: R - Resultant matrix

6. Error Returns: None

7. Calling Sequence: CALL TPRD (A, B, R, N, M, MSA, MSB, L)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is $3EA_{16}$ Bytes.

12. Subroutine User: Utility subroutine

13. Subroutines Required: LOC

14. Remarks

Storage Control of A and B Matrix (MSA and MSB)

0 - General
1 - Symmetric (upper half by columns)
2 - Diagonal

1. Subroutine Name: AI (Function)

2. Purpose: Control operation of the triangular integration package.

3. Equations and Procedures: The integration package will calculate the value of a double definite integral of the form

$$\iint r^p z^q dz\, dr \Bigg|_{r_i}^{r_j} \Bigg|_{z_{kl}}^{z_{mn}}$$

The procedure is to call a series of function subprograms dependent upon the values of p and q. The variables in the above integral are represented by the following program variables, which are defined in the input arguments section below:

Integral Variable	Corresponding Program Variable
r	R
z	Z
p	IP
q	IQ
i	I
j	J
k	K
l	L
m	M
n	N

4. Input Arguments:

I	: r coordinate subscript of i th element defining point
J	: z coordinate subscript of j th element defining point
K, L	: Slope of element line passing through the element defining point z_{kl}
M, N	: Slope of element line passing through element defining point z_{mn}
IP	: Exponent of r coordinate
IQ	: Exponent of z coordinate

R : Array containing r coordinates of element defining points

Z : Array containing z coordinates of element defining points

5. Output Argument:

AI(Function) : Result of performing the indicated integration

6. Error Return: None

7. Calling Sequence:

AI(I, J, K, L, M, N, IP, IQ, R, Z)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total storage required is $9FE_{16}$ Bytes.

12. Subroutine User: TRAIC, DPQINT

13. Subroutines Required:

AM
AK
BINT
F89
FF100
FJAB
F6219
F6211

14. Remarks: None

1. Subroutine Name: BINT

2. Purpose: Perform integration

$$\int_{r_i}^{r_j} r^{V}(a+br)^{W}\,dr$$

3. Equations and Procedures:

Expand $r^{V}(a+br)^{W}$ by binomial theorem and integrate term by term.

4. Input Arguments: I, J, A, B, IV, IW, R, Z

5. Output Arguments: BINT

6. Error Returns: None

7. Calling Sequence: BINT(I, J, A, B, IV, IW, R, Z)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is $35E_{16}$ Bytes.

12. Subroutine User: AI

13. Subroutines Required: COEF, AJ

14. Remarks: None

1. Subroutine name: AK

2. Purpose: Generate slope of line between two points of a triangle

3. Equations and Procedures:

$$AK = \left[Z(J) - Z(I)\right] / \left[R(J) - R(I)\right]$$

4. Input Arguments: I, J, R, Z

5. Output Arguments: AK

6. Error Returns: None

7. Calling Sequence: AK(I, J, R, Z)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total storage required is $18C_{16}$ Bytes.

12. Subroutine User: AI

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: AM

2. Purpose: Generate intercept of line between two points of triangle.

3. Equations and Procedures:

$$AM = \left[R(J)Z(I) - R(I)Z(J)\right] \Big/ \left[R(J) - R(I)\right]$$

4. Input Arguments: I, J, R.Z

5. Output Arguments: AM

6. Error Returns: None

7. Calling sequence: AM (I, J, R, Z)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total storage required is 198_{16} Bytes.

12. Subroutine User: AI

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: IFAC

2. Purpose: Compute N factorial

3. Equations and Procedures: N! = IFAC = n(n-1)(n-2). . . (1)

4. Input Arguments: N

5. Output Arguments: IFAC

6. Error Returns: None

7. Calling Sequence: IFAC(N)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total storage required is 176_{16} Bytes.

12. Subroutine User: FF100
F89

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: FJAB (function)

2. Purpose: To generate

$$\int \left[X^{m-1}/(a + bx) \right] dx$$

3. Equations and Procedures:

$$F = \left[(x^m \log (a+bx))/m \right] - \left[(b/m) \int (x^n/(a-bx)^n)\ dx \right]$$

evaluated at $x = x\ (I)$

4. Input Arguments: I, A, B, M, N, X

5. Output Argument: FJAB

6. Error Returns: None

7. Calling Sequence: FJAB (I, A, B, M, N, X)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total storage required is 286_{16} Bytes.

12. Subroutine User: AI

13. Subroutines Required: F89

14. Remarks: None

1. Subroutine Name: F6219 (function)
2. Purpose: To generate integral of

$$\int (\log (a + bx) / (x^{m+1}))\ dx$$

3. Equation and Procedures:

$$F = (- \log (a + bx) / (mx^{m})) + \left| \int (b/(m(a + bx)\ x^{m}))\ dx \right|$$

evaluated at x = X(I)

4. Input Arguments: I, A, B, M, N, X
5. Output Arguments: F6219
6. Error Returns: None
7. Calling Sequence: Function F6219 (I, A, B, M, N, X)
8. Input Tapes: None
9. Output Tapes: None
10. Scratch Tapes: None
11. Storage Required: Total Storage required is 286_{16} Bytes.
12. Subroutine User: AI
13. Subroutines Required: FF100
14. Remarks: None

1. Subroutine Name: F6211

2. Purpose: To generate

$$\int \left[(\log (A+BX)/X \right] dx$$

3. Equations and Procedures:

$$F = \log (A) \log (X) + \frac{BX}{A} - \frac{B^2 X^2}{4A^2} + \cdots\cdots$$

evaluated at X = X (I)

4. Input Arguments: I, A, B, X

5. Output Arguments: F6211

6. Error Returns: None

7. Calling Sequence: Function F6211 (I, A, B, X)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total storage required is $49C_{16}$ Bytes.

12. Subroutine User: AI

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: AJ (function)
2. Purpose: Generates

   for $M + 1 > 0$ $\quad \left[R(J)^M - R(I)^M\right] / (M+1)$

   for $M + 1 = 0$ $\quad \log \left[R(J)/R(I)\right]$
3. Equations and Procedures: None
4. Input Arguments: I, J, R, M
5. Output Arguments: AJ
6. Error Returns: None
7. Calling Sequence: Function AF(I, J, R, M)
8. Input Tapes: None
9. Output Tapes: None
10. Scratch Tapes: None
11. Storage Required: Total storage required is 262_{16} Bytes.
12. Subroutine User: BINT
13. Subroutines Required: None
14. Remarks: None

1. Subroutine Name: COEF

2. Purpose: Generate binomial coefficient

3. Equations and Procedures:

$$COEF = \binom{n}{r} = {}_nC_r = \frac{n!}{r!\,(n-r)!}$$

(the combination of n items taken r times)

4. Input Arguments: N,R

5. Output Arguments: COEF

6. Error Returns: None

7. Calling Sequence: COEF (N,R)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total storage required is $1FO_{16}$ Bytes.

12. Subroutine User: EINT

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: F89 (Function)

2. Purpose: To generate integral

$$\int (x^m/(a+bx)^n)\, dx$$

3. Equations and Procedures:

$$F89 = \frac{1}{b^{m+1}} \left[\sum_{s=o}^{m} \frac{m!\ (-a)^s X^{m-n-s+1}}{(m-s)!s!\ (m-n-s+1)} \right]$$

where $X = a+bx$

evaluated at x

4. Input Arguments: I, A, B, M, N, X

5. Output Arguments: F89

6. Error Returns: None

7. Calling Sequence: F89 (I, A, B, M, N, X)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total storage required is 476_{16} Bytes.

12. Subroutine User: AI

13. Subroutines Required: IFAC

14. Remarks: None

1. Subroutine Name: FF100 (function)

2. Purpose: generate

$$\int (1/(x^m X^n)\ )\ dx$$

where $X = a + bx$

3. Equations and Procedures:

$$FF100 = \frac{-1}{a^{m+n-1}} \left[\sum_{s=0}^{m+n-2} \frac{(m+n-2)!\ x^{m-s-1}(-b)^s}{(m+n-s-2)!s!\ (m-s-1)x^{m-s-1}} \right]$$

evaluated at x_i

4. Input Arguments: I, A, B, M, N, X

5. Output Arguments: FF100

6. Error Returns: None

7. Calling Sequence: FF100 (I, A, B, M, N, X)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total storage required is $4E8_{16}$ Bytes.

12. Subroutine User: F6219

13. Subroutines Required: IFAC

14. Remarks: None

1. Subroutine Name: PLUG2

2. Purpose: Control generation of element matrices for the triangular thin shell element.

3. Equations and Procedures:

a) Call subroutine ELTEST to verify input control values.
b) Call subroutine DTAPR to calculate sub-element coordinates and boundaries.
c) Call subroutine MATPR to generate material properties matrices.
d) Call subroutine NEWFT1 to apply revised thermal load formulation, if necessary.
e) Call subroutine PTBM to generate sub-element to local geometric coordinate system transformation matrix.
f) Call subroutine PTMGS to generate local geometric coordinates to reference system coordinates transformation matrix.
g) Call subroutine MAB to combine transformation matrices generated in (e) and (f) above into one matrix that will apply transformation from sub-element to reference system coordinates.
h) If grid point axes are to be applied then call subroutine AXTRA2 to appropriately modify final transformation matrix generated in (g) above.
i) Call subroutine DPQINT to evaluate the integrals over the three sub-elements.
j) Call subroutine PKM to generate the membrane contribution to the element stiffness matrix.
k) Call subroutine PMMASS to generate membrane contributions to element mass matrix.
l) Call subroutine PSTM to generate the membrane contribution to the element stress matrix.
m) Call subroutine PFMTS to generate membrane contribution to element thermal load and thermal stress matrices.
n) If requested, call subroutine APRT to print intermediate results.
o) The flexural contributions to the element matrices are then generated with the following flexure subroutines performing the same function as their membrane counterparts.

PTBF	is the	flexural	counterpart	to	PTBM
PTFGS	"	"	"	"	PTMGS
PKF	"	"	"	"	PKM
PFMASS	"	"	"	"	PMMASS
PSTF	"	"	"	"	PSTM
PFFTS	"	"	"	"	PFMTS

p) Call subroutine PFP to generate element pressure load matrix.
q) Call subroutines PNC1 and PNG1 to generate element incremental stiffness matrix (non-functional).
r) Call subroutine PLAS2 to generate plasticity premultipliers (non-functional).

4. Input Arguments:

IPL	-	internal element identification number (2)
NNO	-	number of element defining points (6)
XC	-	coordinates of element defining points
YC		
ZC		
TTL	-	temperatures at element defining point
PEL	-	pressures at element defining points
QS	-	input displacements at element defining points (not used)
IP	-	not used
NORD	-	total element degrees of freedom (36)
NOINK	-	number of storages required for element stiffness matrix (NORD * (NORD + 1)/2)
NN	-	not used
NL	-	array containing grid point numbers of element defining points
KK	-	suppression control for element stiffness matrix
KF	-	suppression control for element thermal and pressure load matrices
K8	-	suppression control for element stress matrix
KTS	-	suppression control for element thermal stress matrix
KM	-	suppression control for element mass matrix
FN	-	not used
KVM	-	not used
KN	-	suppression control for element incremental stiffness matrix
IUSEL	-	not used
EPSLON	-	input pre-strains (not used)
SIGZER	-	input pre-stresses (not used)
MAT	-	input temperature interpolated material properties
EXTRA	-	special element input
GPAXEL	-	grid point axes transformation matrices
NDIR	-	number of directions of element defining points (3)
NDEG	-	number of solution degrees of freedom (2 - translation and rotation)
ICONT	-	grid point axes indicator

5. Output Arguments:

NERR	-	error indicator
AK	-	element stiffness matrix
ANEL	-	element incremental stiffness matrix
FTEL	-	element thermal and pressure load matrix
S	-	element stress matrix
SZALEL	-	element thermal stress matrix
AMASS	-	element mass matrix

5. Output Arguments (Contd):

DAMPV	-	element viscous damping matrix
DAMPS	-	element structural damping matrix
NRSEL	-	number of rows in element stress and thermal stress matrices
NMASS	-	number of storages required for element mass matrix
NDMPV	-	number of storages required for element viscous damping matrix
NDMPS	-	number of storages required for element structural damping matrix
NSEL	-	number of storages required for element stress matrix

6. Error Returns:

If no error, then NERR is set to zero
If IPL $\neq$ 2, then NERR is set to one
If NNO $\neq$ 6, then NERR is set to two
If NORD $\neq$ 36, then NERR is set to four

7. Calling Sequence:

```
Call PLUG2(IPL,NNO,XY,YC,ZC,TTL,PEL,QS,IP,NORD,NERR,NOINK,
          AK,ANEL,FTEL,S,SZALEL,AMASS,DAMPV,DAMPS,NRSEL,NN,
          NL,NMASS,NDMPV,NDMPS,NSEL,KK,KF,K8,KTS,KM,FN,KVM,
          KN,IUSEL,EPSLON,SIGZER,MAT,EXTRA,GPAXEL,NDIR,NDEG,
          ICONT)
```

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes; None

11. Storage Required: Total Storage required is $2A78_{16}$ Bytes.

12. Subroutine User: ELPLUG

13. Subroutines Required:

ELTEST, DTAPR, MATPR, NEWFT1,PTBM, PTMGS, MAB, AXTRA2, DPQINT, MINV, PKM, PSTM, PFMTS, APRT, PTBF, PTFGS, PKF, PFP, PSTF, PFFTS, PNC1, PNG1, EPRT, PLAS2, PFMASS, PMMASS

14. Remarks: None

1. Subroutine Name: PMMASS

2. Purpose: To calculate the membrane contributions to the mass matrix for the triangular thin plate element.

3. Equations and Procedures: The weight of the element is calculated to be the area x thickness x density. This is then distributed equally to the 3 corner points.

4. Input Arguments:

T	=	thickness of element
DOO	=	area of triangle
DENS	=	density of element's material

5. Output Arguments: AMS = local mass matrix

6. Error Returns: None

7. Calling Sequence: Call PMMASS (T,DOO,SINA,DENS,AMS)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is 222_{16} Bytes.

12. Subroutine User: PLUG2

13. Subroutine Required: None

14. Remarks: None

1. Subroutine Name: PFMASS

2. Purpose: To calculate the flexural contribution to the mass matrix for the triangular thin plate element.

3. Equations and Procedures: The weight of the element is calculated to be the area x thickness x density. This is then distributed equally to the three corner points.

4. Input Arguments:

T	=	thickness of element
DOO	=	area of triangle
DENS	=	density of element's material

5. Output Arguments: AMS = local membrane mass matrix

6. Error Return: None

7. Calling Sequence: Call PFMASS (T,DOO,SINA,DENS,AMS)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is $2BA_{16}$ Bytes.

12. Subroutine User: PLUG2

13. Subroutine Required: None

14. Remarks: None

1. Subroutine Name: ASSY2

2. Purpose: Assemble membrane and flexure contributions into element stiffness matrix for triangular thin shell element

3. Equations and Procedures: The elements of the C1 matrix are summed into the C2 matrix as directed by the input array IASY.

4. Input Arguments:

   C1 : Array containing input elements to be assembled
   IASY : Array containing assembly instructions
   N1 : Order of C1

5. Output Arguments:

   C2 : Assembled matrix

6. Error Returns: None

7. Calling Sequence:

   (C2, C1, IASY, N1)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total storage required is 296_{16} Bytes.

12. Subroutine User:

    PKM
    PKF

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: DCD

2. Purpose: To evaluate the triple matrix product of a diagonal matrix D, a symmetric matrix S, and the diagonal matrix D.

3. Equations and Procedures:

$$AN_{nn} = \sum_{n} \sum_{n} D_{nn} * S_{nn} * D_{nn} \quad \text{(See remarks)}$$

4. Input Arguments:

SYM: Elements of symmetric matrix [S]
D: Elements of a diagonal matrix [D]
N: Order of [S] and [D] matrices

5. Output Arguments:

AN: Elements of matrix product

6. Error Returns: None

7. Calling Sequence:

CALL DCD (SYM, D, AN, N)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

Total Storage required is $30A_{16}$ Bytes.

12. Subroutine User: PKF, PKM

13. Subroutines Required: None

14. Remarks: The summations occur over

$$\begin{bmatrix} d_{11} & o & \cdot & \cdot & \cdot & o \\ o & d_{22} & & & & \cdot \\ \cdot & & \cdot & & & \cdot \\ \cdot & & & \cdot & & \cdot \\ \cdot & & & & \cdot & \cdot \\ o & \cdot & \cdot & \cdot & \cdot & d_{nn} \end{bmatrix} * \begin{bmatrix} S_{11} & & & \\ S_{21} & S_{22} & & \\ \cdot & \cdot & & \\ \cdot & \cdot & & \\ \cdot & \cdot & & \\ S_{n1} & S_{n2} & \cdots & S_{nn} \end{bmatrix} * \begin{bmatrix} d_{11} & o & \cdot & \cdot & \cdot & o \\ o & d_{22} & & & & \cdot \\ \cdot & & \cdot & & & \cdot \\ \cdot & & & \cdot & & \cdot \\ \cdot & & & & \cdot & \cdot \\ o & \cdot & \cdot & \cdot & \cdot & d_{nn} \end{bmatrix}$$

All redundant multiplications (i.e. those where zero elements exist in the D matrix and those where the upper elements of the S matrix would be considered) are dispensed within the program and only significant multiplications take place.

1. Subroutine Name: DTAPR

2. Purpose: Create three sub-elements and transformation matrix from system to local coordinates

3. Equations and Procedures: The sub-element coordinates are calculated from the system coordinates by generating a transformation matrix and applying it to the system coordinates array.

4. Input Arguments:

R1,R2,R3	: Reference system coordinates
E1,E2,E3,E	: Arrays containing coordinate differences
R12,R13	: Work storage
COORDS	: Reference system coordinates

5. Output Arguments:

RO	: Origin of sub-elements coordinate system
RL1.RL2,RL3	: Local sub-elements coordinates
TGS	: Transformation from reference system to local sub-element coordinates matrix
COORDL	: Local sub-elements coordinates

6. Error Returns: None

7. Calling Sequence:

(R1, R2, R3, RL1, RL2, RL3, E1, E2, E3, E, TGS, RO, R12, R13, COORDS, COORDL)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total storage required is $6EA_{16}$ Bytes.

12. Subroutine User: PLUG2

13. Subroutines Required: MAB

14. Remarks: None

1. Subroutine Name: MATPR

2. Purpose: Generate material properties matrices for triangular thin shell element

3. Equations and Procedures: The material properties matrix, EM, is generated dependent upon the formulation option selected; plane stress, plane strain or normal. The matrix angle and stress angle is determined by examining the extra element defining points. The material properties matrix is then oriented to the desired material angle and the stress angle transformation matrix is generated.

4. Input Arguments:

NL	: Array containing grid point numbers of element defining points
XC,YC,ZC	: Arrays containing reference system coordinates for element defining points
EX,EY,EZ	: Young's Moduli
GXY	: Rigidity Modulus
VXY,VZX,VYZ	: Poisson's Ratios
ALPHAX,ALPHAY	: Coefficients of thermal expansion
GAMXY	: Material angle
T	: Thickness
EXGRID	: Array containing coordinate differences for stress angle definition points
EXGRDL	: Array containing coordinate differences for material angle definition points
ALPHAM	: Not used
ALPHAG	: Not used
TGS	: Transformation matrix from reference system to sub-element coordinates
IST	: Plane strain, stress control
R1,R2,R3	: Not used
ROB	: Origin of sub-element coordinate system
RL1,RL2,RL3	: Local sub-element coordinates
EES	: Work storage
NEXGR	: Work storage
AMAT	: Local sub-element coordinates
L,M	: Work storage

5. Output Arguments:

EM	: Material properties matrix
EG	: Transformed material properties matrix (oriented to material angle)
TES	: Material angle transformation matrix
TESS	: Stress angle transformation matrix

6. Error Returns: None

7. Calling Sequence:

(NL, XC, YC, ZC, EX, EY, GXY, VXY, EZ, VZX, VYZ,
ALPHAX, ALPHAY, GAMXY, T, EM, EG, EXGRID, EXGRDL,
ALPHM, ALPHG, TGS, IST, R1, R2, R3, ROB, RL1, RL2,
RL3, EES, TES, TESS, NEXGR, AMAT, L,M)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total storage required is BEE_{16} Bytes.

12. Subroutine User: PLUG2

13. Subroutines Required:

MINV
MAB
BCB

14. Remarks: None

1. Subroutine Name: NEWFT1

2. Purpose: Generate membrane and flexural thermal loads for triangular thin shell element in local coordinates

3. Equations and Procedures:

$$BCT = F * CT$$
$$BMT = BCT * TEMM$$
$$BFT = BCT * TEMF$$

where F and CT are geometric matrices of local coordinates, and TEMM and TEMF are membrane and flexure temperatures, respectively, at the element defining points.

4. Input Arguments:

DELTM	: Average membrane temperature
DELTF	: Average flexure temperature
RL1,RL2,RL3	: Local coordinates
TZ	: T_8 for structure
F,BCT,CT	: Work storage
TEL	: Temperatures at element defining points
TEMM,TEMF,L,M	: Work storage

5. Output Arguments:

BMT	: Membrane thermal load in local coordinates
BFT	: Flexure thermal load in local coordinates

6. Error Returns: None

7. Calling Sequence:

(DELTM, DELTF, RL1, RL2, RL3, TZ, BMT, BFT, F, BCT, CT, TEL, TEMM, TEMF, L, M)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total storage required is 516_{16} Bytes.

12. Subroutine User: PLUG2

13. Subroutine Required:

MINV
MAB

14. Remarks: None

1. Subroutine Name: PTBM

2. Purpose: Generate membrane transformation matrix from sub-element to geometric coordinate system

3. Equations and Procedures: The transformation matrix is generated directly from sub-element coordinate values and inversion.

4. Input Arguments:

TGSM	: Not used
RL1,RL2,RL3	: Sub-element coordinates
L,M	: Work storage

5. Output Argument:

TBM	: Sub-element to geometric coordinate system membrane transformation matrix

6. Error Returns: None

7. Calling Sequence:

(TBM, TGSM, RL1, RL2, RL3, L, M)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total storage required is $49A_{16}$ Bytes.

12. Subroutine User: PLUG2

13. Subroutines Required: MINV

14. Remarks: None

1. Subroutine Name: PTMGS

2. Purpose: Generate geometric to reference coordinate system membrane transformation matrix

3. Equations and Procedures: The transformation matrix is generated by utilizing the TGS matrix. The effect of eccentricities and mid-point suppression is also reflected in the generation of the TGSM matrix.

4. Input Arguments:

NL	: Array containing element defining grid point numbers
EEZ	: Eccentricity
TBM	: Not used
TGS	: Reference system to sub-element transformation matrix

5. Output Arguments:

TGSM	: Geometric to reference coordinate system membrane transformation matrix

6. Error Returns: None

7. Calling Sequence:

(NL, EEZ, TBM, TGSM, TGS)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total storage required is 478_{16} Bytes.

12. Subroutine User: PLUG2

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: DPQINT

2. Purpose: Generate integrals over the three sub-elements of a triangular thin shell element

3. Equations and Procedures: The integrals are calculated by using the triangular integration package controlled by the function subprogram AI. The output values of the integrals are placed in the array DELPQ.

4. Input Arguments:

   RL1,RL2,RL3 : Sub-element coordinates
   R,Z,TEMP : Work storage

5. Output Arguments:

   DELPQ : Array containing integral values

6. Error Returns: None

7. Calling Sequence:

   (DELPQ, RL1, RL2, RL3, R, Z, TEMP)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total storage required is $7AC_{16}$ Bytes.

12. Subroutine User: PLUG2

13. Subroutines Required: AI (Function)

14. Remarks: None

1. Subroutine Name: PKM

2. Purpose: Generate membrane contribution to triangular thin shell element stiffness matrix

3. Equations and Procedures: The membrance contribution to the element stiffness matrix is formed by generating sub-element stiffness matrices, assembling them into a work area and then transforming from the work area to the reference coordinate system.

4. Input Arguments:

AK1	: Work storage
DELPQ	: Sub-element integrals
EM	: Material properties matrix
EG	: Material properties matrix oriented to material angle
TMS	: Sub-element to reference coordinate system transformation matrix
TFS	: Not used
IASEM	: Array containing assembly parameters
AD	: Work storage
CM	: Work storage
AIJ	: Work storage
IPRT	: Element print control
EX	: Not used
EY	: Not used
GXY	: Not used
VXY	: Not used
ALPHAX	: Not used
ALPHAY	: Not used
GAMXY	: Not used
T	: Membrane thickness

5. Output Argument:

AK	: Membrane contribution to element stiffness matrix

6. Error Returns: None

7. Calling Sequence:

(AK, AK1, DELPQ, EM, EG, TMS, TFS, IASEM, AD, CM, AIJ, IPRT, EX, EY, GXY, VXY, ALPHAX, ALPHAY, GAMXY, T)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total storage required is 746_{16} Bytes.

12. Subroutine User: PLUG2

13. Subroutines Required:

    SYMPRT
    DCD
    ASSY2
    BCB

14. Remarks: None

1. Subroutine Name: PSTM

2. Purpose: Generate membrane contribution to element stress matrix for the triangular thin shell element

3. Equations and Procedures: The membrane contributions to the element stress matrix are generated by first forming the stress values in local coordinates, then transforming to reference system coordinates and finally applying the stress angle transformation.

4. Input Arguments:

RL1	: Sub-element coordinates
RL2	
RL3	
TMS	: Sub-element to reference coordinate system transformation matrix
TFS	: Not used
EM	: Not used
EG	: Material properties matrix oriented to material angle
SN	: Work storage
AM1	: Work storage
AM2	: Work storage
TES	: Stress angle transformation matrix
EX	: Not used
EY	: Not used
GXY	: Not used
VXY	: Not used
ALPHAX	: Not used
ALPHAY	: Not used
GAMXY	: Not used
T	: Membrane thickness
R	: Work storage
U	: Work storage
X	: Work storage
Y	: Work storage

5. Output Arguments:

S	: Membrane contribution to element stress matrix

6. Error Returns: None

7. Calling Sequence:

   (S, RL1, RL2, RL3, TMS, TFS, EM, EG, SN, AM1, AM2,
   TES, EX, EY, GXY, VXY, ALPHAX, ALPHAY, GAMXY, T, R,
   U, X, Y)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total storage required is $61C_{16}$ Bytes.

12. Subroutine User: PLUG2

13. Subroutines Required:

    MAB
    MSB
    MATB

14. Remarks: None

1. Subroutine Name: PFMTS

2. Purpose: Generate membrane thermal load and membrane thermal stress matrices for the triangular thin shell element

3. Equations and Procedures: Subroutine PFMV1 is called to generate the thermal load matrix in geometric coordinates from BMT. This matrix is then transformed to reference system coordinates by TMS. The thermal stress matrix is generated and the stress angle applied by TESS.

4. Input Arguments:

DELTM	: Average membrane temperature
TES	: Material angle transformation matrix
TESS	: Stress angle transformation matrix
BMT	: Membrane thermal load contribution in sub-element coordinate system
EM	: Not used
EG	: Material properties matrix oriented to material angle
TMS	: Sub-element to reference coordinate system transformation matrix
TFS	: Not used
EX	: Not used
EY	: Not used
GXY	: Not used
VXY	: Not used
FMV	: Work storage
ALPHAX, ALPHAY	: Coefficients of thermal expansion
GAMXY	: Not used
T	: Membrane thickness
TO	: Not used
TI	: Not used
FME	: Work storage
EMI	: Work storage
EM1	: Work storage
SZLM	: Work storage
SZLM1	: Work storage
WRK	: Work storage
DELPQ	: Array containing sub-element integral values

5. Output Arguments:

FT	: Membrane contribution to element thermal load matrix
SZALEL	: Membrane contribution to element thermal stress matrix

6. Error Returns: None

7. Calling Sequence:

   (FT, DELTM, SZALEL, TES, TESS, BMT, EM, EG, TMS, TFS,
   EX, EY, GXY, VXY, FMV, ALPHAX, ALPHAY, GAMXY, T, TO,
   TI, FME, EMI, EM1, SZLM, SZLM1, WRK, DELPQ)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is $60D_{16}$ Bytes.

12. Subroutine User: PLUG2

13. Subroutines Required:

    PFMV1
    MAB
    MATB
    MSB

14. Remarks: None

1. Subroutine Name: PFMV1

2. Purpose: Generate membrane contribution to element thermal load matrix in local coordinates

3. Equations and Procedures: The integral values across the sub-elements are re-arranged. The membrane contribution for each sub-element is generated in FMV by direct formulation as a function of the integral values and the material properties matrix. The sub-element matrices are placed in FMV1 and pre-multipled by BMT.

4. Input Arguments:

DELC	: Array containing sub-element integral values
EG	: Material properties matrix oriented to material angle
BMT	: Array containing revised formulation for membrane thermal load matrix in local coordinates
FMV	: Work storage
T	: Membrane thickness

5. Output Arguments:

FMV1	: Membrane thermal load matrix in local coordinates
DELPQ	: Re-arranged sub-element integral values

6. Error Returns: None

7. Calling Sequence:

(FMV1, DELC, EG, BMT, FMV, DELPQ, T)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total storage required is 766_{16} Bytes.

12. Subroutine User: PFMTS

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: APRT

2. Purpose: Provide print of intermediate triangular thin shell element computations

3. Equations and Procedures: None

4. Input Arguments:

LT	: Membrane/flexure indicator
LT1	: Not used
LT2	: Not used
DELPQ	: Array containing sub-element integral values
RL1, RL2, RL3	: Sub-element coordinates
R1, R2, R3	: Reference system element coordinates
RO	: Origin of sub-element coordinate system
E1, E2, E3, E	: Sub-element coordinate differences
TGS	: Sub-element to geometric coordinates transformation matrix
TBF	: Flexure sub-element to geometric system coordinates transformation matrix
TGSF	: Flexure geometric to reference system coordinates transformation matrix
TMS	: Membrane sub-element to reference system coordinates transformation matrix
TFS	: Flexure sub-element to reference system coordinates transformation matrix
EM	: Material properties matrix
EG	: Material properties matrix oriented to material angle
TES	: Material angle transformation matrix
TBM	: Membrane sub-element to geometric coordinates transformation matrix
TGSM	: Membrane geometric to reference system coordinates transformation matrix

5. Output Arguments: None

6. Error Returns: None

7. Calling Sequence:

   (LT, LT1, LT2, DELPQ, RL1, RL2, RL3, R1, R2, R3, R0,
   E1, E2, E3, E, TGS, TBF, TGSF, TMS, TFS, EM, EG,
   TES, TBM, TGSM)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Stor ₅e Required: Total storage required is $B2C_{15}$ Bytes.

12. Subroutine User: PLUG2

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: PTFGS

2. Purpose: Generate flexure geometric to reference system coordinates transformation matrix

3. Equations and Procedures: The flexure geometric to reference system coordinates transformation matrix is generated from the TGS matrix and the sub-element coordinates. The effect of mid-point suppress contained in this transformation matrix suppression.

4. Input Arguments:

NL	: Array containing element definition grid point numbers
TGS	: Sub-element to geometric transformation matrix
TBF	: Not used
XD, YD, L, AI, BI	: Work storage
AMAT	: Array containing sub-element coordinates

5. Output Arguments:

TGSF	: Flexure geometric to reference system coordinates transformation matrix

6. Error Returns: None

7. Calling Sequence:

(NL, TGS, TBF, TGSF, XD, YD, L, AI, BI, AMAT)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total storage required is 582_{16} Bytes.

12. Subroutine User: PLUG2

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: PKF

2. Purpose: Generate the flexure contribution to the triangular thin shell element stiffness matrix

3. Equations and Procedures: The sub-element flexure contributions are generated and assembled into a work area. A transformation is then applied to the reference coordinate system.

4. Input Arguments:

IASSY	: Control indicating flexure contribution will supplement membrane contribution or flexure contribution alone is requested
DELPQ	: Array containing sub-element integrals
EM	: Not used
EG	: Material properties matrix oriented to material angle
TMS	: Not used
TFS	: Flexure sub-element to system reference coordinates transformation matrix
IASEM	: Work storage for assembly control array
AD	: Work storage
CM	: Work storage
AIJ	: Work storage
EX	: Not used
EY	: Not used
GXY	: Not used
VXY	: Not used
ALPHAX	: Not used
ALPHAY	: Not used
GAMXY	: Not used
T	: Flexure thickness
IPRT	: Intermediate results print control
AK1	: Work storage
ROW	: Work storage
ROWN	: Work storage

5. Output Argument:

AK	: Flexure contribution to element stiffness matrix

6. Error Returns: None

7. Calling Sequence:

   (AK, IASSY, DELPQ, EM, EG, TMS, TFS, IASEM, AD, CM,
   AIJ, EX, EY, GXY, VXY, ALPHAX, ALPHAY, GAMXY, T,
   IPRT, AK1, ROW, ROWN)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total storage required is 772_{16} Bytes.

12. Subroutine User: PLUG2

13. Subroutines Required:

    DCD
    ASSY2
    CCB

14. Remarks: None

1. Subroutine Name: CCB

2. Purpose: Perform triple product multiplication, A^TSA, where S is a symmetric matrix stored lower half by rows

3. Equations and Procedures: A row of the intermediate matrix product A^TS is generated at a time. From the product of this row and A, a row of the final triple product is generated.

   Options are present for scalar multiplication of the triple product, summing the triple product into an existing matrix, and deleting upper rows of the matrices from the operation.

4. Input Arguments:

A	: First input matrix, doubly dimensioned in calling program
SYM	: Second input matrix, symmetric, singly subscripted, stored lower half by rows
ND,MD	: Dimensioned size of A
N,M	: Actual size of A
N1	: Number of upper rows to be deleted in the operation
SCAL	: Scalar multiplier
IASSY	: Sum option indicator
ROW,ROWN	: Work storage

5. Output Argument:

AN	: Triple product of A^TSA, symmetric, singly subscripted, stored lower half by rows

6. Error Returns: None

7. Calling Sequence:

   (A, SYM, AN, ND, MD, N, M, N1, SCAL, IASSY, ROW, ROWN)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total storage required is $5B6_{16}$ Bytes.

12. Subroutine User: PLUG2

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: PFP

2. Purpose: Generate element pressure load matrix for the triangular thin shell element

3. Equations and Procedures: The element pressure load matrix is generated in local coordinates and then transformed to reference system coordinates.

4. Input Arguments:

TMS	: Not used
TFS	: Flexure sub-element to reference system transformation matrix
DELPQ	: Array containing sub-element integral values
P	: Pressures at element definition points
FPB	: Work storage

5. Output Arguments:

FP	: Element pressure load matrix

6. Error Returns: None

7. Calling Sequence:

(FP, TMS, TFS, DELPQ, P, FPB)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

Total storage required is $2A2_{16}$ bytes.

12. Subroutine User: PLUG2

13. Subroutines Required: MATB

14. Remarks: None

1. Subroutine Name: PSTF

2. Purpose: Generate flexure contribution to element stress matrix for the triangular thin shell element

3. Equations and Procedures: The sub-element stress matrices are generated and assembled into one matrix. This matrix is then transformed to reference system coordinates and the stress angle is applied.

4. Input Arguments:

RL1 RL2 RL3	: Sub-element coordinates
TMS	: Not used
TFS	: Flexure sub-element to reference system coordinates transformation matrix
EM	: Not used
EG	: Material properties matrix, oriented the material angle
SNM	: Work storage
TES	: Stress angle transformation matrix
EX	: Not used
EY	: Not used
GXY	: Not used
VXY	: Not used
ALPHAX	: Not used
ALPHAY	: Not used
GAMXY	: Not used
T	: Flexure thickness
R	: Not used
U	: Not used
X	: Work storage
Y	: Work storage
AM1	: Work storage
AM2	: Work storage
AM3	: Work storage
AM4	: Work storage
G	: Work storage

5. Output Argument:

S	: Flexure contribution to element stress matrix

6. Error Returns: None

7. Calling Sequence:

```
(S, RL1, RL2, RL3, TMS, TFS, EM, EG, SNM, TES, EX, EY,
 GXY, VXY, ALPHAX, ALPHAY, GAMXY, T, R, U, X, Y, AM1,
 AM2, AM3, AM4, G)
```

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total storage required is $BD4_{16}$ Bytes.

12. Subroutine User: PLUG2

13. Subroutines Required:

    MAB
    MSB
    MTB

14. Remarks: None

1. Subroutine Name: PFFTS

2. Purpose: Generate flexure contribution to element thermal load and thermal stress matrices for the triangular thin shell element

3. Equations and Procedures: The flexure contribution to the element thermal load matrix in local coordinates is generated by calling subroutine PFFV1. The material angle transformation is applied and the transformation from local to reference system coordinates is performed.

   The flexure contribution to the element thermal stress matrix is generated and transformed to the selected stress angle.

4. Input Arguments:

DELTF	: Average flexural temperature
TES	: Material angle transformation matrix
TESS	: Stress angle transformation matrix
BFT	: Flexural thermal load formulation revision
EM	: Not used
EG	: Material properties matrix, oriented to material angle
TMS	: Not used
TFS	: Flexure sub-element to reference system coordinates transformation matrix
EX	: Not used
EY	: Not used
GXY	: Not used
VXY	: Not used
FFV	: Not used
ALPHAX	: Not used
ALPHAY	: Not used
GAMXY	: Not used
T	: Flexure thickness
TO	: Not used
T1	: Not used
EFI	: Work storage
FFE	: Work storage
FF	: Work storage
SZLF	: Work storage
SZLF1	: Work storage
EF1	: Work storage
WRK	: Work storage
DELPQ	: Array containing sub-element integrals

5. Output Arguments:

   FT : Flexure contribution to element thermal load matrix
   SZALEL : Flexure contribution to element thermal stress matrix

6. Error Returns: None

7. Calling Sequence:

   (FT, DELTF, SZALEL, TES, TESS, BFT, EM, EG, TMS, TFS, EX, EY, GXY, VXY, FFV, ALPHAY, ALPHAY, GAMXY, T, TO, TI, EFI, FFE, FF, SZLF, SZLF1, EF1, WRK, DELPQ)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total storage required is 656_{16} Bytes.

12. Subroutine User: PLUG2

13. Subroutine Required: PFFV1

    PFFV1
    MAB
    MATB
    MSB

14. Remarks: None

1. Subroutine Name: PFFV1

2. Purpose: Generate flexure contribution to element thermal load matrix in local coordinates

3. Equations and Procedures: The array containing the subelement integral values is re-arranged. The sub-element thermal load matrices are generated from the integral values and the material properties matrix. The sub-element thermal load matrices are assembled into one matrix and then multiplied by BFT to apply the revised thermal load formulation.

4. Input Arguments:

DELC	: Array containing sub-element integral values
EG	: Material properties matrix, oriented to material angle
BFT	: Array containing revised thermal load formulation
FFV	: Work storage

5. Output Arguments:

FFV1	: Flexure contribution to element thermal load matrix in local coordinates
DELPQ	: Array containing re-arranged sub-element integral values

6. Error Returns: None

7. Calling Sequence:

(FFV1, DELC, EG, BFT, FFV, DELPQ)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total storage required is $9A0_{16}$ Bytes.

12. Subroutine User: PFFTS

13. Subroutine Required: None

14. Remarks: None

1. Subroutine Name: PNC1
2. Purpose: Non-functional
3. Equations and Procedures: None
4. Input Arguments: None
5. Output Arguments: None
6. Error Returns: None
7. Calling Sequence: None
8. Input Tapes: None
9. Output Tapes: None
10. Scratch Tapes: None
11. Storage Required: Total storage required is $F6_{16}$ Bytes.
12. Subroutine User: PLUG2
13. Subroutine Required: None
14. Remarks: None

1. Subroutine Name: PNG1
2. Purpose: Non-functional
3. Equations and Procedures: None
4. Input Arguments: None
5. Output Arguments: None
6. Error Returns: None
7. Calling Sequence: None
8. Input Tapes: None
9. Output Tapes: None
10. Scratch Tapes: None
11. Storage Required: Total storage required is $F6_{16}$ Bytes.
12. Subroutine User: PLUG2
13. Subroutines Required: None
14. Remarks: None

1. Subroutine Name: EPRT

2. Purpose: Print generated triangular thin shell element matrices

3. Equations and Procedures: None

4. Input Arguments:

AK	: Final element stiffness matrix
S	: Final element stress matrix
ANEL	: Non-functional
FN	: Non-functional
FT	: Final element thermal load matrix
FP	: Final element pressure load matrix
SZALEL	: Final element thermal stress matrix

5. Output Arguments: None

6. Error Returns: None

7. Calling Sequence:

(AK, S. ANEL, FN, FT, FP, SZALEL)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total storage required is $4FO_{16}$ Bytes.

12. Subroutine User: PLUG2

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: PLAS2D
2. Purpose: Non-functional
3. Equations and Procedures: None
4. Input Arguments: None
5. Output Arguments: None
6. Error Returns: None
7. Calling Sequence: None
8. Input Tapes: None
9. Output Tapes: None
10. Scratch Tapes: None
11. Storage Required: Total storage required is $F6_{16}$ Bytes.
12. Subroutine User: PLUG2
13. Subroutines Required: None
14. Remarks: None

1. Subroutine Name: PTBF

2. Purpose: Generate flexure sub-element to geometric axes transformation matrix

3. Equations and Procedures: The inverse of the desired matrix is generated by direct assignment into a work area. Inversion is performed to obtain the final transformation matrix.

4. Input Arguments:

TGSF	: Not used
RL1 RL2 RL3	: Sub-element coordinates
IPRT	: Intermediate element print indicator
L	: Work storage
M	: Work storage
U	: Work storage
TI	: Work storage
B	: Work storage
BFF	: Work storage
BFO	: Work storage

5. Output Arguments: None

TBF	: Flexure sub-element to geometric transformation matrix

6. Error Returns: None

7. Calling Sequence:

(TBF, TGSF, RL1, RL2, RL3, IPRT, L, M, U, TI, B, BFF, BFO)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total storage required is CCC_{16} Bytes.

12. Subroutine User: PLUG2

13. Subroutines Required:

MAB
MINV

14. Remarks: None

1. Subroutine Name: PLUG 6

2. Purpose: To form the element matrices for a triangular cross section ring discrete element with applications towards the analysis of thick walled and solid axisymmetric structures of finite length. It may be used to form the assembly of any axisymmetric structure taking into account:

(1) Arbitrary axial variations in geometry
(2) Axial variation in orientation of material axes of orthotropy
(3) Radial and axial variations in material properties
(4) Any axisymmetric loading systems including pressure, prestrain, prestress, and temperature

The complete discrete element representation, consists of the algebraic expressions for the following matrices:

(1) Stiffness
(2) Pressure load
(3) Thermal load
(4) Pre-strain load
(5) Pre-stress load
(6) Stress
(7) Mass
(8) Structural damping
(9) Viscous damping

3. Equations and Procedures: The development of the complete element representation arises from the Lagrangian (variational) equation

$$\frac{\partial \phi_1}{\partial q_r} + i \frac{\partial \phi_2}{\partial q_r} + \frac{\partial \phi_3}{\partial q_r} + \frac{d}{dt}\left(\frac{\partial \phi_4}{\partial q_r}\right) = 0$$

where

q_r = r generalized displacement coordinates
ϕ_1 = total potential energy
ϕ_2 = structural damping dissipation energy
ϕ_3 = viscous
ϕ_4 = kinetic energy

The subsequent development of the element matrices is then provided in algebraic form to the coded program, which follows the format:

(1) The input data, used in forming the matrices, is processed and organized for computation.

(2) By subroutine TRAIC, the coordinate transformation matrices, and the table of integrals is formed. In routine TRAIE, the material properties matrices are formed.

(3) Using the above mentioned matrices and integrals, the program then generates, the stiffness, pressure load, thermal load, stress, pre-strain, pre-stress, mass, structural damping, and viscous damping matrices, and the stress, thermal stress, pre-strain, pre-stress, pre-strain load, and pre-stress load vectors.

(4) After each significant matrix, vector, etc., is formed, the program prints out the desired results.

4. Input Arguments:

IPL	: Plug number
NNØ	: Number of node points
XC	: X - coordinates of nodes points
YC	: Y - coordinates of node points
ZC	: Z - coordinates of node points
TEL	: Temperatures at the node points
PEL	: Pressures at the node points
Q$EL	: Input displacements of the node points
IP	: Number of extra cards
NØRD	: Order of element stiffness matrix
NR$EL	: Number of rows in the stress matrix
INØ	: Number of nodes
NØDØRD	: Node point numbers
KK	: Code for computation of element stiffness matrix
KF	: Code for computation of element thermal load
K$	: Code for computation of element stress matrix
KM	: Code for computation of element mass matrix
KD$	: Code for computation of structural damping matrix
KDV	: Code for computation of viscous damping matrix
KN	: Code for computation of incremental damping matrix
IU$EL	: Dummy
EP$LØN	: Pre-strain load vector
$IGZER	: Pre-stress load vector
MAT	: Material properties matrix
EXTRA	: Extra information (angles, etc.)
NDIR	: Number of directions of movement per grid point
NDEG	: Number of types of movement allowed per grid point
ICØNT	: Code for use of grid point axes

5. Output Arguments:

NERR	:	Error return
NØINK	:	Number of elements in lower half matrices
AKELXP	:	Stiffness matrix
ANEL	:	Incremental stiffness matrix
FTXP	:	Thermal load + pressure load matrix
TRXP	:	Stress matrix
T$	:	Thermal stress matrix
XMA$XP	:	Mass matrix
DAMPV	:	Viscous damping matrix
DAMP$	:	Structural damping matrix
N$EL	:	Number of elements in stress matrix
NMA$$	:	Number of elements in mass matrix
NP$L	:	Number of elements in viscous damping matrix
NP$$	:	Number of elements in structural damping matrix
GPAXEL	:	Grid point axes transformation

6. Error Returns

NERR	=	0	No Error
	=	1	Plug Number Incorrect
	=	2	Number of Nodes Incorrect
	=	3	Number of Input Points Incorrect
	=	4	Order of Matrix (nord) Incorrect

7. Culling Sequence:

(IPL, NNØ, XC, YC, ZC, TEL, PEL, Q$EL, IP, NØRD, NERR, NØINK,
AKELXP, ANEL, FTXP, TRXP, T$, XMA$XP, DAMPV, DAMP$,
NR$EL, INØ, NØDØRD, NMA$$, NP$L, NP$$, NEL, KK, KF, K,
KM, KD$, KDV, KN, IU$EL, EP$LØN, $IGZER, MAT, EXTRA,
GPAXEL, NDIR, NDEG, ICØNT)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage:

GAMABQ(6,6) DELINT(12) DCURL(4,6) EM(10) E(10) TEØ(4,4)
AKEL(21) FP(6) F$(6) F$T(6) EP$LØN(6) $IGZER(6) EXTRA(1)
ALFBAR(4) FT(6) $TRE$$(4,6) T$(4) XMA$$(21) D$M(4)
D$(21) DV(21) AKELXP (45) XMA$XP (45) D$XP(45) DVXP(45)
XC(3) YC(3) ZC(3) NODORD(3) X(3) Y(3) Z(3) P$LMAT (6,4)
PEL (6) Q$EL(6) ANEL(6) TEMP2(6,6) IU$EL (6) LI$TP(6)
TEL (12,3) E1 (4,4) DAMP$(6) DAMPV(6) A(3,6) B(4,6)
ALI$TP (6) FTXP(9) TRXP (4,9) P$LXP (9,4) P$$(4)
MAT(1) AKEL1 (6,6) AKEL2 (6,6) AMCURL (21) TEMP (6)
TEMP1 (4) XMA$$1 (6,6) TMG (2,2) AMCURL (21) AMBAR (2,2)
DZERØ (6,4)

12. Subroutine User: ELPLUG

13. Subroutines Required:

ELTE$T	TRAIE	TRAIK	TRAIFP	TFTPRT
TRAIT$	TRAIM	TF$PRT	EXPCØL	TRAIC
TIEPRT	TIKPRT	TFPPRT	TRAI$	TT$PRT
TIMPRT	TRAI$T	PL6PRT	TRCPRT	TPRD
EXP$IX	TRAIFT	TI$PRT	MPRD	TRAIF$
T$TPRT				

14. Remarks: None

1. Subroutine Name: EXPCØL

2. Purpose: To generate a matrix [B] , given a specific input matrix [A] , for Plug 6.

   The purpose of this operation is to impose the conditions that flexure terms "v" are zero.

3. Equations and Procedures: The matrix terms are formed by direct assignment.

4. Input Arguments: [A] : Input Matrix

5. Output Arguments: [B] : Output Matrix

6. Error Returns: None

7. Calling Sequence: (A, B)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is 166_{16} Bytes.

12. Subroutine User: PLUG 6

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: EXP$IX

2. Purpose: To generate a symmetric matrix [B], given a specific input symmetric matrix [A], for Plug 6. The purpose of this operation is to impose the condition that flexure terms "v" are zero.

3. Equations and Procedures: The matrix terms are formed by direct assignment.

4. Input Arguments: [A] : Input Matrix

5. Output Arguments: [B] : Output Matrix

6. Error Returns: None

7. Calling Sequence: (A, B)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is 206_{16} Bytes.

12. Subroutine User: Plug 6

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: TRAIC

2. Purpose: To generate coordinate transformation matrices for triangular ring which vary with coordinates and generate integrals for future use.

3. Equations and Procedures: The coordinate matrix [GAMABQ] is formed by algebraic assignment. The table of integrals, DELINT, is formed by algebraic methods using the function subprogram AI.

4. Input Arguments: R,Z: Coordinates of node points
WIPR: Print control

5. Output Arguments:

GAMABQ: Coordinate matrix
DELINT: Table of integrals
DCURL: Matrix transformation
ISING: Error return code

6. Error Returns: If GAMABQ cannot be generated due to singular matrix then ISING is set to one.

7. Calling Sequence: (R, Z, GAMABQ, DELINT, DCURL, ISING, WIPR)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: R(3), Z(3), GAMABQ (6,6), DELINT (12), DCURL (4,6), LL(6), MM(6)

12. Subroutine User: PLUG6

13. Subroutines Required: MINV, AI, TESTJ, TRCPRT

14. Remarks: None

1. Subroutine Name: TESTJ

2. Purpose: To check DELINT (PLUG6) for any negative or incorrect integrals; If any errors are noted, the integrals are recomputed by an approximation method.

3. Equations and Procedures: The checks are performed by logical if statements. The integral approximation is

$$\iint X^P Z^Q \, d\,x\, dZ \approx \bar{X}^P \cdot \bar{Z}^Q \cdot A$$

where

$$\bar{X} = \frac{1}{3}\left[X_1 + X_2 + X_3\right]; \quad \bar{Z} = \frac{1}{3}\left[Z_1 + Z_2 + Z_3\right]$$

$$A = \frac{1}{2}\left[X_1(Z_2-Z_3) + X_2(Z_3-Z_1) + X_3(Z_1-Z_2)\right]$$

4. Input Arguments: DELINT (I) value of the i th integral
X: X coordinates
Z: Z coordinates
WIPR: print control

5. Output Arguments: DELINT (I): recomputed integral

6. Error Returns: None

7. Calling Sequence: CALL TESTJ (DELINT, X, Z, WIPR)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

DELINT (12), DLINT (12), X (1), Z (1), XO (3), ZO (3), DELTAX (1), DELTAZ (1), XHAT (1), ZHAT (1)

12. Subroutine User: TRAIC

13. Subroutine Required: None

14. Remarks: If the test necessitates recomputation, the new integral values will be stored in the old locations, thus destroying the originals.

1. Subroutine Name: TRCPRT

2. Purpose: To print elements formed in TRAIC

3. Equations and Procedures: None

4. Input Arguments:

   GAMABQ: coordinate matrix
   DELINT: table of integrals
   DCURL : matrix of integrals

5. Output Arguments: None

6. Error Returns: None

7. Calling Sequence: (GAMABQ, DELINT, DCURL)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is 320_{16} Bytes.

12. Subroutine User: PLUG6

13. Subroutine Required: None

14. Remarks: None

1. Subroutine Name: TRAIE

2. Purpose: To generate the transformed matrix of elastic constants

3. Equations and Procedures: The routine

a) Generates the transformation matrix
b) Generates the elastic constants matrix
c) Generates the transformed elastic constant matrix

4. Input Arguments:

ER, ETHETA, EZ	: Moduli of elasticity (Young's)
VRØ, VØZ, VZR	: Poissons ratio
GRZ	: Modulus of rigidity
GAM	: Angle between material axes and element axes
E1	: Work storage

5. Output Arguments:

TEØ	: Transformation matrix
EM	: Elastic constants matrix
E	: Transformed elastic constant matrix

6. Error Returns: None

7. Calling Sequence:

(ER, ETHETA, EZ, VRØ, VØZ, VZR, GRZ, GAM, TEØ, EM, E, E1, WIPR)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required. Total Storage required is 602_{16} Bytes.

12. Subroutine User: PLUG6

13. Subroutines Required: MPRD, TPRD

14. Remarks: None

1. Subroutine Name: TIEPRT

2. Purpose: To print matrices formed in TRAIE

3. Equations and Procedures: None

4. Input Arguments: TEØ : Transformation matrix
EM : Elastic constant matrix
E : Transformed elastic constant matrix

5. Output Arguments: None

6. Error Returns: None

7. Calling Sequence: (TEØ, EM, E)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is 294_{16} Bytes.

12. Subroutine User: PLUG6

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: TRAIK

2. Purpose: Generate stiffness matrix for triangular ring

3. Equations and Procedures: The program uses the table of integrals to form the first intermediate matrix. This matrix is then transformed to form the final stiffness matrix.

4. Input Arguments:

GAMABQ: Transformation matrix
E : Transformed elastic constant matrix
DELINT: Table of integrals
WIPR : Print control
AKEL1, AKEL2, ACURL: Work storage

5. Output Arguments: AKEL : Stiffness matrix

6. Error Returns: None

7. Calling Sequence: (GAMABQ, E, DELINT, AKEL, WIPR, AKEL1, AKEL2, ACURL)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

GAMABQ (6,6), E(10), DELINT (12), AKEL (21), AKEL1 (6,6), AKEL2 (6,6), ACURL(21)

12. Subroutine User: PLUG6

13. Subroutines Required: TPRD, MPRD

14. Remarks: None

1. Subroutine Name: TIKPRT

2. Purpose: To display matrices generated in TRAIK

3. Equations and Procedures: None

4. Input Arguments: AKEL : Stiffness matrix
   ACURL : Intermediate stiffness matrix

5. Output Arguments: None

6. Error Returns: None

7. Calling Sequence: (AKEL, ACURL)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is $1E8_{16}$ Bytes.

12. Subroutine User: PLUG6

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: TRAIFP

2. Purpose: To generate the pressure load vector for triangular ring.

3. Equations and Procedures: The program

   1.) Generates necessary constants
   2.) Generates pressure load vector (non-transformed)
   3.) Transforms pressure load vector

4. Input Arguments:

R,Z:	Coordinates of node points
P:	Node point pressures
GAMABQ:	Coordinate transformation matrix
WIPR:	Print control

5. Output Arguments:

FCURLP:	Non-transformed pressure load vector
FP:	Transformed pressure load vector

6. Error Returns: None

7. Calling Sequence:

   (R, Z, P, GAMABQ, FP, WIPR, FCURLP)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: R(3),Z(3),P(3),GAMABQ(6,6),FP(6), F(3),FCURLP(6),DELTA(6)

12. Subroutine User: PLUG6

13. Subroutines Required: TPRD

14. Remarks: None

1. Subroutine Name: TFPPRT

2. Purpose: To display the non-transformed and transformed pressure load vectors.

3. Equations: None

4. Input arguments:

   FP: transformed pressure load vector
   FCURLP: non-transformed pressure load vector

5. Output arguments: None

6. Error returns: None

7. Calling sequence: (FP, FCURLP)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage: Total Storage required is $1E8_{16}$ Bytes

12. Subroutine User: PLUG6

13. Subroutines required: None

14. Remarks: None

1. Subroutine Name: TRAIFT

2. Purpose: To generate a thermal load vector for a triangular ring element

3. Equations & Procedures: The input matrices are manipulated by matrix algebra to form the thermal load vector.

4. Input arguments:

ALFBAR:	vector of coefficients of linear thermal expansion
TMTZRO:	base temperature
GAMABQ:	transformation matrix
DCURL:	matrix containing integral values
E:	transformed elastic constant matrix
WIPR:	print control

5. Output arguments:

FT: thermal load vector

6. Error returns: None

7. Calling sequence:

(ALFBAR, TMTZRO, GAMABQ, DCURL, E, FT, WIPR)

8. Input tapes: None

9. Output tapes: None

10. Scratch tapes: None

11. Storage: ALFBAR(4), GAMABQ(6,6), DCURL(4,6), E(10), FT(6), TEMP1(4), TEMP2(6), SAVE(4)

12. Subroutine User: PLUG6

13. Subroutines Used: MPRD, TPRD

14. Remarks: None

1. Subroutine Name: TFTPRT

2. Purpose: To display thermal load vector for triangular ring element

3. Equations: None

4. Input arguments:

FT:	thermal load vector
ALFBAR:	coefficients of linear expansion
TMTZRØ:	base temperature

5. Output arguments: None

6. Error Returns: None

7. Calling Sequence: (FT, ALFBAR, TMTZRØ)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is $25C_{16}$ Bytes.

12. Subroutine User: PLUG6

13. Subroutines Used: None

14. Remarks: None

1. Subroutine Name: TRAI$

2. Purpose: To generate the stress matrix for triangular ring element

3. Equations and Procedures: Given input constants, an intermediate matrix is formed, which is then multiplied by the system matrices to form the final matrix

4. Input Arguments:

R, Z:	coordinates of node points
GAMABQ:	coordinate transformation matrix
E:	elastic constant matrix
WIPR:	print control
DZERØ:	work space
TEMP:	node point temperatures

5. Output Arguments: $TRE$$: stress matrix

6. Error returns: None

7. Calling sequence: (R, Z, GAMABQ, E, STRESS, WIPR, DZERO, TEMP)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage: R(3) Z(3), GAMABQ(6,6), E(10), STRESS(4,6), DZERØ(4,6), TEMP(4,6)

12. Subroutine User: PLUG6

13. Subroutines Used: MPRD

14. Remarks: None

1. Subroutine Name: TI$PRT

2. Purpose: To display the stress matrix for a triangular ring element

3. Equations: None

4. Input Arguments:

   $TRE$$: stress matrix

5. Output Arguments: None

6. Error Returns: None

7. Calling Sequence: ($TRE$$)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage: Total Storage required is $1FC_{16}$ Bytes.

12. Subroutine User: PLUG6

13. Subroutines Used: None

14. Remarks: None

1. Subroutine Name: TRAIT$

2. Purpose: To generate thermal stress vector for a triangular ring element

3. Equations and Procedures: The input matrices are combined, using matrix algebra, to form the thermal stress vector.

4. Input Arguments:

E:	elastic constant matrix
ALFBAR:	linear thermal expansion coefficients
TMTZRØ:	base temperature
WIPR:	print control

5. Output Arguments:

T$: thermal stress matrix

6. Error Returns: None

7. Calling Sequence: (E, ALFBAR, TMTZRØ, T$, WIPR)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage:

E(10), ALFBAR(4), T$(4), SAVE(4)

12. Subroutine User: PLUG6

13. Subroutines Used: MPRD

14. Remarks: None

1. Subroutine Name: TT$PRT

2. Purpose: to display the thermal stress vector of a triangular ring element

3. Equations: None

4. Input Arguments:

   T$: thermal stress vector

5. Output Arguments: None

6. Error Returns: None

7. Calling Sequence: (T$)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage: Total Storage required is $1CO_{16}$ Bytes

12. Subroutine User: PLUG6

13. Subroutines Used: None

14. Remarks: None

1. Subroutine Name: TRAIM

2. Purpose: To generate a mass matrix for a triangular ring element

3. Equations and Procedures: The program

   (1) Forms a transformation matrix [TMG]
   (2) Generates a matrix $[\bar{M}]$ which is a function of the mass coefficients.
   (3) Generates a matrix $[\tilde{M}]$ which is a function of $[\bar{M}]$ and the table of integrals.
   (4) Generates the mass matrix [M] which is a function of $[\tilde{M}]$ and the transformation matrix [GAMABQ] .

$$[M] = [GAMABQ]^T [\tilde{M}] [GAMABQ]$$

4. Input Arguments:

AMASS1, AMASS2: mass coefficients
GAM: angle between material axes and element axes
GAMABQ: coordinate transformation matrix
DELINT: table of integrals
WIPR: print control
XMASS1, TEMP, AMCRUL, TEMP1, AMBAR: storage
TMG: transformation matrix

5. Output Arguments:

XMASS: mass matrix

6. Error Return: None

7. Calling Sequence:

(AMASS1, AMASS2, GAM, GAMABQ, DELINT, XMA$$, WIPR, XMASS1, TEMP, TMG, AMCURL, TEMP1, AMBAR)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage:
AMASS(2), GAMABQ (6,6), DELINT(12), XMASS(21), XMASS1(6,6), TEMP(6,6), TMG(2,2), AMBAR(2,2), TEMP1(2,2), AMCURL(21)

12. Subroutine User: PLUG6

13. Subroutines Used: TPRD, MPRD

14. Remarks: None

1. Subroutine Name: TIMPRT

2. Purpose: To display the mass matrix of a triangular ring element

3. Equations: None

4. Input Arguments:

   XMASS: mass matrix
   AMCURL: intermediate mass matrix

5. Output Arguments: None

6. Error Returns: None

7. Calling Sequence: (XMA$$, AMCURL)

8. Output Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage:

    Total Storage required is $1E8_{16}$ Bytes.

12. Subroutine User: PLUG6

13. Subroutines Used: None

14. Remarks: None

1. Subroutine Name: TRAIF$

2. Purpose : To generate pre-strain load vector for a triangular ring element.

3. Equations and Procedures: The routine uses the inputed matrices and combines these to form the pre-strain load vector.

4. Input Arguments:

EP$LØN:	Input pre-strain values
GAMABQ:	Transformation matrix
DCURL:	Integral matrix
E:	Elastic constant matrix
WIPR:	Print control
TEMP:	Dummy storage
TEMP1, TEMP2:	Dummy storage
P$LMAT:	Dummy storage

5. Output Arguments:

F$: Pre-strain load vector

6. Error Returns: None

7. Calling Sequence:

(EP$LØN, GAMABQ, DCURL, E, F$, WIPR, TEMP, TEMP1, TEMP2, P$LMAT)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

EPSLON(4), GAMABQ (6,6), DCURL (4,6), E(10), FS(6), TEMP(1), TEMP1(1), TEMP2(6,4), P$LMAT(6,4)

12. Subroutine User: PLUG6

13. Subroutines Required: MPRD, TPRD

14. Remarks: None

1. Subroutine Name: TF$PRT

2. Purpose: Display pre-strain load vector for triangular ring

3. Equations: None

4. Input arguments: F$: pre-strain load vector

5. Output arguments: None

6. Error returns: None

7. Calling sequence: (F$)

8. Input tapes: None

9. Output Tapes: None

10. Scratch tapes: None

11. Storage required: Total Storage required is 100_{16} Bytes.

12. Subroutine user: PLUG5

13. Subroutines required: None

14. Remarks: None

1. Subroutine Name: TRAI$T

2. Purpose: To generate the pre-stress load vector for a triangular ring element

3. Equations & Procedures: The input matrices are combined by matrix manipulations to form the pre-stress load vector.

4. Input arguments:

$IGZER:	column of pre-stresses
GAMABQ:	transformation matrix
DCURL:	Integral value matrix
WIPR:	print control

5. Output arguments:

F$T: pre-stress load vector

6. Error Returns: None

7. Calling sequence:

($IGZER, GAMABQ, DCURL, F$T, WIPR)

8. Input tapes: None

9. Output tapes: None

10. Scratch tapes: None

11. Storage Required:

$IGZER(4),GAMABQ(6,6),DCURL(4,6),F$T(6),TEMP(6)

12. Subroutine User: PLUG6

13. Subroutines required: TFRD

14. Remarks: None

1. Subroutine Name: T$TPRT

2. Purpose: Display pre-stress load vector for triangular ring element

3. Equations: None

4. Input arguments: F$T

5. Output arguments: None

6. Error returns: None

7. Calling Sequence: (F$T)

8. Input tapes: None

9. Output tapes: None

10. Scratch tapes: None

11. Storage: Total Storage required is 100_{16} Bytes.

12. Subroutine User: PLUG6

13. Subroutines used: None

14. Remarks: None

1. Subroutine Name: PL6PRT

2. Purpose: To display structural damping, viscous damping, pre-strain and pre-stress matrices for a triangular ring element

3. Equations: None

4. Input Arguments:

   D$XP: structural damping vector
   DVXP: viscous matrix
   E1: pre-stress multiplier matrix
   P$LMAT: pre-strain multiplier matrix

5. Output Arguments: None

6. Error Returns: None

7. Calling Sequence: (D$XP, DVXP, E1, P$LMAT)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage: D$XP(45), DVXP(45), E1(4,4), P$LMAT(4,4)

12. Subroutine User: PLUG6

13. Subroutines Used: None

14. Remarks: None

1. Subroutine Name: PLUG 5

2. Purpose: To form the element matrices for a doubly curved ring (toroidal ring) discrete element. This ring configuration, defined by an arbitrary section of revolution of a complete right circular toroidal shell, enables smoothly continuous idealization of general axisymmetric thin shell problems.

   The matrices which are formed are:

   (1) Stiffness matrix
   (2) Stress matrix
   (3) Thermal load matrix + pressure load matrix
   (4) Thermal stress matrix

3. Equations and Procedures: There are two cases treated for this type of element. They are:

   (1) The angles of the interior and exterior membranes are not equal (Toroidal section)
   (2) The angles of the interior and exterior membranes are equal. (Conic section).

   In the second case, the interior angle is increased by a factor of .50° so that they can be treated as in case one. A special case arises for the degenerate situation where the two angles equal 90°. In this case a different path is followed.

   A variational (Lagrangian) approach is taken in formulating the discrete element representation. On account of this, it has been found necessary to use numerical integration techniques, namely the Romberg technique and the numerical quadrature technique.

   The sequence of procedures is as follows:

   (1) The first general part of the routine processes input information, forming constants to be used in calculations. Also, several constants are extracted from the inputed material and extra matrices.

   (2) After testing as to the relative values of the membrane angles (i.e. equal or not), the program selects the correct path to take in forming the integrals used in later calculations. Either the Romberg or Numerical Quadrature methods are used to evaluate the integrals.

   (3) Using the integrals and the program constants, the program forms several intermediate element matrices.

(4) By several matrix operations (multiplications, transformations, etc.), the stiffness matrix, AKEL, is formed.

(5) In like manners, the program forms the thermal load (FTEI) matrix, the pressure load (FPEL) matrix, the combined thermal and pressure load (TPEL) matrix, the stress ($EL) matrix, and the thermal stress (T$EL) matrix.

(6) After all the calculations are completed, the program calls a subroutine to print all the matrices.

4. Input Arguments:

IPL: Plug Number
NNØ: Number of node points
R: X - coordinates of nodes
Y: Y - coordinates of nodes
Z: Z - coordinates of nodes
TEMP: Node point temperatures
P: Node point pressures
Q$: Node point computed displacements
IP: Number of extra cards
KK: Code for computation of element stiffness matrix
KF: Code for computation of element thermal load
K$: Code for computation of element stress matrix
KM: Code for computation of element mass matrix
KN: Code for computation of element incremental matrix
KD$: Code for computation of element structural damping
KDV: Code for computation of element viscous damping
NØRD: Order of element stiffness matrix
MAT: Material properties table
EXTRA: Specific element information
NDIR: Number of directions for each grid point
NDEG: Number of types of movement allowed
IU$EL: Dummy
EP$IØN: Pre-strain load vector
$Ø: Pre-stresses
INNØ: Number of nodes
ICØNT: Code for use of grid point axes
NR: Number of rows in stress matrix
NØDE$: Node point numbers

5. Output Arguments:

NERR: Error return
NØINK: Number of elements in lower half matrices
AKEL: Stiffness matrix
ANEL: Incremental matrix
TPEL: Thermal load + pressure load matrix
$EL: Stress matrix
T$EL: Thermal stress matrix
AMA$$: Mass matrix
DAMPV: Viscous damping matrix
DAMP$: Structural damping matrix
N$EL: Number of elements in stress matrix
NMA$$: Number of elements in mass matrix
NDAMPV: Number of elements in viscous damping matrix
NDAMP$: Number of elements in structural damping matrix
GPAXEL: Grid point axes transformation matrix

6. Error Returns:

NERR = 0 No error
= 1 Plug number incorrect
= 2 Number of nodes incorrect
= 3 Number of input points incorrect
= 4 Order of matrix (nord) incorrect

7. Calling Sequence:

(IPL, NNØ, R, Y, Z, TEMP, P, Q$, IP, NORD, NERR, NØINK,
AKEL, ANEL, TPEL, EL, TEL, AMA$$, DAMPV, DAMP$, NR,
INNØ, NØDE$, NMA$$, NDMPV, NDMP$, N$EL, KI, KF, K$, KM,
KD$, KDV, KN, IU$EL, EP$LØN, $Ø MAT, EXTRA, GPAXEL,
NDIR, NDEG, ICØNT)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

T(21) W(10,18) W1(18,18) R(2) Y(2) Z(2) P(2) TEMP (12,3)
NØDE$(1) W2 (18,18) W3 (18,18) TAKEL (18,18) AKEL(171)
GAMM (10,18) XI(6,12) YI(6,12) X(6) B(10,18) D(10,10)
FTEL (18,1) GAM(10,18) FMEØ (10,2) FME1 (10,2) FFEQ(10,2)
FFE1 (10,2) E(2,2) AIK(2,2) AJK (2,2) ETØ (2,1) ET1(2,1)
ALTØ (2,1) ALT1 (2,1) V1 (18,2) V2 (18,2) V3 (18,1) V4(18,1)
V5 (18,1) V6 (18.1) FPEL (18) FPCQ (10,1) TPEL (18) $EL
(15,18) XXI (3) EXTRA(1) $CURL (15,10) T$SEL (15) TE1 (2,1)
TE2 (2,1) EM1 (2,1) EM2 (2,1) EP$LØN (1) $Ø (1) MAT(1)

12. Subroutine User: ELPLUG

13. Subroutines Required:

F4, F5, F6, ELTE$T, MPRD, GAMMAT, $CRLM, BMATRX, TPRD, FCURL, $ØLVE, DMATRX, M$TR, PLMX, PRINT5

14. Remarks: None

1. Subroutine Name: MØTR

2. Purpose: To change the storage mode of a matrix.

3. Equations and Procedures: MSTR will perform the operation on the right when MSA and MSR are equal to

MSA	MSR	PROCEDURE
0	0	[A] is moved to [R]
0	1	The upper triangle elements of a general matrix are used to form a symmetric matrix
0	2	The diagonal element of a general matrix are used to form a diagonal matrix
1	0	A symmetric matrix is expanded to form a general matrix
1	1	[A] is moved to [R]
1	2	The diagonal elements of a symmetric matrix are used to form a diagonal matrix
2	0	A diagonal matrix is expanded to form a general matrix
2	1	A diagonal matrix is expanded to form a symmetric matrix
2	2	[A] is moved to [R]

The codes for MØA and MØR stand for

0 General matrix form
1 Symmetric matrix form
2 Diagonal matrix form

4. Input Arguments:

[A] Input matrix

N Number of rows and columns in [A] and [R]

MØA Code designating storage mode of [A]

MØR Code designating storage mode of [R]

5. Output Arguments:

   [R] : Output matrix.

6. Error Returns: None

7. Calling Sequence: (A, R, N, M, Ø A, M Ø R)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Requir. Total Storage required is $29C_{16}$ Bytes.

12. Subroutine User: PLUG 5

13. Subroutines Required: L ØC

14. Remarks: Matrix [A] may not be in the same storage as [R].

1. Subroutine Name: RØMBER

2. Purpose: To integrate f (x) from x = a to x = b.

3. Equations and Procedures: The precision of large numbers in terms of number of significant digits and the accuracy of small numbers in terms of number of significant digits is measured. The subroutine terminates when either of these conditions is met.

4. Input Arguments:

A:	Lower limit
B:	Upper limit
NØSIG:	Number of correct significant digits (not more than 7)
NUM:	maximum number of halvings of (a,b) to be made (not more than 99)
KØDE:	controls the form of the print-out
FUNCT:	function of X - F4, F5, F6
X:	variable of integration

5. Output Arguments:

ITDØNE:	number of iterations
FINTG:	value of the integral
PRECIS:	actual number of significant digits attained

6. Error Returns: None

7. Calling Sequence: (A, B, NØSIG, PRECIS, NUM, ITDØNE, FINTG, KØDE, FUNCT, X)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

Total Storage required is 742_{16} Bytes.

12. Subroutine User: PLUG5

13. Subroutines Required: FUNCT

14. Remarks: None

1. Subroutine Name: F4

2. Purpose: To set up a function to be used by RØMBER in the computation of I_5, one of the six basic integrals used in PLUG5.

3. Equations and Procedures:

$$F4 = (X_1)^{X6-1} \text{ SIN}^2 (X_1,) /DEN$$

where

$$DEN = X_3 - X_2 X_5 + X_2 X_5 \text{ COS } (X_1) + X_2 X_4 \text{ SIN } (X_1)$$

4. Input Argument: X: array containing integration arguments

5. Output Arguments: F4: functional value

6. Error Returns: None

7. Calling Sequence: F4 (X)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is 256_{16} Bytes.

12. Subroutine User: RØMBER

13. Subroutines Required: SIN, CØS

14. Remarks: None

1. Subroutine Name: F5

2. Purpose: To set up a function to be used by RØMBER in the computation of i_5^j, one of the six basic integrals used in PLUG 5.

3. Equations and Procedures:

$$F5 = (X_1)^{X6^{-1}} \; 2 \sin(x_1) \cos(x_1)/DEN$$

where

$$DEN = x_3 - x_2 x_5 + x_2 x_5 \cos(x_1) + x_2 x_4 \sin(x_1)$$

4. Input Arguments:

X: array containing integration arguments

5. Output Arguments: F5 - functional value

6. Error Returns: None

7. Calling Sequence: F5(X)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is 282_{16} Bytes.

12. Subroutine User: RØMBER

13. Subroutines Required: SIN, CØS

14. Remarks: None

1. Subroutine Name: F6

2. Purpose: To set up a function to be used by RØMBER in the computation of i_6^I, one of the six basic integrals used in PLUG5

3. Equations and Procedures:

$$F6 = CONST \cdot \cos(X_1)/DEN$$

where

$$DEN = x_3 - x_2 x_5 + x_2 x_5 \cos(x_1) - x_2 x_4 \sin(x_1)$$

$$CONST = \begin{cases} 1, & x_6 = 1 \\ (x_1)^{x_6 - 1}, & x_6 \neq 1 \end{cases}$$

4. Input Arguments:

X: array containing integration arguments

5. Output Arguments:

F6: functional value

6. Error Returns: None

7. Calling Sequence: F6(X)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is 256_{16} Bytes.

12. Subroutine User: RØMBER

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: QUADI

2. Purpose: To evaluate integrals by an enclosed quadrature formula

3. Equations and Procedures:

Given the integrals of the form

$$i_2^j = \int_0^s \frac{\xi^j}{r_1 + \xi \cos \alpha_1} d\xi$$

when it is true that

$$\frac{s \cos \alpha_1}{r_1} < 1$$

it follows that

$$i_2^j \approx \frac{s^{j+1}}{r_1} \sum_{m=1}^{j+1} \frac{(-1)^{m-1}}{j+m} \left(\frac{s \cos \alpha_1}{r_1} \right)^{m-1}$$

where E_n is the error term.

The formula converges when

$$|E_n| \leq \frac{1}{j+m+1} \left(\frac{s \cos \alpha_1}{r_1} \right)^m$$

4. Input Arguments:

R1:	Change in coordinates (distance)
S:	Upper bound of integration
N:	Number of integral (N = j + 1)
CTRM:	Criteria for convergence CTRM = $\frac{S \cos \alpha_1}{r_1}$

5. Output Arguments:

XI: Value of approximation

6. Error Returns: If the quadrature doesn't converge after 1000 iterations, the program terminates.

7. Calling Sequence:

CALL QUADI (R1, S, N, CTRM, XI)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is $3C8_{16}$ Bytes.

12. Subroutine User: PLUG5

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: BMATRX

2. Purpose: To generate a matrix [B] , given specific input, for PLUG5

3. Equations and Procedures: The routine forms the terms of the matrix by direct assignment

4. Input Arguments:

   S: Variable used to form terms of matrix

5. Output Arguments

   B: completed transformation matrix

6. Error Returns: None

7. Calling Sequence: (B , $)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required. Total Storage required is 398_{16} Bytes.

12. Subroutine User: PLUG5

13. Subroutines Required: None

14. Remarks: Typical Element

    B(6, 9) = -1.0/2.0 * S * S * S)

1. Subroutine Name: DMATRIX

2. Purpose: To generate a matrix [D] , for Plug 5, given specific input.

3. Equations and Procedures: The routine forms the terms of the matrix by direct algebraic assignment

4. Input Arguments:

V C CA CA2 VA DM DB YI	: All variables used to form the terms

5. Output Arguments:

[D] : Completed Matrix

6. Error Returns: None

7. Calling Sequence

(D, V, C, CA, CA2, VA, DM, DB, YI)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is $BA6_{16}$ Bytes.

12. Subroutine User: Plug 5

13. Subroutines Required: None

14. Remarks: Typical Element

```
D(3,2) = DB * (2.*V*YI(4, 1) - 2.* YI(6,2) + D(4,1))
```

1. Subroutine Name: GAMMAT

2. Purpose: To generate a matrix [GAMM], given another matrix

3. Equations and Procedures: The routine rearranges the rows of the input matrix to form the output matrix.

4. Input Arguments:

   B: Input Matrix

5. Output Arguments:

   GAMM: Output Matrix

6. Error Returns: None

7. Calling Sequence: (GAMM, B)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is $1AA_{16}$ Bytes.

12. Subroutine User: PLUG5

13. Subroutines Required: None

14. Remarks:

    Typical Element GAMM (4, 3) = B (10, 3)

1. Subroutine Name: FCURL

2. Purpose: To generate 4 matrices, [FMEØ] ,[FME1] , [FFEØ] , and [FFE1] , given specific input, for Plug 5

3. Equations and Procedures: The routine forms the terms of the matrices by direct algebraic assignment.

4. Input Arguments:

   YI, S, LAM1 } : variables used to form the terms of the matrices.

5. Output Arguments:

   FMEØ, FME1, FFEØ, FFE1 } : output matrices

6. Error Returns: None

7. Calling Sequence: (FMEØ, FME1, FFEØ, FFE1, YI, $, LAM1)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

    FMEØ(10,2), FFEØ(10,2), FME1(10,2), FFE1(10,2), YI(6,12)

12. Subroutine User: PLUG5

13. Subroutines Required: None

14. Remarks: Typical Element

    FME1(4,2) = $1 * YI(4,5)

1. Subroutine Name: PLMX

2. Purpose : to generate a matrix [FPCQ], given specific input for Plug 5.

3. Equations and Procedures: The routine forms the terms of the matrix by direct algebraic assignment.

4. Input Arguments:

YI CØNSTl CØNST2 Pl	}	: Variables used to form the terms of the matrix

5. Output Arguments:

FPCQ	Output Matrix

6. Error Returns: None

7. Calling Sequence: (FPCQ, YI, CØNSTl, CØNST2, Pl)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

Total Storage required is 228_{16} Bytes.

12. Subroutine User

PLUG 5

13. Subroutines Required: None

14. Remarks: Typical Element

FPCQ (6,1) = CONSTl * (Pl * YI (1,2) - CONST2 * YI(1,3))

1. Subroutine Name: SCRLM

2. Purpose: To generate a matrix [$CURL] , given specific input, for PLUG5

3. Equations and Procedures: This routine forms the terms of the matrix by direct algebraic assignment.

4. Input Arguments:

   XXI:
   E:
   H: Variables used to form the terms of the matrix
   CONT:
   RP:
   ALF1:
   R1:
   IAM1:

5. Output Arguments:

   SCURL: output element stress matrix

6. Error Returns: None

7. Calling Sequence:

   ($CURL, XXI, E, H, CØNT, RP, ALF1, R1, LAM1)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

    Total Storage required is $9F8_{16}$ Bytes.

12. Subroutine User: PLUG5

13. Subroutines Required: None

14. Remarks:

    Typical Element

    $CURL(4,8) = $CURL(4,6) * 3.0 * XX2 - E(1,2) * 6.0 * XX1

1. Subroutine Name: $ØLVE

2. Purpose: To solve for lambdas as functions of XI.

   i.e. $\lambda = f(XI)$

3. Equations and Procedures: The routine uses algebraic techniques to arrive at a solution.

   eg.)

   $$LAM2 = \frac{\cos\left(A1 + \frac{XI}{RP}\right)}{R1 - RP * \left(\sin(A1) + \sin\left(A1 + \frac{XI}{RP}\right)\right)}$$

   where A1, R1, RP are constants

   LAM3 and LAM4 are similar

4. Input Arguments: A1, R1, RP, XI: Variables used for calculation of the lambdas

5. Output Arguments: LAM2, LAM3, LAM4: Output values

6. Error Returns: None

7. Calling Sequence: (A1, R1, RP, XI, LAM2, LAM3, LAM4, CØNT)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is 334_{16} Bytes.

12. Subroutine User: PLUG 5

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: PRINT5

2. Purpose: To print, as output, the intermediate matrices and single valued variables, generated in Plug 5.

3. Equations and Procedures: The routine contains the proper write and format statements.

4. Input Arguments: All the variables to be printed. (See calling sequence)

5. Output Arguments: None

6. Error Returns: None

7. Calling Sequence: (C, DM, DB, PHIB, RP, $, BB, RT, P$I1, P$I2, CØ$1, $IN1, XI, YI, B, D, W, W1, H, ALF2, ALF1, W3, R1, R2, Z1, Z2, EP, ET, VPT, AXI, ABETA, T1I, T1Ø, T2I, T2Ø, LAM1, AIK, AJK, ETØ, ET1, ALTØ, ALT1, E, FMEØ, FME1, FFEØ, FFE1, FTEL, P1, P2, CON$T1, CØN$T2, FPCQ, FPEL, $CURL)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: W(10,18), W1(18,18), XI(6,12), YI(6,12), B(10,18), D(10,10), FTEL(18,1), FMEØ (10,2), FME1(10,2), FFEØ(10,2), FFE1(10,2), E(2,2), AIK(2,2), AJK(2,2), ETØ(2,1), ET1(2,1), ALTØ(2,1), ALT1(2,1), FPEL(18), FPCQ(10,1), W(18,18), $CURL(15,10)

12. Subroutine User: PLUG 5

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: PLUG 14

2. Purpose: To compute the element stiffness, stress and diagonal mass matrix.

3. Equations and Procedures: The routine first generates the transformation matrix, PH, and prints it out (using P14PRT) if option is in effect. It then calculates the stress matrix depending on input code KI $\neq$ 2. It now calculates the stiffness matrix, transforms it to system coordinates using MULTF, and expands it using PØØF. If KI = 2, the routine will then calculate the lumped mass matrix and expand it using PØØF.

4. Input Arguments:

IPL:	Plug Number (must equal 14)
NNØ:	Number of node points (must equal 4)
X,Y,Z:	Three vectors of length four each having the X,Y,Z coordinates of the 4 node points.
NØRD:	Order of stiffness and mass matrix (must equal 24)
KI:	Selective calculation code
MAT:	Material properties array MAT (2) = E - Young's Modulus MAT (5) = u - Poisson Ratio MAT (22) = DENSM - mass density MAT (23) = CØNT - print control
EXTRA:	Extra input array (EXTRA (1) = T = thickness)

5. Output Arguments:

NERR:	Error return code
NØINK:	Number of elements in symmetric stiffness matrix (equals 300)
AKELXP:	Singly subscripted array of element stiffness matrix (symmetric lower half by rows)
SELXP:	Singly subscripted array of element stress matrix of size 1 x 24
AMASS:	Singly subscripted array of element mass matrix (symmetric lower half by rows)
NRSEL:	Number of rows in stress matrix (equals 1)
NMASS:	Number of elements in symmetric mass matrix (equals 300)
NSELXP:	Number of elements in stress matrix (equals 24)
TSELXP:	Thermal stress vector of length 1 is set to zero.
TPELXP:	Applied load vector of length 24 is set to zero.

6. Error Returns: If NERR $\neq$ 0 then error was detected in input arguments. (See ELPLUG)

7. Calling Sequence:

(IPL, NNØ, X,Y,Z, TEMP, P, QS, IP, NØRD, NØINK, AKELXP, ANEL, TPELXP, SELXP, TSELXP, AMASS, NDMPV, NDMPS, NSELXP, KI, KF, KS, KM, KDS, KDV, KN, IUSEL. EPSIØ, SØ, MAT, EXTRA, GPAXEL, NDIR, NDEG, ICØNT)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11: Storage Required: 505 decimal locations of work storage used from unlabeled common block.

12: Subroutine User: ELPLUG

13: Subroutines Required:

ELTEST, P14PRT, MULTF, PØØF

14: Remarks: All arguments in calling sequence not defined were not used in subroutine.

1. Subroutine Name: MULTF

2. Purpose: To preform the matrix multiplication B transpose times A times B, where A is a symmetric matrix and B is a rectangular matrix.

3. Equations and Procedures:

   C = B (transpose) * A * B

   The routine first generates the product of a row of B transpose times each column of A and stores this in a temporary storage V. It then multiplies V times the appropriate columns of B to generate the corresponding row of C.

4. Input Arguments:

   A : The symmetric input matrix doubly dimensioned 8x8 with only symmetric lower half needed.

   NA: Order of A must be less than 9.

   B: The rectangular input matrix doubly dimensioned 8x12 with size NA x NBC

   NBC: Number of columns of B (less than 12)

   V: A work storage vector of length NA.

5. Output Arguments:

   C: The results of the multiplication, doubly dimensioned 12x12 with only symmetric lower half returned. Size is NBC x NBC.

6. Error Returns: None

7. Calling Sequence:

   (A, NA, B, NBC, V, C)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is 350_{16} Bytes.

12. Subroutine User: PLUG 14

13. Subroutines Require: None

14. Remarks: None

1. Subroutine Name: POOF

2. Purpose: Expand element stiffness matrix (lower symmetric by rows or upper symmetric by col) and element thermal load vector and add the components into the expanded matrix and vector in their appropriate positions.

3. Procedure: Using the decoding vector determine the locations of the components of the element stiffness matrix in the new expanded (assembled stiffness) matrix and add these old element components into their new positions. The same procedure is used for the thermal load vector.

4. Input Arguments:

LIST	-	decoding vector consisting of NORD components the subscript of each component gives the old (element) row or column and the component itself gives the row or column in the new expanded matrix.
NORD	-	order of old element stiffness matrix (AKEL) also length of old thermal load vector (FTEL).
AKEL	-	old element stiffness matrix [upper symmetric by cols].
FTEL	-	old thermal load vector.

5. Output Arguments:

AK	-	expanded stiffness matrix [upper symmetric by cols] .
FCOL	-	expanded thermal load vector.

6. Error Returns: None

7. Calling Sequence:

(LIST,NORD,AKEL,FTEL,AK,FCOL)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is 300_{16} Bytes.

12. Subroutine Users: PLUG8

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: P14PRT

2. Purpose: To print out on the system output unit the variables in the input argument list.

3. Equations and Procedures:

4. Input Arguments:

D12:	variable	printed	out	and	labled	D12
D15:	"	"	"	"	"	D15
D35:	"	"	"	"	"	D35
ALX:	"	"	"	"	"	LAMX
ALY:	"	"	"	"	"	LAMY
ALZ:	"	"	"	"	"	LAMZ
PX:	"	"	"	"	"	PSIX
PY:	"	"	"	"	"	PSIY
PZ:	"	"	"	"	"	PSIZ
XP4:	"	"	"	"	"	XP4
YP4:	"	"	"	"	"	YP4

PH: An 8 x 12 matrix printed out and labled ELEMENT TRANSFORMATION MATRIX

5. Output Arguments: None

6. Error Returns: None

7. Calling Sequence:

(D12, D15, D35, ALX, ALY, ALZ, PX, PY, PZ, XP4, YP4, PH)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is 418_{16} Bytes.

12. Subroutine User: PLUG 14

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: PLUG8

2. Purpose: Generate element matrices for the trapezoidal ring element.

3. Equations and Procedures:

a) Call subroutine ELTEST to verify input control values.
b) Initialize material properties, node point pressures, geometric constants and integration constants.
c) Call subroutine SUBINT to calculate other integrals.
d) Define transformation matrix to transform to displacement degrees of freedom.
e) Generate mechanical property matrix, thermal coefficient matrix, stiffness matrix and thermal load matrix.
f) Call subroutine POOF to calculate pressure load vector.
g) Call subroutine ERIC to inflate stiffness matrix and element thermal load vector.
h) Generate stress matrix and thermal stress.
i) Call P8MASS to generate element mass matrices.
j) Print debug print if requested.

4. Input Arguments:

IPL	-	internal element identification number (8)
NNO	-	number of element defining points (4)
XC	-	coordinates of element defining points
YC	-	coordinates of element defining points
ZC	-	coordinates of element defining points
TPS	-	temperatures at element defining points
PVP	-	pressures at element defining points
QS	-	input displacements at element defining points (not used)
IP	-	not used
NORD	-	total element degrees of freedom (12)
K1	-	number of storages required for element stiffness matrix (NORD*(NORD + 1)/2)
INNO	-	not used
NL	-	array containing grid point numbers of element defining points
KK	-	suppression control for element stiffness matrix
KAF	-	suppression control for element thermal and pressure load matrices
KS	-	suppression control for element stress matrix
KTS	-	suppression control for element thermal stress matrix
KAM	-	suppression control for element mass matrix
KDS	-	suppression control for structural damping matrix
KDV	-	suppression control for structural viscous matrix
KSN	-	suppression control for element incremental stiffness matrix
IUMEL	-	not used
EPSIO	-	input pre-strains

4. Input Arguments, Contd:

SO	-	input pre-stresses
MAT	-	input temperature interpolated material properties
EXTRA	-	special element input
GPAXEL	-	grid point axes transformation matrices (not used)
NDIR	-	number of directions of element defining points (3)
NDEG	-	number of solution degrees of freedom (1-translation)
ICONT	-	grid points axes indicator

5. Output Arguments:

NERR	-	error indicator
2K	-	element stiffness matrix
ANEL3	-	element incremental stiffness matrix
XT	-	element thermal and pressure load matrix
SEL	-	element stress matrix
SZALEL	-	element thermal stress matrix
AMASS	-	element mass matrix
DAMPV	-	element viscous damping matrix
DAMPS	-	element structural damping matrix
NRSEL	-	number of rows in element stress and thermal stress matrices
NMASS	-	number of storage required for element mass matrix
NDMPV	-	number of storages required for element viscous damping matrix
NDMPS	-	number of storages required for element structural damping matrix
NSEL	-	number of storages required for element stress matrix

6. Error Returns:

If no error, then NERR is set to zero
If IPL ≠ 28, then NERR is set to one
If NNO ≠ 4, then NERR is set to two
If NORD ≠ 12, then NERR is set to four.

7. Calling Sequence:

```
Call PLUG8(IPL,NNO,XC,YC,ZC,TPS,PVP,QS,IP,NORD,NERR,K1,ZK,
          ANEL3,XT,SEL,SZALEL,AMASS,DAMPV,DAMPS,NRSEL,INNO,
          NL,EPSIO,SO,MAT,EXTRA,GPAXEL,NDIR,NDEG,ICONT)
```

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is $588E_{16}$ Bytes.

12. Subroutine User: ELPLUG

13. Subroutines Required:

ELTEST,SYMPRT,LOC,ELTEST,MPRD,TPRD,MSTR,SUBINT,ZMRD,ZTRD,
KMPY,ERIC,POOF,P8MASS

14. Remarks: None

1. Subroutine Name: P8MASS

2. Purpose: To generate element mass matrix for PLUG8.

3. Equations and Procedures: The (8x8) reduced mass matrix AMEL3 is formed in terms of the integration constants. Then the transformation to displacement degrees of freedom is performed. The matrix is then expanded to order (NORDxNORD) by subroutine POOF.

4. Input Arguments:

DENSM	-	element mass density vector (first element)
HH	-	transformation to displacement degrees of freedom
NORD	-	order of mass matrix (= 12)
NMASS	-	number of elements in mass matrix (= 78)
I10 - I32	-	integration constants for rectangular cross section ring
CHH	-	working storage (64)
SH	-	working storage (64)
LIST	-	code list for transforming system reduced degrees of freedom to system expanded degrees of freedom
AMASS	-	work storage (36)

5. Output Arguments:

AMASE	-	resultant mass matrix (symmetric 12 x 12)
AMEL3	-	order 8 MASS matrix before transformation and expansion to order 12

6. Error Returns: None

7. Calling Sequence:

```
Call P8MASS(DENSM,HH,AMASE,NORD,NMASS,I10,I11,I12,I20,I21,
           I22,I30,I31,I32,CHH,SH,AMEL3,LIST,AMASS)
```

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

Total Storage required is 6C2 Bytes.

12. Subroutine User: PLUG8

13. Subroutines Required:

MPRD
TPRD
MSTR
POOF

14. Remarks: None

1. Subroutine Name: SUBINT

2. Purpose: Solve integral used in integration constants for PLUG8 element matrix definitions.

3. Equations and Procedures:

$$H = \iint \bar{Z}^{Q} \, DR \, DZ$$

Solve for H given R, Z and Q for values of 0, 1 and 2. The R and Z values are coordinates of a trapezoid area. The area is divided into two triangles (A and B). The centroid and area of each triangle is found

$$(\bar{R}_A, \bar{Z}_A, \bar{R}_B, \bar{Z}_B) \; (A_A, A_B)$$

$$H_A = \frac{(A_A \, \bar{Z}_A{}^{Q})}{\bar{R}_A} \qquad H_B = \frac{(A_B \, \bar{Z}_B{}^{Q})}{\bar{R}_B}$$

$$H = H_A + H_B$$

4. Input Arguments:

R	-	variable (double precision) array
Z	-	variable (double precision) array
Q	-	integer (exponent)

5. Output Arguments:

H	-	value of integral (double precision)

6. Error Returns: None

7. Calling Sequence:

```
Call SUBINT(R,Z,Q,H)
```

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is $2A6_{16}$ Bytes.

12. Subroutine User: PLUG8

13. Subroutines Required: None

14. Remarks: R, Z and H must be double precision in calling program.

1. Subroutine Name: ZMRD

2. Purpose: Multiply two matrices to form a resultant matrix. (This is a modification of MPRD to include double precision.)

3. Equations and Procedures:

$$[R] = [A]\,[B]$$

4. Input Arguments:

A	-	first input matrix (double precision)	
B	-	second input matrix (single precision)	
N	-	number of rows in A matrix	
M	-	number of rows in B matrix	
L	-	number of columns in B	
MSA	-	control on storage mode of A	} See remarks
MSB	-	control on storage mode of B	

5. Output Arguments:

R - resultant matrix (double precision)

6. Error Returns: None

7. Calling Sequence:

Call ZMRD(A,B,R,N,M,MSA,MSB,L)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is $3FE_{16}$ Bytes.

12. Subroutine User: PLUG8

13. Subroutines Required: LOC

14. Remarks:

1. General subroutine.
2. Storage control of A and B matrix
   - 0 - General
   - 1 - Symmetric (upper half)
   - 2 - Diagonal
3. A and R must be double precision in calling program.

1. Subroutine Name: ZTRD

2. Purpose: Transpose a matrix and post multiply by another to form a resultant matrix.

   This routine is a modification of TPRD to include double precision.

3. Equations and Procedures:

   $[R] = [A]^T[B]$

   $[A]$ is not actually transposed.

4. Input Arguments:

A	-	first input matrix (single precision)
B	-	second input matrix (double precision)
N	-	number of rows in A and B
M	-	number of columns in A and rows in R
L	-	number of columns in B and rows in R
MSA	-	control of storage mode of A } See remarks
MSB	-	control of storage mode of B

5. Output Arguments:

R	-	resultant matrix

6. Error Returns: None

7. Calling Sequence:

   Call ZTRD(A,B,R,N,M,MSA,MSB,L)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is 406_{16} Bytes.

12. Subroutine User: PLUG8

13. Subroutines Required: LOC

14. Remarks:

    1. General subroutine.
    2. Storage control of A and B matrix
       - 0 - General
       - 1 - Symmetric (upper half)
       - 2 - Diagonal
    3. B must be double precision in calling program.

1. Subroutine Name: KMPY

2. Purpose: Multiply each element of a matrix by a scalar to form a resultant matrix.

3. Equations and Procedures: This subroutine multiplies each element in the input matrix A, by a scalar C and places the result in R. Soubroutine LOC calculates the vector length IT of the resultant vector R.

4. Input Arguments:

A	-	name of input matrix
C	-	scalar multiplier
N	-	number of rows in matrix A
M	-	number of columns in matrix A
MS	-	storage mode of matrix A - 0 General - 1 Symmetric - 2 Diagonal

5. Output Arguments:

R	-	name of output matrix
N,M,MS	-	defined above, refer to the R matrix also.

6. Error Returns: None

7. Calling Sequence:

```
Call KMPY(A,C,R,N,M,MS)
```

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is $1F8_{16}$ Bytes.

12. Subroutine User: PLUG8

13. Subroutine Required: LOC

14. Remarks: Good comments are available in the subroutine listing.

1. Subroutine Name: ERIC

2. Purpose: Compute pressure load vector (FP)

3. Equations and Procedures:

$$FP = SCAL \quad [HH^T] \; [QP] \; [HP] \; [PV]$$

50 Multiply ITH col of HH * QP to get WORK(8) vector
Multiply WORK * HP to get WORK2(8) vector
Multiply WORK2 * PV * SCAL to get FP(I)
Update I and go to 50

4. Input Arguments:

HH - EQ. 2.10
QP - EQ. 4.3.1.27 (less 2Π)
HP - EQ. 4.3.1.29
PV - EQ. 4.3.1.29
SCAL - 2Π See 4.3.1.27

5. Output Arguments:

FP - pressure load vector

6. Error Returns: None

7. Calling Sequence:

(HH, QP, HP, PV, FP, SCAL)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is $36C_{16}$ Bytes

12. Subroutine User: PLUG8

13. Subroutines Required: None

14. Remarks: Equation are given in "Formulation and Evaluation of a Trapezoidal Cross Section Ring Discrete Element", R. H. Mallett, S. Jordan, November, 1966.

1. Subroutine Name: PLUG17

2. Purpose: (1) To generate both membrane and flexural element matrices of a triangular thin plate, (2) If applicable, generate incremental matrices for instability.

3. Equations and Procedures:

A. Formulation of Equation - The formulation for any computations involved in evaluating the element matrices will be found in references (1) and (2). (See Remarks section of this report.) Modifications were, however, necessary to make the notation compatable with 3648 procedures and applications. The formulations and coding are not necessarily in the same sequence or labeling.

B. Initial Computations -

1. Constants have to be set for:
   a) If the element matrices are to be computed, LAMDA(I) = 1 where

      I = 1, for membrane stiffness and stress
      I = 2, for flexural stiffness and stress
      I = 3, for membrane thermal load and stress
      I = 4, for flexural thermal load and stress.

   b) The incrementals will not be computed: INCREM = 0 since ICONT = 0.
2. Material properties and element data from MAT and EXTRA array noting that if either membrane or flexure thickness is zero, the appropriate LAMDA above is reset to zero.
3. According to reference (2), transformation matrices have to be formulated with the appropriate direction cosines.
   a) From cylinder coordinates to local coordinates

$$\{x_\ell\} = [T_{\ell c}] \{x_o\} \qquad (1)$$

   b) From cylinder coordinates to oblique coordinates

$$\{x_o\} = [T_{oc}] \{x_c\} \qquad (2)$$

   where $\{x_\ell\}$ are the local x, y, z coordinates
   $\{x_c\}$ are the cylinder x, y, z coordinates
   $\{x_c'\}$ are some other orthogonal X' Y' Z' coordinates

   $[T_{\ell c}]$ $[T_{oc}]$ contain the respective direction cosines.

   Since the element displacements are in local coordinates, combining equations (1) and (2) yields

$$\{x_\ell\} = [T_{\ell c}] [T_{oc}]^T \{x_o\} = [TTOBL] \{x_o\} \qquad (3)$$

3. Equations and Procedures (Contd.):

4. Transformation of the above cited displacements, x_1, x_2, x_3, y_1, etc. into 3648 notation x_1, y_1, z_1, x_2, etc. will result in the formulation of

$$\{x_1, x_2, x_3, \text{etc}\} = [T1718] \{x_1, y_1, z_1, \text{etc}\} \qquad (4)$$

C. Flexural Computations - (All equations cited are in Reference 1).

1. Using equation IV-6, the $[\bar{B}]$ matrix is formulated. However, it should be noted that the SEL array is used to relabel the displacements as W, θ_x and θ_y (instead of θ_x, θ_y, W).
2. Using equations IV-15, 16, and 17, the geometric properties of the element are first defined in local and then in global coordinates. These are shown as Figures IV-3 and IV-2 respectively.
3. If the incrementals are to be computed (N_x, N_y, N_{xy}) the following sequence of operations take place:
   a) Using equation IV-14, the respective $[\bar{C}]$ matrices are formulated.
   b) The respective incremental is formulated according to equation IV-11 and then transferred to 3648 notation by $[T1718]$.
4. The remaining element matrices are then formulated according to the respective equations cited:
   a) Stiffness - Equations IV-2, 6 and 10
   b) Stress - Equation IV-24
   c) Thermal Load - Equation IV-21
   d) Thermal Stress - Equation IV-26

D. Membrane Computations - (All equations cited are in Reference 1). The following membrane matrices are then formulated according to the respective equation cited:

a) Element - Equations II-1, 5 and 11
b) Stress - Equation II-16
c) Thermal Stress - Equation II-25
d) Thermal Load - Equation II-22

E. Remaining Operations - The element stiffness, stress and thermal load matrices are then transformed first to global and then to 3648 notation.

4. Input Arguments:

NCE1	-	number of node points
ZELC,YELC,ZELC	-	X, Y and Z coordinates
TEL,PEL	-	temperature and pressure array
NORD	-	order of element stiffness matrix
NCEI	-	node point numbers
GPAXEL	-	grid point axes transformation for element
KN	-	control for instability (if set = 1, incrmentals computed)
ICONT	-	control of grid point axes transformation
MAT	-	material properties array
EXTRA	-	element properties array

5. Output Arguments:

NOINK	-	number of elements in stiffness matrix
AKELX	-	elements of stiffness matrix (symmetric - bottom half)
FTELX	-	elements of thermal load matrix
SELX	-	elements of stress matrix
PTEL	-	elements of thermal stress matrix
NRSEL	-	number of rows in stress matrix (5)
NSEL	-	number of elements in stress matrix (90)

6. Error Returns:

(a) NERR - standard plug checks from ELTEST
(b) If points (1) and (2) have same coordinates call EXIT
(c) If B^{-1} is singular - call EXIT.

7. Calling Sequence:

```
Call PLUG17(IPL,NCE1,XELC,YELC,ZELC,TEL,PEL,QSEL,IP,NORD,NERR,
            NOINK,AKELX,ANELX1,FTELX,SELX,PTEL,AMASS,DAMPV,
            DAMPS,NRSEL,NNO,NCEI,NMASS,NDMPV,NDMPS,NSEL,KK,
            KF,KS,KM,KDS,KDV,KN,IUSEL,EPSIO,SO,MAT,EXTRA,
            GPAXEL,NDIR,NDEG,ICONT)
```

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

a) Variables		b) Definition
T1718 (24,24)	-	Transformation matrix to 3648 notation
TTOBL (3,12)	-	Transformation matrix from local to global or oblique coordinates
SEL (17,24)	-	Working area and stress matrix
ANELX (300)	}	Incremental matrices in Cylinder Notation
ANELY (300)		
ANELXY (300)		
ANELEX (300,3)	-	Incremental matrices for Instability in 3648 notation

12. Subroutine User: ELPLUG

13. Subroutines Required:

DIRCOS	BCB
BCB12	MATB
FTELQ	MAB
SELQ	KOBLIQ

14. Remarks:

a) Controls are reset in programs to compute everything but the incrementals. Initial test phase had KN = 1 to check these computations.

b) Plug not tested out if either the flexural or membrane thickness is zero (certain portions of plug will be bypassed as LAMDA is set = 0).

c) Thermal load will probably have to be rederived as 2nd input TEMP is thermal moments M^X and not the thermal gradient as prescribed for flexural elements (PLUGS 1 and 2).

d) References:

(1) Bell Report No. D2114-95005, "Derivation of the Force - Displacement Properties of Triangular and Quadrilateral Orthotropic Plates in Plane Stress and Bending" - Gallagher, Huff - dated Jan. 1964.

(2) Bell Report No. D2114-95008, "Detailed Description Computer Program for Stiffened Cylinder Analysis" Gallagher, Huff, Dale - dated Jan. 1964.

1. Subroutine Name: DIRCOS

2. Purpose: To evaluate the direction cosines given any 3 points that define a plane.

3. Equations and Procedures: Subscripts 1, 2 and 3 refer to the 3 points of the plane. Dropping a perpendicular from point 3 to the line connecting 1 and 2 results in point a. The following computations are done in order to determine the direction cosines.

$$l_{12}^2 = (x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2$$

$$l_{13}^2 = (x_3 - x_1)^2 + (y_3 - y_1)^2 + (z_3 - z_1)^2$$

$$l_{23}^2 = (x_3 - x_2)^2 + (y_3 - y_2)^2 + (z_3 - z_2)^2$$

$$l_{1a} = (l_{12}^2 + l_{13}^2 - l_{23}^2)/2l_{12}$$

$$\lambda x = \frac{x_2 - x_1}{l_{12}}, \quad \lambda y = \frac{y_2 - y_1}{l_{12}}, \quad \lambda z = \frac{z_2 - z_1}{l_{12}}$$

$$x_a = x_1 + \lambda_x \; l_{1a}$$

$$y_a = y_1 + \lambda_y \; l_{1a}$$

$$z_a = z_1 + \lambda_z \; l_{1a}$$

$$l_{a3}^2 = (x_3 - x_a)^2 + (y_3 - y_a)^2 + (z_3 - z_a)^2$$

3. Equations and Procedures,(Contd.):

$$\Psi_x = \frac{x_3 - x_a}{l_{a3}}, \quad \Psi_y = \frac{y_3 - y_a}{l_{a3}}, \quad \Psi_z = \frac{z_3 - z_a}{l_{a3}}$$

$$\nu_x = \lambda_y \Psi_z - \lambda_z \Psi_y$$

$$\nu_y = \lambda_z \Psi_x - \lambda_x \Psi_z$$

$$\nu_z = \lambda_x \Psi_y - \lambda_y \Psi_x$$

4. Input Arguments:

XEL1,YEL1,ZEL1 - X, Y, Z coordinates of plane

5. Output Arguments:

XLAMD1
YLAMD1
ZLAMD1
XPSI1
ZPSI1
XNU1
YNU1
ZNU1
} - direction cosines.

ALI21 - distance between point 1 and 2 of the plane.

6. Error Returns: None

7. Calling Sequence:

```
Call DIRCOS(XEL1,YEL1,ZEL1,XLAMDL,YLAMD1,ZLAMD1,XPSI1,YPSI1,
     :    XNV1,YNU1,ZNU1,ALI21)
```

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is 576_{16} Bytes.

12. Subroutine User: PLUG17, PLUG18

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: BCB12

2. Purpose: To evaluate a triple product matrix where all matrices are square.

3. Equations and Procedures: Dependent upon an input variable (ISIN2) when ISIN2 = 5.

$$AKEL = [A^{-1}]^T [C] [A^{-1}]$$

when ISIN2 = 11

$$AKEL = [A]^T [C] [A]$$

where A now contains elements of A^{-1}.

4. Input Arguments:

A	-	matrix to be inverted or the inverted matrix
C	-	symmetric matrix bottom half
NOR2	-	order of matrices
JNL1	-	dummy - set equal to 1
IKELW	-	print option
JEL1	-	dummy - set equal to 1
ISIN2	-	input code for above
NEC1	-	node points
SUBTI1	-	title of matrix
SUBTI2	-	type of element
NCE2	-	number of grid points

5. Output Arguments:

AKEL - results of the above triple product.

6. Error Returns: If A is singular - print out error and EXIT.

7. Calling Sequence:

```
Call BCB12(A,C,NOR2,JNL1,IKELW,JEL1,ISIN2,AKEL,NCE2,NCE1,
          SUBTI1,SUBTI2)
```

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is 1208_{16} Bytes.

12. Subroutine User: PLUG17, PLUG18

13. Subroutines Required: None - has a built in inverse routine.

14. Remarks: Note that maximum size of matrix is only 12.

1. Subroutine Name: KOBLIQ

2. Purpose: To evaluate

$$(TTOBL)^T \ (AKEL) \ (TTOBL)$$

3. Equations and Procedures: TTOBL is a compressed transformation matrix (3,12) that is labeled u, v, w for each node point. Since AKEL is labeled u_1, u_2, u_3, etc., the appropriate manipulation is done in this routine to do the above product.

4. Input Arguments:

NI	-	order of matrices
TTOBL	-	transformation matrix of element
AKEL	-	element stiffness matrix
SUBTI1, SUBTI2	-	labeling of printout
IKELW	-	print option
C	-	working storage
NA1	-	number of nodes defining element
NAI	-	node points
ROW	-	working storage

5. Output Arguments:

AKEL	-	element stiffness matrix

6. Error Return: None

7. Calling Sequence:

```
Call KOBLIQ(NI,TTOBL,AKEL,SUBTI1,SUBTI2,IKELW,C,NA1,NAI,ROW)
```

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is AFO_{16} Bytes.

12. Subroutine User: PLUG17, PLUG18

13. Subroutine Required: None

14. Remarks: Note that dimension for ROW dictates size of multiplication.

1. Subroutine Name: P1718M

2. Purpose: Initialize element properties from the material table for membrane properties with flexural data only from PLUG 17 and PLUG 18.

3. Equations and Procedures:

EXEL = MAT(14)
EYEL = MAT(15)
BETA = EXEL/EYEL
XYMU = MAT(16)
ALFAEL = MAT(17)
GXYEL = MAT(18)

4. Input Arguments: MAT

5. Output Arguments:

EXEL
BETA
XYMU
GXYEL
ALFAEL

6. Error Return: None

7. Calling Sequence:

P1718M(MAT,EXEL,BETA,XYMU,GXYEL,ALFAEL)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is $1CO_{16}$ Bytes.

12. Subroutine User: PLUG 17 ; PLUG 18

13. Subroutine Required: None

14. Remarks: None

1. Subroutine Name: SELQ

2. Purpose: To transform the stress matrix (generated by PLUG17 and PLUG18) to the stress system required - generally local).

3. Equations and Procedures:

$$[S]_{TRANS} = [S] \quad [TTOBL]$$

where [S] is the stress matrix generated by 17 and/or 18. [TTOBL] is the transformation matrix from global to local or global to oblique.

4. Input Arguments:

NORD6	-	number of columns in stress matrix
TTOBL	-	transformation matrix
IKELW	-	print option
A	-	element stress matrix
NRSEL	-	number of rows in stress matrix
ROW	-	working storage

5. Output Arguments:

A	-	stress matrix transformed to local system

6. Error Returns: None

7. Calling Sequence:

```
Call SELQ(NORD6,TTOBL,IKELW,A,NRSEL,ROW)
```

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is 664_{16} Bytes.

12. Subroutine User: PLUG17, PLUG18

13. Subroutines Required: None

14. Remarks:

    1. 12 elements at one time (membrane or flexure) are put into the working area (ROW).
    2. Note again the labeling of:
       (a) TTOBL = u_1, v_1, w_1, etc.
       (b) SEL = u_1, u_2, u_3, u_4, etc.

1. Subroutine Name: FTELQ

2. Purpose: To transform the element thermal (local) load into global or oblique system.

3. Equations and Procedures:

$$\{F\}_x^{\alpha} = [\text{TTOBL}]^T \{F_{ele}^t\}$$

where TTOBL is the transformation matrix.

$\{F_{ele}^t\}$ is the element local thermal load

$\{F\}_x^{\alpha}$ is the transformed load

4. Input Arguments:

NORD6	-	size of the load vector
TTOBL	-	transformation matrix
IKELW	-	print option
THMOEL	-	local thermal load
ROW	-	working storage

5. Output Arguments:

THMOEL - transformed thermal load

6. Error Returns: None

7. Calling Sequence:

Call FTELQ(NORD6,TTOBL,IKELW,THMOEL,ROW)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is $56C_{16}$ Bytes.

12. Subroutine User; PLUG17, PLUG18

13. Subroutines Required: None

14. Remarks:

1. Note that dimension of ROW(12) indicates 12 elements at a time transformed (membrane or flexure).
2. Note labeling of:

(a) $[\underline{TTOBL}]$ - F_{x_1}, F_{y_1}, F_{z_1}, etc.

(b) $\{F^t_{ele}\}$ - F_{x_1}, F_{x_2}, F_{x_3}, etc.

1. Subroutine Name: PLUG18

2. Purpose: (1) To generate both membrane and flexural element matrices of a quadrilateral thin plate, (2) If applicable, generate incremental matrices for instability.

3. Equations and Procedures:

A. Formulation of Equation - The formulation for any computations involved in evaluating the element matrices will be found in references (1) and (2). (See Remarks Section of this report.) Modifications were, however, necessary to make the notation compatable with 3648 procedures and applications. The formulations and coding are not necessarily in the same sequence or labeling.

B. Initial Computations -

1. Constants have to be set for:
   a) If the element matrices are to be computed, LAMBA(I) = 1 where
      I = 1, for membrane stiffness and stress
      I = 2, for flexural stiffness and stress
      I = 3, for membrane thermal load and stress
      I = 4, for flexural thermal load and stress.
   b) The incrementals will not be computed INCREM = 0 since ICONT = 0.
2. Material properties and element data from MAT and EXTRA array noting that if either membrane or flexure thickness is zero, the appropriate LAMDA above is reset to zero.
3. According to reference (2), transformation matrices have to be formulated with the appropriate direction cosines.
   a) From cylinder coordinates to local coordinates

$$\{x_\ell\} = [T_{\ell c}]\ \{x_o\} \qquad (1)$$

   b) From cylinder coordinates to oblique coordinates

$$\{x_o\} = [T_{oc}]\ \{x_o\} \qquad (2)$$

   where $\{x_\ell\}$ are the local x, y, z coordinates
   $\{x_c\}$ are the cylinder x, y, z coordinates
   $\{x_o\}$ are some other orthogonal X' Y' Z' coordinates
   $[T_{\ell c}]$ $[T_{oc}]$ contain the respective direction cosines.
   Since the element displacements are in local coordinates, combining equations (1) and (2) yields

$$\{x_\ell\} = [T_{\ell c}]\ [T_{oc}]^T \{x_o\} = [TTOBL]\ \{x_o\} \qquad (3)$$

3. Equations and Procedures (Contd):

4. Transformation of the above cited displacements x_1, x_2, x_3, y_1, etc. into 3648 notation x_1, y_1, z_1, x_2, etc. will result in the formulation of

$$\{x_1, x_2, x_3, \text{etc}\} = [T1718] \{x_1, y_1, z_1, x_2, \text{etc}\} \quad (4)$$

C. Flexural Computations - (All equations cited are in Reference 1):

1. Using equation V-5, the (B) matrix is formulated. However, it should be noted that the SEL array is used to relabel the displacements as W, θ_x and θ_y (instead of θ_x, θ_y, W).
2. Using equations V-19, and 21, the geometric properties of the element are first defined in local and then in global coordinates. These are shown as Figures V-8 and V-7 respectively.
3. If the incrementals are to be computed (N_x, N_y, N_{xy}) the following sequence of operations take place:
   a) Using equation V-11, the respective (C) matrices are formulated.
   b) The respective incremental is formulated according to equation V-12 and then transferred to 3648 notation by (T1718).
4. The remaining element matrices are then formulated according to the respective equations cited:
   a) Stiffness - Equations V-2, 3rd and 5
   b) Stress - Equations V-9 and 30
   c) Thermal Load - Equations V-26
   d) Thermal Stress - Equation V-32

D. Membrane Computations - (All equations cited are in Reference 1). The following membrane matrices are then formulated according to the respective equation cited:

a) Element - Equations III-2, 8 and 12
b) Stress - Equation III-26
c) Thermal Stress - Equation III-25
d) Thermal Load - Equation III-22

E. Remaining Operations - (1) The element stiffness, stress and thermal load matrices are then transformed first to global and then to 3648 notation, (2) The stress matrix is now expanded to be consistent with 3648 applications by (T18ST).

4. Input Arguments:

NCEl	-	number of node points
XELC,YELC,ZELC	-	X, Y and Z coordinates
TEL,PEL	-	temperature and pressure array
NORD	-	order of element stiffness matrix
NCEI	-	node point numbers
GPAXEL	-	grid point axes transformation for element
KN	-	control for Instability (If set = 1, Incrementals Computed)
ICONT	-	control of grid point axes transformations
MAT	-	material properties array
EXTRA	-	element properties array

5. Output Arguments:

NOINK	-	number of elements in stiffness matrix
AKELK	-	elements of stiffness matrix (symmetric - bottom half)
FTELK	-	elements of thermal load matrix
SELKP	-	elements of stress matrix
PTELK	-	elements of thermal stress matrix
NRSEL	-	number of rows in stress matrix (40)
NSEL	-	number of elements in stress matrix (900)

6. Error Returns:

(a) NERR - standard plug checks from ELTEST
(b) If points (1) and (2) have same coordinates call EXIT
(c) If B^{-1} is singular - call EXIT

7. Calling Sequence:

```
Call PLUG18(IPL,NCEl,XELC,YELC,ZELC,TEL,PEL,QSEL,IP,NORD,
            NERR,NOINK,AKELX,ANELX1,FTELX,SELKP,PTELK,AMASS,
            DAMPV,DAMPS,IRSEL,NNO,NCEI,NMASS,NDMPV,NDMPS,NSEL,
            KK,KF,KS,KM,KDS,KDV,KN,IUSEL,EPSIO,SO,MAT,EXTRA,
            GPAXEL,NDIR,NDEG,ICONT)
```

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

a) Variables		b) Definitions
T1718 (24,24)	-	Transformation matrix to 3648 notation
TTOBL (3,12)	-	Transformation matrix from local to global or oblique coordinates
SEL(17,24)	-	Working area and stress matrix
ANELX (300)	}	Incremental matrices in cylinder notation
ANELY (300)	}	
ANELXY (300)	}	
ANELEX (300,3)	-	Incremental matrices for Instability in 3648 notation

12. Subroutine User: ELPLUG

13. Subroutines Required:

DIREC	BCB
BCB12	MATB
FTELQ	MAB
SELQ	
KOBLIQ	

14. Remarks:

a) Controls are reset in programs to compute everything but the incrementals. Initial test phase had KN = 1 to check these computations.

b) Plug not tested out if either the flexural or membrane thickness is zero (certain portions of plug will be bypassed as LAMDA is set = 0).

c) Thermal load will probably have to be rederived as 2nd input TEMP is thermal moments M^x and not the thermal gradient as prescribed for flexural elements (PLUGS 1 and 2).

d) References:

(1) Bell Report No. D2114-95005, "Derivation of the Force - Displacement Properties of Triangular and Quadrilateral Orthotropic Plates in Plane Stress and Bending" - Gallagher, Huff - dated Jan. 1964.

(2) Bell Report No. D2114-95008, "Detailed Description - Computer Program for Stiffened Cylinder Analysis" - Gallagher, Huff, Dale - dated Jan. 1964.

1. Subroutine Name: TR18ST

2. Purpose: To form transformation for stress and thermal stress matrices to u, v, w notation

3. Equations and Procedures: See element write-up for defined transformations.

4. Input Arguments:

   NODE - element nodes

5. Output Arguments:

   T1718
   T18ST - transformation matrices

6. Error Returns: None

7. Calling Sequence:

   Call TR18ST(NODE,T1718,T18ST)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is 352_{16} Bytes.

12. Subroutine User: PLUG18

13. Subroutine Required: None

14. Remarks: None

1. Subroutine Name: FBMP18

2. Purpose: To evaluate B matrix for quadrilateral plate elements; out of plane.

3. Equations and Procedures: See element write-up for definition of B matrix generation.

4. Input Arguments:

   XEL - X coordinates
   YEL - Y coordinates

5. Output Arguments:

   B - output matrix

6. Error Returns: None

7. Calling Sequence:

   Call FBMP18(XEL,YEL,B)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is $39A_{16}$ Bytes.

12. Subroutine User: PLUG18

13. Subroutine Required: None

15. Remarks: NOne

1. Subroutine Name: PLUG22

2. Purpose: Element matrix generation for the incremental frame element.

3. Equations and Procedures: None

4. Input Arguments:

IPL	-	plug number
NNO	-	number of node points
XC	-	X-coordinates of nodes points
YC	-	Y-coordinates of node points
ZC	-	Z-coordinates of node points
TEL	-	temperatures at the node points
PEL	-	pressures at the node points
QS	-	input displacements of the node points
IP	-	number of extra cards
NORD	-	order of element stiffness matrix
NRSEL	-	number of rows in the stress matrix
NN	-	number of nodes
NL	-	node point numbers
KK	-	code for computation of element stiffness matrix
KF	-	code for computation of element thermal load
KS	-	code for computation of element stress matrix
KM	-	code for computation of element mass matrix
KTS	-	code for computation of element thermal stress matrix
ET	-	code for computation of structural damping matrix
KVM	-	code for computation of viscous damping matrix
KN	-	code for computation of incremental damping matrix
IUSEL	-	dummy
EPS	-	pre-strain load vector
SO	-	pre-stress load vector
MAT	-	material properties matrix
EXTRA	-	extra information (angles, etc.)
NDIR	-	number of directions of movement per grid point
NDEG	-	number of types of movement allowed per grid point
ICONT	-	code for use of grid point axes

5. Output Arguments:

NERR	-	error return
NOINK	-	number of elements in lower half matrices
KSEL	-	stiffness matrix
CNX	-	incremental stiffness matrix
FTEL	-	thermal load + pressure load matrix
SEL	-	stress matrix

5. Output Arguments (Contd):

SZALEL	-	therma- stress matrix
AMASS	-	mass matrix
DAMPV	-	viscous damping matrix
DAMPS	-	structural damping matrix
NSEL	-	number of elements in stress matrix
NMASS	-	number of elements in mass matrix
NDMPV	-	number of elements in viscous damping matrix
NDMPS	-	number of elements in structural damping matrix
GPAXEL	-	grid point axes transformation

6. Error Returns:

NERR = 0 no error
= 1 plug number incorrect
= 2 number of nodes incorrect
= 3 number of input points incorrect
= 4 order of matrix (NORD) incorrect

7. Calling Sequence:

```
Call PLUG22(IPL,NNO,XC,YC,ZC,TEL,PEL,QS,IP,NORD,NERR,NOINK,
            KSEL,CNX,FTEL,SEL,SZALEL,AMASS,DAMPV,DAMPS,NRSEL,
            NN,NL,NMASS,NDMPV,NDMPS,NSEL,KK,KF,K8,KM,KTS,ET,
            KVM,KN,IUSEL,EPSIO,SO,MAT,EXTRA,GPAXEL,NDIR,NDEG,
            ICONT)
```

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is $3FCC_{16}$ Bytes.

12. Subroutine User: ELPLUG

13. Subroutines Required:

ELTEST, CTS, CTCQ, CECC, MAB, AXTRA2,SYMPRT, BCB, MATB, MSB, FINP22,SQRT

14. Remarks: None

1. Subroutine Name: FINP22

2. Purpose: To form the incremental matrix for the incremental frame element.

3. Equations and Procedures:

AIN(10) = (L)*A
AIN(14) = (L2)*A
AIN(15) = (4*L3/3)*A
AIN(19) = (L3)*A
AIN(20) = (3*L4/2)*A
AIN(21) = (9*L5/5)*A
AIN(36) = (L)*A
AIN(44) = (L2)*A
AIN(45) = (4*L3/3)*A
AIN(53) = (L3)*A
AIN(54) = (3*L4/2)*A
AIN(55) = (9*L5/5)*A
All other values of AIN are zero

4. Input Arguments:

$L = X^2 + Y^2 + Z^2$

L1 = 1/L

$L2 = L^2$

$L3 = L^3$

$L4 = L^4$

$L5 = L^5$

PRINT = print control
A = area of member (A)

5. Output Arguments:

AIN - incremental matrix

6. Error Return: None

7. Calling Sequence:

Call FINP22(L,L2,L3,L4,L5,AIN,PRINT,A)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is $3EC_1$, Bytes.

12. Subroutine User: PLUG22

13. Subroutine Required: None

14. Remarks: None

1. Subroutine Name: AXTRA1

2. Purpose: Apply grid point axes transformation by pre-multiplication using either the actual transformation matrix or its transpose.

3. Equations and Procedures:

$$[M_{OUT}] = [\Gamma_{GPA}]\,[M_{IN}] \quad \text{or} \quad [M_{OUT}] = [\Gamma_{GPA}]^{T}\,[M_{IN}]$$

where $[M_{IN}]$ is the input element matrix,

$[\Gamma_{GPA}]$ is the element grid point axes transformation matrix,

$[M_{OUT}]$ is the output transformed element matrix.

$[M_{OUT}]$ is stored in the same location as $[M_{IN}]$, therefore the input element matrix is lost once the multiplication has been effected. Advantage is taken, during multiplication, of the fact that $[\Gamma_{GPA}]$ is structured as a set of (3 x 3) or (2 x 2) matrices with main diagonal positions lying on the main diagonal of $[\Gamma_{GPA}]$.

4. Input Arguments:

GPAXEL : Element grid point axes transformation matrix, $[\Gamma_{GPA}]$
QSEL : Input element matrix, $[M_{IN}]$
NCOL : Number of columns in QSEL
NNO : Number of element node points
NDEG : Number of degrees of freedom
NDIR : Number of directions
ITRAN : Control code, if ITRAN = 0, then $[M_{OUT}] = [\Gamma_{GPA}]\,[M_{IN}]$
if ITRAN = 1, then $[M_{OUT}] = [\Gamma_{GPA}]^{T}[M_{IN}]$

5. Output Arguments:

QSEL : Output transformed element matrix, $[M_{OUT}]$

6. Error Returns: None

7. Calling Sequence:

CALL AXTRA1 (GPAXEL, QSEL, NCOL, NNO, NDEG, NDIR, ITRAN)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage required is $41E_{16}$ Bytes.

    COL (3)
    ISAVE (3)

12. Subroutine User: ELPLUG

13. Subroutines Required: None

14. Remarks: The output matrix is stored in the input matrix storage.

1. Subroutine Name: AXTRA3

2. Purpose: Apply grid point axes transformation by triple product multiplication.

3. Equations and Procedures:

$$[AN] = [\Gamma GPA]^T * [SYM] * [\Gamma GPA]$$

where

$[\Gamma GPA]$ is the element grid point axes transformation matrix

$[SYM]$ is symmetric input element matrix

$[AN]$ is symmetric output transformed element matrix

The triple product is obtained by computing a row of the intermediate product of $[\Gamma GPA]^T * [SYM]$ and then multiplying this intermediate row with $[\Gamma GPA]$ to obtain a row in $[AN]$. Advantage is taken, during multiplication, of the facts that $[SYM]$ and $[AN]$ are symmetric and also that $[\Gamma GPA]$ is structured as a set of (3x3) or (2x2) matrices with main diagonal elements lying on the main diagonal of $[\Gamma GPA]$.

4. Input Arguments:

GPAXEL	: Element grid point axes transformation matrix, $[\Gamma GPA]$
SYM	: Input element matrix, symmetric, singly subscripted, stored lower half by rows, $[SYM]$
NCOL	: Number of columns in SYM (also number of rows in SYM)
NNO	: Number of element node points
NDEG	: Number of degrees of freedom
NDIR	: Number of directions

5. Output Arguments:

AN	: Output transformed element matrix, symmetric, singly subscripted, stored lower half by rows, $[AN]$

6. Error Returns: None

7. Calling Sequence: Call AXTRA3

   (GPAXEL, SYM, AN, NCOL, NNO, NDEG, NDIR)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage:

    ROW(48)
    Total Storage = $517_8 = 335_{10}$.

12. Subroutine User: ELPLUG

13. Subroutines Required: LOC

14. Remarks:

    SYM must be stored lower half by rows,
    AN will be stored lower half by rows.

    Internal intermediate storage in variable ROW is dimensioned 48. If the order of [SYM] is greater than 48, an appropriate increase must be made in this intermediate storage.

1. Subroutine Name: ELPRT

2. Purpose: Print generated element matrices.

3. Equations and Procedures: Non-suppressed matrices are printed, complete with titles.

4. Input Arguments:

NOINK	-	Number of storages in element stiffness, incremental stiffness and mass matrices
AKEL	-	Array containing element stiffness matrix
NORD	-	Number of element degrees of freedom
FTEL	-	Vector containing element load matrix
NNO	-	Number of element defining points
NODES	-	Array containing element defining grid point numbers
NSEL	-	Number of storages in element stress matrix
NRSEL	-	Element stress order
SEL	-	Array containing element stress matrix
SZALEL	-	Vector containing element thermal stress matrix
ANEL	-	Array containing element incremental stiffness matrix
INEL	-	Element number
NMASS	-	Number of storages in element mass matrix
AMASS	-	Array containing element mass matrix
NDMPV	-	Not used
DAMPV	-	Not used
NDMPS	-	Not used
DAMPS	-	Not used
ILP	-	Element type code number
NUMOT	-	Number of output matrices
NAMOUT	-	Array containing output matrix names

5. Output Arguments: None

6. Error Returns: None

7. Calling Sequence:

```
CALL ELPRT (NOINK,AKEL,NORD,FTEL,NNO,NODES,NSEL,NRSEL,SEL,
           SZALEL,ANEL,INEL,NMASS,AMASS,NDMPV,DAMPV,NDMPS,
           DAMPS,ILP,NUMOT,NAMOUT)
```

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

Total Storage required is $A3C_{16}$ Bytes.

12. Subroutine User: ELPLUG

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: OUTMAT

2. Purpose: Sort element matrices on scratch tape and output to Format Execution Monitor in an optimal manner

3. Equations and Procedures: First the array controlling the selection and order of output of the matrices (IKNOW) is formed. The IKNOW array will contain the pass number on which each computed output matrix will be written on an output tape. Correspondence between the IKNOW array and the output matrices is as follows:

IKNOW(6)	: Transformation assembly matrix (TA)
IKNOW(7)	: Master element stiffness matrix (KEL)
IKNOW(8)	: Master element applied load matrix (FTEL)
IKNOW(9)	: Master element stress matrix (SEL)
IKNOW(10)	: Master element thermal stress matrix (SZALEL)
IKNOW(11)	: Master element incremental matrix (N)
IKNOW(12)	: Master element mass matrix (M)

For each output matrix except the element applied load and thermal stress matrices the following procedure takes place:

a. If the matrix has not been calculated, as determined by a slash in its position in the NAMOUT array, its position in the IKNOW array is set equal to zero.

b. If the matrix has been calculated, then its corresponding output tape number is obtained from the IOSPEC array and a search is done from the beginning of the IOSPEC array to this matrice's position, counting the number of times this tape number has been encountered. This final count is the pass number on which this matrix will be written and is placed into the matrix's corresponding position in the IKNOW array.

After the IKNOW array has been formed it is searched for the greatest number. This number will be the number of passes required to output all of the computed matrices.

On each pass the following procedure is used. The scratch tape containing the element matrices is rewound. This tape consists of two records per element. The first record contains element definition data, the second contains the matrices for that element. The second record is read into a dynamic storage area and interpreted by locating key numbers that appeared in the record. A loop is entered from one to NELEM. The contents of the IKNOW array are

compared to the pass number. When a match is found the corresponding matrix is written on its output tape. Before writing the first element's contribution on its output tape, the appropriate matrix header is written. In most cases the matrices will be output in compressed format. However, in small applications when the maximum element order (NORDM) or the maximum element stress order (NRSELM) is greater than one-half the sum of the element orders (NORSUM) or the element stress orders (NRSSUM), respectively, then the matrices will be output in uncompressed format. A count is maintained in IR and IC for each output matrix in order to place each element's contribution in the correct position in the output matrix. At the end of the pass the appropriate matrix trailer and data set trailer labels are written. The TA matrix is a special case in that it is generated from the element definition data and then placed on its output tape. For output of the element applied load and element thermal stress matrices the following procedure is invoked. During the first pass of the tape, if they were not suppressed, the element applied load and thermal stress matrices were stored in the blank common work area. Following the first pass these two matrices are output in either compressed or uncompressed format, dependent upon the same criteria as all other matrices.

4. Input Arguments:

NUMOT	: Number of output matrices
NAMOUT	: Names of output matrices
IOSPEC	: Unit information regarding output matrices
NTAP3	: Scratch tape containing system information
NTAP4	: Scratch tape containing element matrices
NSYS	: System order
NTD	: Number of degrees of freedom per grid point
NORSUM	: Summation of element orders
NRSSUM	: Summation of element stress rows
NELEM	: Number of elements
NWORKR	: Number of working storages available
WORK	: Common work area
NORDM	: Maximum element order
NRSELM	: Maximum element stress order

5. Output Arguments: None

6. Error Returns: None

7. Calling Sequence:

```
CALL US460(NUMOT, NAMOUT, IOSPEC, NTAP3, NTAP4, NSYS,
NTD, NORSUM, NRSSUM, NELEM, NWORKR, WORK, NØRDM, NRØELM)
```

8. Input Tapes:

NTAP3 : Contains system information
NTAP4 : Contains element matrices in compact form

9. Output Tapes: Output tape units are supplied by the Format Execution Monitor; matrices are output by columns in compressed format. Appropriate matrix header and trailer labels are written. An output matrix consists of all the element matrices of that type placed such that their main diagonal positions lie on the main diagonal of the output matrix in succeeding positions.

10. Scratch Tapes: None

11. Storage Required:

Total Storage required is $136E_{16}$ Bytes.

12. Subroutine User: US04B

13. Subroutines Required:

US461
US462
US463

14. Remarks: None

1. Subroutine Name: US461

2. Purpose: Write a column of an output matrix in uncompressed or compressed format.

3. Equations and Procedures: If KODE is zero, the IWORK array has NSUM zeros placed into it. Then, starting at ISTART, NROW's of ISTORE are placed into the corresponding positions in IWORK. The variable NUM = NSUM is the number of words from IWORK that will be written on tape. If KODE is one, each element of ISTORE is compared to zero. If it is zero, it is ignored. If the element is not zero, then it is placed in the IWORK array in the first unused position and the next position in IWORK is filled by the row number in the output matrix of the non-zero element. The row number is corrected by ISTART in order to place the contribution in the correct row of the output matrix. NUM is a counter used to record the number of non-zero numbers found and the number of words that will be written from IWORK (NUM = 2* number of non-zero elements in ISTORE).

4. Input Arguments:

ISTORE	: Matrix column to be written
ICOL	: Column number of ISTORE in matrix
ISTART	: Starting row number in output matrix
NROW	: Number of rows in ISTORE
NTAPE	: Output tape number
IWORK	: Work area for compression of ISTORE
KODE	: Determines whether matrix is to be put into compressed form
NSUM	: Sum of element orders

5. Output Arguments: None

6. Error Returns: None

7. Calling Sequence: Call US461

(ISTORE, ICOL, ISTART, NROW, NTAPE, IWORK, KODE, NSUM)

8. Input Tapes: None

9. Output Tape: NTAPE

Record format is ICOL, KODE, NUM, (IWORK(I), I=1, NUM) where ICOL is column number, KODE equals one or zero, NUM is number of words remaining in record and IWORK is the compressed or uncompressed version of ISTORE. Each record then contains NUM + 3 words.

10. Scratch Tapes: None

11. Storage Required: Total storage required is $3E0_{16}$ Bytes.

12. Subroutine User: US460

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: US462

2. Purpose: Create a list which defines the location of the contributions of an element to the assembly transformation matrix.

3. Equations and Procedures: The degrees of freedom for each node point, with respect to the system of grid points, are calculated and placed in LIST. LIST is therefore of length NNO*NTD. The formula for determining this location is:
LIST(K) = (NODES(I) - 1)*NTD + L

where K = 1, 2, ..., NNO*NTD
I = 1, 2, ..., NNO
L = 1, 2, ..., NTD

4. Input Arguments:

NNO - number of element node points
NODES - array containing element node point numbers
NTD - number of degrees of freedom per grid point

5. Output Arguments:

LIST - array containing row number in TA matrix for each degree of freedom for each element node point.

6. Error Returns: none

7. Calling Sequence: CALL US462 (NNO, NODES, NTD, LIST)

8. Input Tapes: none

9. Output Tapes: none

10. Scratch Tapes: none

11. Storage Required: total storage required is $1FO_{16}$ Bytes.

12. Subroutine User: US460

13. Subroutines Required: none

14. Remarks: none

1. Subroutine Name: US463

2. Purpose: Obtain full column from symmetrically stored matrix

3. Equations and Procedures: For a symmetric matrix column is equivalent to row. The corresponding row to ICOL is located and the elements of that row up to and including the diagonal element are placed in the first and succeeding position of COL. If ICOL was the last column of the matrix the process is complete and control is returned to the calling program. If ICOL was not the last column then each element in the ICOL position of the remaining rows is placed into COL and control is returned to the calling program.

4. Input Arguments:

   SYM symmetric matrix stored lower half by rows, singly subscripted
   N order of SYM
   ICOL Column number of SYM desired

5. Output Arguments;

   COL - full column number ICOL

6. Error Returns: none

7. Calling Sequence: CAL US463 (SYM, N, ICOL, COL)

8. Input Tapes: none

9. Output Tapes: none

10. Scratch Tapes: none

11. Storage Required: Total storage required is $29A_{16}$ Bytes.

12. Subroutine User: US460

13. Subroutines Required: none

14. Remarks: none

1. Subroutine Name: ELMAT

2. Purpose: To output as a format matrix element matrices in compressed form to be used by structural modules outside of the USER04 module.

3. Equations and Procedures: The tape containing the information generated by subroutines REC3 and REC4 is read and then merged to form one record on the output tape for each element. The record written for each element is as follows:

```
JCOL,KODE,NUM,IEL,IPL,NORD,(LISTEL(I),I=1,NORD)
                       NOINK,(AKEL(I),I=1,NOINK)
                       NORD,(FTEL(I),I=1,NORD)
                       NNO,(NODES(I),I=1,NNO)
                       NSEL,(SEL(I),I=1,NSEL)
                       NRSEL,(SZALEL(I),I=1,NRSEL)
                       NOINK,(ANEL(I),I=1,NOINK)
                       NMASS,(AMASS(I),I=1,NMASS)
```

where,

JCOL	-	is the column number
KODE	-	is equal to 0 to indicate non-compression
NUM	-	is the number of words remaining in the record NUM=2*NOINK+2*NORD+NNO+NSEL+NRSEL+NMASS+10
IEL	-	is the element number
IPL	-	is the element type (plug number)
LISTEL	-	a list array used to reorder the system degrees of freedom of length NORD
AKEL	-	the element stiffness matrix of length NOINK
FTEL	-	the element thermal load matrix of length NORD
NODES	-	an array containing the reference points for the element of length NNO
SEL	-	the element stress matrix of order NSEL
SZALEL	-	the element thermal stress matrix of order NRSEL
ANEL	-	the element incremental matrix of order NOINK
AMASS	-	the element mass matrix of order NMASS

4. Input Arguments:

NELEM	-	number of elements in analysis
MAXELM	-	length of maximum element record
NAME	-	array containing name of output matrix
NSET	-	data set number of output matrix
NTAPE	-	data set number of input element tape
NWORK	-	number of words of work storage available
MAT	-	work storage for reading NTAPE element data

5. Output Arguments: None

6. Error Returns: None

7. Calling Sequence:

   ELMAT(NELEM,MAXELM,NAME,NSET,NTAPE,NWORK,MAT)

8. Input Tapes: NTAPE

9. Output Tape: NSET

10. Scratch Tapes: None

11. Storage Required: Total Storage required is $6F2_{16}$ Bytes.

12. Subroutine User: US04B

13. Subroutine Required:

    EUTL5
    EUTL6

14. Remarks: None

1. Subroutine Name: PLUG10

2. Purpose: Generate the following element matrices of a tetrahedron according to equations furnished by J. Batt.

   (a) Stiffness if KK $\neq$ 0.
   (b) Stress if KS $\neq$ 0.
   (c) Thermal load if KF $\neq$ 0.
   (d) Thermal Stress if KTS $\neq$ 0.
   (e) Mass if KM $\neq$ 0.

3. Equations and Procedures:

   (a) Zero out stiffness (AKEL3), stress (SEL3), thermal load (FEL3) and thermal stress (SIGEL) matrices.

   (b) Call subroutine VOL to find volume.

   (c) Call subroutines to form [EM] and [B] matrices.

   (d) Form TM and TMC matrices if pressure is present or mass matrix is to be calculated

   (e) If KTS and KF $\neq$ 0 form

      1) $\{\alpha\} = \{\alpha\Delta T + \varepsilon\}$

   If KTS $\neq$ 0, form

      2) $\{S^{\alpha}\} = [EM]\{\alpha\}$

   (f) If KK, KS, KF = 0, go to (i), otherwise do the following

      $[SEL3] = [EM][B]$

   (g) If KK $\neq$ 0, form

      1) $[AKEL] = [B]^T[SEL3] = [B]^T[EM][B]$

      2) Call MSTR to store [AKEL] into [AKEL3].

   (h) If KF $\neq$ 0, form

      1) $\{FTH\} = [EM][B]\{\alpha\} = [SEL3]\{\alpha\}$

      2) Form Pressure load matrix according to new equations: {FP} → (See P10FP writeup)

      3) Add {FPL} and {FEL} to form {FEL3}.

   (i) If KM$\neq$0, form

      1) Call CMASS to form $[\tilde{M}]$ where $[\tilde{M}]$ is in the local u,v,w system

      2) Form $[AMASS] = [T_M]^T[\tilde{M}][T_M]$ in global system.

4. Input Arguments:

IPL - Plug number - 10
NNO - Number of nodes = 4
X,Y,Z - Coordinates of element
TEL,PEL - Temperature and pressure of element
NORD - Order of stiffness matrix
NODES - Nodes of element
KK,KF,KS,KTS,KM - Controls on calculations of stiffness, thermal load, stress, thermal stress and mass matrices
EPSLO,SO- Input strains and stresses
MAT - Material properties array
EXTRA - External data - not used.

5. Output Arguments:

AKEL3 - Stiffness matrix
FEL3 - Thermal load matrix
SEL3 - Stress matrix
SIGEL - Thermal stress matrix
AMASS - Consistent mass matrix
NRSEL - Number of rows in stress and thermal stress matrices = 6
NOINK,NMASS,NSEL - Number of elements in stiffness, mass and stress matrices (respectfully 78,78,72)

6. Error Returns: Standard error return by ELTEST

7. Calling Sequence:

```
Call PLUG10(IPL,NNO,X,Y,Z,TEL,PEL,QSEL,IP,NORD,NERR,NOINK,
            AKEL3,ANEL3,FEL3,SEL3,SIGEL,AMASS,DAMPV,DAMPS,
            NRSEL,INNO,NODES,NMASS,NDMPV,NDMPS,NSEL,KK,KF,
            KS,KTS,KM,KDS,KDV,KN,IUSEL,EPSLO,SO,MAT,EXTRA,
            GPAXEL,NDIR,NDEG,ICONT)
```

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

SIGEL(6), X(4), Z(4), EM(21), FEL3(6), TSTR(6), B(6,12), SEL3(6,12), AKEL(12,2), AKEL3(1), TEL(12,2), PEL(12,2), AMASS(1), EMASS(78), FPL(12), FP(12), TM(12,12), TMC(3,3), A(4,4), EPSLO(1), SO(1), MAT(1), EXTRA(1), NODES(1), Y(4)

12. Subroutine User: PLUG13

13. Subroutines Required:

ELTEST, VOL, TRAAE, BDEF, PlOTM, MPRD, TPRD, MSTR, PlOFP, CMASS, BCB, SYMPRT, TXOUT

14. Remarks:

PEL(1,1) is pressure of face 431
PEL(2,1) is pressure of face 432
PEL(3,1) is pressure of face 421
PEL(4,1) is pressure of face 321

See AFFDL-TR-72-42, Volume I, Engineer's Manual for PLUGlO equations.

1. Subroutine Name: VOL

2. Purpose:

   To set up a function to calculate the volume of the tetrahedron element

3. Equations and Procedures:

$$V = 1/6 \begin{vmatrix} 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \\ 1 & x_4 & y_4 & z_4 \end{vmatrix}$$

4. Input Arguments:

   X,Y,Z - coordinates of tetrahedron

5. Output Arguments:

   VOL - volume of tetrahedron

6. Error Returns: None

7. Calling Sequence:

   Function VOL (X,Y,Z)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Subroutine User: PLUG10

12. Subroutine Required: None

13. Remarks: None

1. Subroutine Name: CMASS

2. Purpose: Form the local consistent mass matrix.

3. Equations and Procedures:

(a) Compute the CONST = $\frac{\rho V}{20}$

(b) Consistent mass matrix and labeling is

$$\text{CONST} \begin{bmatrix} 2 & & & \\ 1 & 2 & & \\ 1 & 1 & 2 & \\ 1 & 1 & 1 & 2 \end{bmatrix} \qquad \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ v_3 \end{Bmatrix} \quad \text{etc.}$$

(c) Assemble above constants as follows:

$$\text{CONST} \begin{bmatrix} U_1 & V_1 & W_1 & U_2 & V_2 & W_2 & U_3 & V_3 & W_3 & U_4 & V_4 & W_4 \\ 2 & & & & & & & & & & & \\ & 2 & & & & & & & & & & \\ & & 2 & & & & & & & & & \\ 1 & & & 2 & & & & & & & & \\ & 1 & & & 2 & & & & & & & \\ & & 1 & & & 2 & & & & & & \\ 1 & & & 1 & & & 2 & & & & & \\ & 1 & & & 1 & & & 2 & & & & \\ & & 1 & & & 1 & & & 2 & & & \\ 1 & & & 1 & & & 1 & & & 2 & & \\ & 1 & & & 1 & & & 1 & & & 2 & \\ & & 1 & & & 1 & & & 1 & & & 2 \end{bmatrix}$$

4. Input Arguments:

   V - Volume of the tetrahedron
   RHO - Mass density

5. Output Arguments:

   CMAS- Element mass matrix in local reordered system

6. Error Returns: None

7. Calling Sequence:

   Call CMASS (V,RHO,CMAS)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: CMAS(1)

12. Subroutine User: PLUG10

13. Subroutine Required: None

14. Remarks: None

1. Subroutine Name: BDEF

2. Purpose: Form the coordinate and area matrices matrix

3. Equations and Procedures:

   (a) Form the A matrix from input coordinates.
   (b) Invert the A-matrix which gives projected areas of element
   (c) Form the B-matrix according to equations from the inverted A-matrix in U,V,W notation.

4. Input Arguments:

   X,Y,Z - Coordinates of element
   WIPR - Print option
   NRB,NCB - Number of rows and columns in B matrix

5. Output Arguments:

   A - Projected area matrix
   B - Coordinate matrix

6. Error Returns: None

7. Calling Sequence:

   Call BDEF(X,Y,Z,WIPR,NRB,NCB,A,B)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

    B(NRB,NCB), Z(1), Y(1), X(1)
    A(4,4), EL(4), EM(4)

12. Subroutine User: PLUG10

13. Subroutine Required: MINV

14. Remarks: None

1. Subroutine Name: P10TM

2. Purpose: Form the transformation matrix of direction cosines for the tetrahedron if pressures or mass is present.

3. Equations and Procedures:

$$e_{11} = x_3 - x_1 \quad , \quad e_{12} = y_3 - y_1 \quad , \quad e_{13} = z_3 - z_1$$

$$e_{31} = (y_3 - y_1)(z_2 - z_1) - (y_2 - y_1)(z_3 - z_1)$$

$$e_{32} = (x_2 - x_1)(z_3 - z_1) - (x_3 - x_1)(z_2 - z_1)$$

$$e_{33} = (x_3 - x_1)(y_2 - y_1) - (x_2 - x_1)(y_x - y_1)$$

$$e_{21} = e_{32}e_{13} - e_{12}e_{33}$$

$$e_{22} = e_{11}e_{33} - e_{31}e_{13}$$

$$e_{23} = e_{31}e_{12} - e_{11}e_{32}$$

$$\bar{e}_1 = (e_{11}^2 + e_{12}^2 + e_{13}^2)^{1/2}$$

$$\bar{e}_2 = (e_{21}^2 + e_{22}^2 + e_{23}^2)^{1/2}$$

$$\bar{e}_3 = (e_{31}^2 + e_{32}^2 + e_{33}^2)^{1/2}$$

$$[\tilde{T}_M] = \begin{bmatrix} \frac{e_{11}}{\bar{e}_1} & \frac{e_{12}}{\bar{e}_1} & \frac{e_{13}}{\bar{e}_1} \\ \frac{e_{21}}{\bar{e}_2} & \frac{e_{22}}{\bar{e}_2} & \frac{e_{23}}{\bar{e}_2} \\ \frac{e_{31}}{\bar{e}_3} & \frac{e_{32}}{\bar{e}_3} & \frac{e_{33}}{\bar{e}_3} \end{bmatrix}$$

Then store $[\widetilde{T}_M]$ into $[T_M]$ as

$$[T_M] = \left[\begin{array}{c|c|c|c} [\widetilde{TM}] & 0 & 0 & 0 \\ \hline 0 & [\widetilde{TM}] & 0 & 0 \\ \hline 0 & 0 & [\widetilde{TM}] & 0 \\ \hline 0 & 0 & 0 & [\widetilde{TM}] \end{array}\right]$$

4. Input Arguments:

   X, Y, Z - Input x,y,z coordinates
   WIPR - Print option

5. Output Arguments:

   $[\widetilde{T}_M]$ - Point transformation matrix

   [TM] - Element transformation matrix

6. Error Returns: None

7. Calling Sequence:

   Call PlOTM(X,Y,Z,WIPR,TMC,TM)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

    X(1), Y(1), Z(1), TMC(3,3), TM(12,12), E1(3), E2(3), E3(3), EB(3)

12. Subroutine User: PLUG10

13. Subroutine Required: None

14. Remarks: None

1. Subroutine Name: P10FP

2. Purpose: Form the local pressure matrix in the u,v, and w notation.

3. Equations and Procedures:

(a) The equation for the local pressure load is

$$\{F_{p\ell}\} = [T]^T\{\tilde{F}_p\} + [T_1]^T\{\tilde{F}_{p1}\} + [T_2]^T\{\tilde{F}_{p2}\}$$

(b) Procedure:

1. Noting the [T] is a reordering matrix to u, v, w notation only, the product of

$$[T]^T\{\tilde{F}_p\}$$

in the reordered system is

$$\begin{Bmatrix} FP_{u1} \\ FP_{v1} \\ FP_{w1} \\ FP_{u2} \\ FP_{v2} \\ FP_{w2} \\ FP_{u3} \\ FP_{v3} \\ FP_{w3} \\ FP_{u4} \\ FP_{v4} \\ FP_{w4} \end{Bmatrix} = \begin{Bmatrix} 0 \\ p_1 \\ p_2 - p_3 \\ 0 \\ 0 \\ p_2 \\ 0 \\ p_1 \\ p_2 - p_3 \\ 0 \\ p_1 \\ -p_3 \end{Bmatrix}$$

Where:

$$p_1 = \frac{1}{3} p_{431} A_{431} \cos\theta$$

$$p_2 = \frac{1}{3} p_{321} A_{321}$$

$$p_3 = +\frac{1}{3} p_{431} A_{431} \sin\theta$$

$$\theta = \tan^{-1}\left(\frac{y_4}{z_4}\right)$$

$$\begin{Bmatrix} x_4 \\ y_4 \\ z_4 \end{Bmatrix} = [\tilde{T}_M] \begin{Bmatrix} XEL(4) \\ YEL(4) \\ ZEL(4) \end{Bmatrix}$$

2. Noting that T_1 contains 4 pairs of 3 constants in the 12 x 4 matrix, the contribution here becomes

$$\left\lfloor FP_{x1}, FP_{y1}, \text{etc.} \right\rfloor^T = \left\lfloor 0,0,0,c_1,c_2,c_3 \middle| c_1,c_2,c_3 \middle| c_1,c_2,c_3 \right\rfloor^T$$

where $c_1 = pT_1$

$c_2 = pT_2$

$c_3 = pT_3$

$p = \frac{1}{3} p_{432} A_{432}$

T_1, T_2, T_3 are typical direction cosines

3. Noting that T_2 is similar to T_1 (just different values of T_1, T_2, T_3), the contribution becomes

$$\left[FP_{x1}, FP_{y1}, \text{etc.}\right]^T = \left[c_1, c_2, c_3, c_1, c_2, c_3, 0, 0, 0, c_1, c_2, c_3\right]^T$$

where $c_1 = pT_1$

$c_2 = pT_2$

$c_3 = pT_3$

$p = \frac{1}{3} p_{421} A_{421}$

T_1, T_2, T_3 are typical direction cosines

4. Input Arguments:

PEL - Pressures on faces of tetrahedron
B - Matrix which contains local areas
XEL,YEL,ZEL - Coordinates of element
TMC - Transformation matrix
A - Area matrix
IPRT - Print option

5. Output Arguments:

FPL - Pressure load
FP - Working pressure array

6. Error Returns: None

7. Calling Sequence:

Call P10FP(PEL,XEL,YEL,ZEL,TMC,A,IPRT,FPL,FP)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

PEL(12,2), TMC(3,3), A(4,4), SA(3), SMA(3), FP(12), XEL(1), YEL(1), ZEL(1), X(3), XL(3), FPL(1)

12. Subroutine User: PLUG10

13. Subroutines Required: MPRD

14. Remarks:

p431 is pressure on face 431 = PEL(1,1)
p432 is pressure on face 432 = PEL(2,1)
p421 is pressure on face 421 = PEL(3,1)
p321 is pressure on face 321 = PEL(4,1)

A_{431} is total projected area of face 431
A_{432} is total projected area of face 432
A_{421} is total projected area of face 421
A_{321} is total projected area of face 321

1. Subroutine Name: PLUG13

2. Purpose: To form the element stiffness, stress, thermal load, thermal stress and mass matrices for the triangular prism.

3. Equations and Procedures:

The triangular prism is composed of 3 tetrahedrons:

A. For each tetrahedon, the following is done.

(1) Pick up coordinates, pressures, nodes, temperatures of the tetrahedron.

(2) Call PLUG10 to generate the appropriate element matrices the tetrahedron.

(3) Assemble the tetrahedron contribution to the stiffness and thermal load matrices of the prism.

(4) Assemble the tetrahedron contribution to the stress and thermal stress matrices of the prism.

B. If symmetric wing analysis (Nodes 4,5,6 = 0), form a transformation matrix and transform the generated element matrices of the prism for the reduced system.

4. Input Arguments:

IPL - Plug number = 13
NNO,NODES - Number of nodes and node points of elements
X,Y,Z,TEL,PEL - Coordinates, temperatures and pressures of element
KK,KF,KS,KTS,KM- Controls on stiffness, thermal load, stress, thermal stress and mass calculations
EPSLO,SO- Initial strains and stresses
MAT - Material properties array

5. Output Arguments:

NORD - Order of stiffness
NOINK,NMASS - Number of terms in stiffness and mass matrices
NRSEL,NSEL - Number of rows and terms in stress matrix
AKEL3,FEL3,SEL3,SIGEL,AMAS1 - Element stiffness, thermal load, stress, thermal stress and mass matrices

6. Error Returns: Standard returns by ELTEST

7. Calling Sequence:

```
Call PLUG13 (IPL,NNO,X,Y,Z,TEL,PEL,QSEL,IF,NORD,NERR,
             NOINK,AKEL3,ANEL3,FEL3,SEL3,SIGEL,AMAS1,
             DAMPV,DAMPS,NRSEL,INNO,NODES,NMASS,NDMPV,
             NDMPS,NSEL,KK,KF,KS,KTS,KM,KDS,KDV,KN,IUSEL,
             EPSLO,SO,MAT,EXTRA,GPAXEL,NDIR,NDEG,ICONT)
```

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

SIGEL(18), X(6), Z(6), FEL3(18), SEL3(18,18), AKEL3(171), NODES(1), Y(6), EXTRA(1), EPSLO(1), SO(1), TEL(12,2), PEL(21,2), MAT(1), LISTA(4,3), LISTB(12,3), LIST(12), AMAS1(1), DUM(18), XX(4), YY(4), ZZ(4), NODESZ(4), TTL(12,2), PPL(12,2), EXTPR(6), AKEL3Z(78), FEL3Z(12), SEL3Z(6,12), SIGEL(6), AMASS(78), TR(18,9), SEL3T(18,9), AKELR(45), FELR(9)

12. Subroutine User: ELPLUG

13. Subroutines Required:

ELTEST, PLUG10, POOF, TXOUT, SYMPRT, BCB, MPRD, TPRD

14. Remarks:

PEL(1,1) is pressure on face 123
PEL(2,1) is pressure on face 456
PEL(3,1) is pressure on face 2365 }
PEL(4,1) is pressure on face 1364 } has 2 tetrahedrons
PEL(5,1) is pressure on face 2541 }

1. Subroutine Name: PLUG20

2. Purpose: Generate stiffness, stress, thermal force, thermal stress matrices and mass matrices for the rectangular prism.

3. Equations and Procedures:

A. Set constants and if nodes (9)≠0, then constant strain approach is used for thermal stress and thermal load matrices.
B. Zero out element matrices.
C. Call GEOMD to calculate lengths and transformation matrix of element.
D. Call TRAAE to generate [EM]. [KCPSYM]
E. Generate integrals and form K-curl - prime from integrals.
F. Call GAMMAG to form reordering transformation matrix.
G. Call GAMMAS to form local to global transformation.
H. Form final transformation matrix [GPGGS].
I. If KK≠0, then

(i) Form [KCURL] = $1/(8abc)^2$ [KCPSYM]

(ii) Form [AK] = $[GPGGS]^T[KCURL][GPGGS]$.

J. If KS≠0, or constant strain approach is used, then
(i) Call subroutine to form [DISP]

(ii) Generate [STR] = [EM][DISP][GPGGS].

K. If KM=Ø, then
(i) Call MASS20 to form local mass in [TEMP]

(ii) Generate [MASXP] = $[GPGGS]^T[TEMP][GPGGS]$.

L. If KTS and KF≠0, then call P20FT to form thermal load and thermal stress.

4. Input Arguments:

IPL - PLUG Number = 20
NNO - Number of node points = 8
XC,YC,ZC - Coordinates of element
TEL - Temperatures at grid points
NODES - Node points of element
NPSL - Number of input strains = 6
NPSS - Number of input stresses = 6
KK,KM,KS,KF,KTS - Controls to calculate stiffness, mass, stress, thermal load, and thermal stress matrices.
EPSLON - Initial input strains
MAT - Material properties arrays
EXTRA - External data -- not used

5. Output Arguments:

AK,STR,FTXP,TS,MASXP - Stiffness, stress, thermal load, thermal stress and mass matrices
NOINK,NMASS - Number of elements in stiffness and mass matrices = 300
NRSEL - Number of rows in stress matrix = 6
NSEL - Number of elements in stress matrix

6. Error Returns:

NERR - Set if wrong plug or bad input data

7. Calling Sequence:

```
Call PLUG20(IPL,NNO,XC,YC,ZC,TEL,PEL,QSEL,IP,NORD,NERR,NOINK,
            AK,ANEL,FTXP,STR,TS,MASXP,DAMPV,DAMPS,NRSEL,INO,
            NODES,NMASS,NPSL,NPSS,NSEL,KK,KF,KS,KTS,KM,KDS,
            KDV,KN,IUSEL,EPSLON,SIGZER,MAT,EXTRA,GPAXEL,NDIR,
            NDEG,ICONT)
```

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

```
EM(21),AK(300),STR(144),MAT(1),MASXP(1),XIXYZ(64),XISYM(36),
KCPSYM(300),KCURL(300),GAMGGP(576),GAMGS(576),XC(1),YC(1),
ZC(1),TGS(9),XG(24),XS(24),TEMP(300),G2GGP(24,24),G2GS(24,24),
FTXP(1),TS(1),DISP(144),DELTAT(8),XYZ(108),ALPHAC(24,8),P(24,24),
TEL(12,2),BM(3,24),GPGGS(24,24),TEMP(24),EPSLON(1),SIGZER(1),
NODES(1),AMP(36),PEL(12,2),FP(24),EXTRA(1)
```

12. Subroutine User: ELPLUG

13. Subroutine Required:

```
ELTEST,GEOMD,TRAAE,MPOWJF,GAMMAG,GAMMAS,BCB,DISPMT,MPRD,
MASS20,P20FT
```

14. Remarks:

EXTRA(1)	is pressure on force 1234	x element pressures
(2)	" " " " 5678	
(3)	" " " " 1458	y element pressures
(4)	" " " " 2367	
(5)	" " " " 1256	z element pressures
(6)	" " " " 3478	

1. Subroutine Name: TRAAE

2. Purpose:
   form the elastic constant matrix [EM]

3. Equations and Procedures: None

4. Input Arguments:

ER,ETHETA,EZ	-	modulus of elasticity
VRO,VOZ,VZR	-	poisson's ratio
GRZ,GOZ,GZR	-	shear modulus
NORDER	-	order of matrix = 6

5. Output Arguments:

EM	-	elastic constants matrix

6. Error Returns: None

7. Calling Sequence:
   Call TRAAE (ER,ETHETA,EZ,VRO,VOZ,VZR,GRZ,GAM,TEO,EM,E,El, WIPR, NORDER, GOZ,GZR)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Subroutine Used: Several plugs

12. Subroutine Required: None

13. Remarks:
    TEO,GAM,E, and El are not used

1. Subroutine Name: GEOMD

2. Purpose: Generate geometric data from coordinates of the 8 node points of the rectangular prism element.

3. Equations and Procedures:

   A. Calculate A, B, and C half-edges from coordinates.
   B. Generate unit vectors along element X, Y and Z axis.
   C. Generate TGS element-to-system transformation matrix.
   D. Generate XG array of rotated element coordinates.

4. Input Arguments:

   XC - X-coordinates of the 8 element grid points in system coordinates
   YC - Same for Y-coordinates
   ZC - Same for Z-coordinates.

5. Output Arguments:

   A - Half-edge of prism along element X-axis
   B - Half-edge of prism along element Y-axis
   C - Half-edge of prism along element Z-axis
   TGS - Point transformation matrix for element: Rotates element axes parallel to system axes (3x3)
   XS - Array of system coordinates of grid points (3x8)
   XG - Array of (transformed) element coordinates of grid points (3x8)

6. Error Returns: None

7. Calling Sequence:

   Call GEOMD (XC,YC,ZC,A,B,C,TGS,XG,XS)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

    XC(8); YC(8); ZC(8)
    TGS(9); XG(24); XS(24) } In calling program

    RX(3); RY(3); RZ(3) — In subroutine

12. Subroutine User: PLUG20

13. Subroutines Required: MPRD

14. Remarks: None

1. Subroutine Name: GAMMAG

2. Purpose: Generate transformation matrix (24x24) to reorder variable sequence from $U_1,U_2,U_3,U_4,\ldots,U_8,V_1,V_2,V_3,\ldots,V_8,$ $W_1,W_2,W_3,\ldots,W_8$ to $U_1,V_1,W_1,U_2,V_2,W_2,\ldots,U_8,V_8,W_8$

3. Equations and Procedures:

   Store 1.0 in the appropriate location of the transformation matrix.

4. Input Arguments: None

5. Output Argume ts:

   GAMGGP - The desired matrix (24x24) stored singly - dimensioned, column-sequentially.

6. Error Returns: None

7. Calling Sequence:

   Call GAMMAG(GAMGGP)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

    GAMGGP(576) - In calling program

12. Subroutine User: PLUG20

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: GAMMAS

2. Purpose: Generate transformation matrix from element coordinates to system coordinates using the point-transformation matrix of element.

3. Equations and Procedures:

   See Engineer's Manual for mathematical formulation of Gamma-Sub=GS matrix.

4. Input Arguments:

   TGS - Point transformation matrix generated from geometric data by "GEOMD", stored singly dimensioned, column-wise (3x3).

5. Output Arguments:

   GAMGS- Element-to-system transformation matrix stored singly-dimensioned, column-sequentially

6. Error Returns: None

7. Calling Sequence:

   Call GAMMAS (GAMGS,TGS)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

    GAMGS(576), TGS(9) - In calling program

12. Subroutine User: PLUG20

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: DISPMT

2. Purpose: Generate strain-displacement (D) matrix for rectangular prism element with forces concentrated at the centroid.

3. Equations and Procedures:

   See Engineer's Manual for mathematical formulation of the D-matrix.

4. Input Arguments:

   A - Half-edge of prism along element X-axis
   B - Half-edge of prism along element Y-axis
   C - Half-edge of prism along element Z-axis

5. Output Arguments:

   DISP - The desired strain-displacement matrix (6x24), stored singly-subscripted, columnwise.

6. Error Returns: None

7. Calling Sequence:

   Call DISPMT(A,B,C,DISP)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

    DISP(144) - In calling program

12. Subroutine User: PLUG20

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: MPOWJF

2. Purpose: Print out a matrix, labeling rows and optionally labeling columns.

3. Equations and Procedures:

If number of columns≤8, print one-number column titles only:

```
Column
1 2 3 4 5 6 7 8
```

If 8<number of columns≤16, print two-number column titles only:

```
Column
1,9 2,10 3,11 4,12 5,13 6,14 7,15 8,16
```

If 16<number of columns≤24, print three-number columns titles:

```
Columns
1,9,17 2,10,18 3,11,19 4,12,20 5,13,21
6,14,22 7,15,23 8,16,24
```

Label rows:

```
Row 1 - - -
Row 2 - - - ... etc.
```

4. Input Arguments:

A - Matrix to be printed out - stored singly-dimensioned
N - Number of rows in A
M - Number of columns in A
L - Storage Code
  0 = General (column sequential)
  1 = Symmetric (lower symmetric row-wise or upper symmetric column-wise)
  2 = Diagonal (diagonal elements only)
ITITLE - Code for desired column titling
  0 = Do not title - label supplied externally (skip one line and start)
  1 = Title for 8 columns only
  2 = Title for 16 columns only
  3 = Title for 24 columns

5. Output Arguments: None

6. Error Returns: None

7. Calling Sequence:

   Call MPOWJF (A,N,M,L,ITITLE)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

    A( ) - Variable up to 576 - in calling program
    B(50) - In subroutine

12. Subroutine User: PLUG20

13. Subroutine Required: None

14. Remarks:

    This printout subroutine stands by itself, and may be used for general purpose writing of matrices with or without column titles. Built-in titles extend only up to 24 columns, but if no title is requested, there is no limitation on the number of rows or columns for a general or symmetric matrix. Diagonal matrices are limited to less than 50x50.

1 Subroutine Name: P20FP

2. Purpose: Form the pressure vector in local coordinates based upon pressures input on element level.

3. Equations and Procedures:

See Engineer's Manual for equations of the local pressure vector.

4. Input Arguments:

A,B,C - Lengths of sides of prism
PEL - Pressure array

5. Output Arguments:

FP - Local pressure vector

6. Error Returns: None

7. Calling Sequence:

Call P20FP (A,B,C,PEL,FP)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

PEL(12,2); FP(1)

12. Subroutine User: P20FT

13. Subroutine Required: None

14. Remarks: None

1. Subroutine Name: TXOUT

2. Purpose:
   Print element matrices dependent upon input data

3. Equations and Procedures: None

4. Input Arguments:

ICODE	-	title of matrix
A	-	matrices to be printed
N	-	number of rows
M	-	number of columns
MS	-	storage mode of matrix = 0 for general = 1 for symmetric = 2 for diagonal
LINS	-	number of lines/page = 60
IPOS	-	number of characters per line = 132
ISP	-	line spacing code = 1 for single space = 2 for double space

5. Output Arguments: None

6. Error Returns: None

7. Calling Sequence:
   Call TXOUT (ICODE,A,N,M,MS,LINS,IPOS,ISP)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Subroutine Used: Various Plugs

12. Subroutine Required: None

13. Remarks: None

1. Subroutine Name: P20FT

2. Purpose: Form the thermal load and thermal stress matrices for the rectangular prism dependent upon whether the element is at a constant or varying strain.

3. Equations and Procedures:

A. For Constant Strain Approach,

(1) Determine $\{\alpha\Delta T + \varepsilon\}$ where $\Delta T = T_{average} - T_0$

(2) If KTS $\neq$ 0 form thermal stress

$$\{TS\} = [EM]\{\alpha\Delta T + \varepsilon\}$$

(3) If KF $\neq$ 0, form temporary local thermal load

$$[P] = [DISP]^T[EM]$$

$$\{TEMP\} = [P]\{\alpha\Delta T + \varepsilon\}$$

B. For Varying Strain Approach,

(1) Form $\{\Delta T\} = \{TEMP_i - T_0\}$ where $TEMP_i$ is temperature of grid points

(2) Form $[\alpha] = \begin{bmatrix} \alpha_x [I] \\ \alpha_y [I] \\ \alpha_z [I] \end{bmatrix}$ where [I] is an 8^{th} order identity matrix

(3) Compute $\{TEMP1\} = [\alpha]\{\Delta T\}$

(4) Add initial strains (ε) to {TEMP1}

(5) If KTS $\neq$ 0, do the following for thermal stress

(a) Form [B]

(b) Compute {TEMP} = [B] {TEMP1}

(c) Compute thermal stress {TS} = [EM] {TEMP}

C. Call subroutine to form the loads due to pressure.

D. Add pressure load to local thermal load {TEMP}

E. Form global thermal load

$$\{FTXP\} = [GPGGS]^T\{TEMP\}$$

4. Input Arguments:

MAT - Material properties
EM - Material properties matrix
XYZ - Table of integrals
TEL,PEL - Temperature and pressures of element
A,B,C - Lengths of element
ICON - Control for constant or varying strain calculations
KF,KTS - Controls for thermal load and stress calculations
IPR - Print option
EPSLON - Input strains
DISP - Strain - displacement matrix
GPGGS - Transformation matrix to global system
CONST - Constant for thermal load

5. Output Arguments:

FTXP - global thermal load
TS - Thermal stress

6. Error Returns: None

7. Calling Sequence:

```
Call P20FT(MAT,EM,XYZ,TEL,PEL,A,B,C,ICON,KF,KTS,IPR,EPSLON,
           DISP,GPGGS,CONST,FTXP,TS,DELTAT,ALPHAC,P,BM,
           TEMP,TEMP1,FP)
```

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

DELTAT(1), MAT(1), EM(1), XYZ(1), ALPHAC(24,8), P(24,24), TEL(12,2), BM(3,24), PEL(12,2), EPSLON(1), DISP(6,24), GPGGS(24,24), FTXP(1), TS(1), TEMP(1), TEMP1(1), FP(1)

12. Subroutine User: PLUG20

13. Subroutines Required:

MPOWJF, MPRD, TPRD, P20FP, MATB

14. Remarks: None

1. Subroutine Name: MASS20

2. Purpose: Form the consistent mass matrix for the rectangular prism.

3. Equations and Procedures:

   (a) Form symmetric [AMP] according to equations in Engineer's Manual.

   (b) Assemble [AMP] into UU portion of [AMCURL] where [AMCURL] matrix can be partitioned as:

	u	v	w
u	[AMP]		
v	0	[AMP]	
w	0	0	[AMP]

   (c) Call subroutine to assemble [AMP] into VV and WW parts of [AMCURL].

4. Input Arguments:

   A,B,C - Dimensions of elements
   DENS - Density of material

5. Output Arguments:

   AMP - Consistent mass matrix [M]
   AMCURL - Full consistent mass matrix in local system

6. Error Returns: None

7. Calling Sequence:

   Call MASS20 (A,B,C,DENS,AMP,AMCURL)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: AMP(1); AMCURL(1)

12. Subroutine User: PLUG20

13. Subroutines Required:

   PASSYM - Assembles a small symmetric matrix into a larger symmetric matrix

14. Remarks: None

1. Subroutine Name: PASSYM

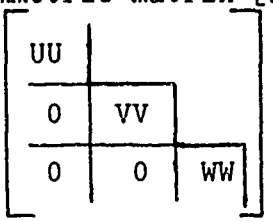
2. Purpose: Assemble a small symmetric matrix into a larger symmetric matrix.

3. Equations and Procedures:

Call Small symmetric matrix [CURL].

Call Large symmetric matrix [A] which is partitioned as follows:

[A] =



symmetric so only bottom half shown.

Symmetric Matrix [CURL] is assembled into UU, VV and WW parts.

4. Input Arguments:

NR - Number of rows to be assembled
IC,IR - Starting column and row of the assembled partition
i.e., 1,1 for UU; 2,2 for VV; and 3,3 for WW
[CURL]- Small symmetric matrix

5. Output Arguments:

[A] - Large assembled symmetric matrix

6. Error Returns: None

7. Calling Sequence:

Call PASSYM (NR,IC,IR,CURL,A)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: CURL(1), A(1)

12. Subroutine User: MASS20

13. Subroutine Required: None

14. Remarks: None

1. Subroutine Name: PLUG23

2. Purpose: Form the stiffness and stress matrices for a symmetric shear web.

3. Equations and Procedures:

A. Clear element matrices.

B. Compute length of panel and form direction cosines of element into [TR].

C. If KK $\neq$ 0, form the local stiffness matrix as

$$[K_R] = AC_1 \begin{bmatrix} 1 & & & \\ \frac{-2Z_1}{L} & 4\left(\frac{Z_1}{L}\right)^2 & & \\ \left(\frac{Z_1}{Z_2}\right) & \frac{-2Z_1^2}{LZ_2} & \left(\frac{Z_1}{Z_2}\right)^2 & \\ \frac{+2Z_1}{L} & -4\left(\frac{Z_1}{L}\right)^2 & \frac{2Z_1^2}{LZ_2} & 4\left(\frac{Z_1}{L}\right)^2 \end{bmatrix}$$

where

$$C_1 = \frac{tG_{xz}}{60(11Z_1^2 + 38Z_1Z_2 + 11Z_2^2)}$$

$$A = \frac{45L}{Z_1^2}(Z_1+Z_2)(Z_1^2+8Z_1Z_2+Z_2^2)$$

D. Form global stiffness matrix

$$[AK] = [TR]^T[K_R][TR]$$

E. If KS $\neq$ 0,

(a) Form global stress matrix [S]
(b) Form stress matrix into local system [SEL] = [S] [TR].

4. Input Arguments:

IPL - Element type = 23
NNO - Number of grid points defining element = 2
X,Y,Z - Coordinates of element
TEL,PEL - Temperature and pressure arrays - not used
NORD - Order of stiffness matrix = 6
NODES - Grid points of elements
KK,KF,KS,KTS,KM - controls on stiffness, thermal load, stress, thermal stress and mass calculations
EPSIO,SO- Initial stresses and strains-not used
MAT - Material properties array
EXTRA - Internal data containing thicknesses of web

5. Output Arguments:

NOINK - Number of elements in stiffness matrices
AK,SEL - Stiffness and stress matrices
FT,SIGEL- Thermal load and stress set = 0.0
AMASS - Mass matrix - set = 0.0
NRSEL,NSEL - Number of rows and number of elements in stress matrix
NMASS - Number of terms in null mass matrix

6. Error Returns:

If length = 0 or neg - error

If Z_1 or Z_2 = 0 or neg - error

7. Calling Sequence:

```
Call PLUG23 (IPL,NNO,X,Y,Z,TEL,PEL,ZSEL,IP,NORD,NERR,NOINK,
             AK,AN,FT,SEL,SIGEL,AMASS,DAMPV,DAMPS,NRSEL,INO,
             NODES,NMASS,NDMPV,NDMPS,NSEL,KK,KF,KS,KTS,KM,
             KDS,KDV,KN,IUSEL,EPSIO,SO,MAT,EXTRA,GPAXEL,
             NDIR,NDEG,ICONT)
```

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

X(1), Y(1), Z(1), PEL(12,2), TEL(12,2), AK(1), FT(1), SEL(1), SIGEL(1), AMASS(1), NODES(1), EPSIO(1), SO(1), MAT(1), EXTRA(1), TR(4,6), AKR(10), S(4)

12. Subroutine Used: ELPLUG

13. Subroutines Required: ELTEST, BCB, SYMPRT, MPRD, TXOUT

14. Remarks: None

1. Subroutine Name: PLUG 25

2. Purpose: To form the element matrices for a triangular cross-section ring element with applications toward the analysis of thick walled and solid axisymmetric structures of finite length. They may be used to idealize any axisymmetric structure by taking into account:

   (1) arbitrary axial variation in geometry
   (2) axial variation in orientation of material axes of orthotropy
   (3) Radial and axial variations in material properties
   (4) any asymmetric loading system including pressure and temperature
   (5) degradation of material properties due to temperature

   The complete finite element representation, consists of the algebraic expressions for the following matrices:

   (1) stiffness
   (2) stress
   (3) mass
   (4) thermal load
   (5) pressure load

3. Equations and Procedures: The development of the complete element representation arises from the Lagrange Equation which is:

$$\frac{\partial \Phi_1}{\partial q_R} + \frac{d}{dt}\left(\frac{\partial \Phi_2}{\partial \dot{q}_r}\right) = 0$$

   where q_R = γ^{th} generalized displacement coordinate

   Φ_1 = total potential energy

   Φ_2 = kinetic energy

   $\dot{q}\gamma$ = γ^{th} generalized velocity coordinate

   The element generalized displacements γ^{th} can be expressed in fourier series form.

The following procedure is included:

(a) call subroutine eltest to varify the input control values

(b) call subroutine HRAICS to form the coordinate transformation matrix and area matrix

(c) call subroutine HRAIES to form the material properties matrices

(d) call subroutine HRAIKS to compute the element stiffness matrix

(e) call subroutine HRAISS to form the element stresses matrix

(f) call subroutine HLOGEN to form the pressure load matrix

(g) call subroutine HTHGEN to form the thermal load matrix

4. Input Arguments

IPL - internal element identification number (25)
NNO - number of element defining points (4)
XC - coordinates of element defining points
YC - coordinates of element defining points
TEL - used by elplug
PEL - used by elplug
QSEL - used by elplug
FP - used by elplug
NORD - order of element stiffness matrix
NERR - error return
NOINK - number of element in lower half of stiffness matrix
AKEL - stiffness matrix
ANEL - used by elplug
FTEL - pressure load + thermal load matrix
STRESS - element stress matrix
SZALEL - element thermal stress matrix
AMASS - mass matrix
DAMPV - used by elplug
DAMPS - used by elplug
NRSEL - number element in stress matrix
INO - used by elplug
NODES - node point numbers
NMASS - number element in mass matrix
NDMPV - used by elplug
NDMPS - used by elplug
NSEL - number of elements in stress matrix
KK - element stiffness matrix control variable
KF - used by elplug
KS - element stress matrix control variable
KTS - used by elplug
KM - used by elplug
FN - used by elplug
KVM - used by elplug
KN - used by elplug
IUSEL - used by elplug
EPSIO - used by elplug

SO - used by elplug
MAT - input temperature interpolated material properties
EXTRA - used by elplug
GPAXEL - used by elplug
NDIR - number of directions of element defining points (3)
NDEG - number of solution degrees of freedom
ICONT - not used
AJ - harmonic number for pressure load
PR - harmonic coefficient for pressure load in radial direction
PZ - harmonic coefficient for pressure load in axial direction
JT - harmonic number for thermal load
PRT - harmonic coefficient for thermal load in radial direction
PZT - harmonic coefficient for thermal load in axial direction
IIRT - pressure and thermal load matrix generated control variable

5. Output Arguments:

NERR - error indicator
AKEL - element stiffness matrix
FTEL - element thermal and pressure load matrix
STRESS - element stress matrix
SZALEL - element thermal stress

6. Error Returns:

If not error, then NERR is set to zero

7. Calling Sequence:

(IPL,NNO,XC,YC,ZC,TEL,PEL,OSEL,IP,NORD,NERR,NOINK,AKEL,ANEL,
FTEL,STRESS,SZALEL,AMASS,DAMPS,NRSEL,INO,NODES,NMASS,NDMPV,
NSEL,KK,KF,KS,KTS,KM,FN,KVM,KN,IUSEL,EPSIO,SO,MAT,EXTRA,GPAXEL,
NDIR,NDEG,ICONT,AJ,PR,PZ,JT,PRT,PZT,IIRT)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Calling Subroutine: Elplug

12. Subroutines Called:

ELTEST,HRAICS,HRAIES,HRAIKS,HRAISS,HLOGEN,HTHGEN, and HMASSG

1. Subroutine Name: HRAICS

2. Purpose: To generate coordinate transformation matrix and area integrals table for asymmetric triangular cross section ring element.

3. Equations and Procedures: The coordinate matrix [GAMAB] is formed by algebraic assignment. The area integrals table, DELINT, is formed by algebraic methods using the function subroutine AI.

4. Input Arguments: R, Y: coordinates of node points
WIPR: print control

5. Output Arguments: GAMABQ: coordinate matrix
DELINT: area table of integrals

6. Error Returns: None

7. Calling Sequence:
(R,Y,Z,GABABQ,DELINT,DCURL,JSING,WIPR)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Subroutine User: Plug 25

12. Subroutine Required: AI, TESTJ

1. Subroutine Name: HRAIKS

2. Purpose: Compute stiffness matrix for asymmetric triangular cross section ring element.

3. Equations and Procedures: This subroutine uses harmonic number JA, elastic matrix E, and area integral table to form intermediate matrix ACURL. This matrix is then multiplied with the coordinate transformation matrix GAMABQ to form the final stiffness matrix AKEL.

4. Input Arguments:

   GAMABQ: coordinate transformation matrix
   E : material constant matrix
   DELINT: area integral table
   WIPR : print control variable
   JA : harmonic number

5. Output Arguments:

   AKEL : final stiffness matrix

6. Error Returns: None

7. Calling Sequence:

   (GAMABQ,E,DELINT,AKEL,WIPR,JA)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Subroutine User: Plug 25

12. Subroutine Required: TPRD,MPRD

1. Subroutine Name: HRAISS

2. Purpose: To generate the stress matrix for asymmetric triangular cross section ring element.

3. Equations and Procedures:

   For a given harmonic number and element grids point constant, an intermediate matrix WJ is formed. This matrix is then multiplied with the coordinate transformation matrix GAMABQ and material constants matrix to form the final matrix STRXP.

4. Input Arguments:

   X,Z: coordinates of node points
   GAMABQ: coordinate matrix
   E: material constants matrix
   WIPR: print control
   JA: harmonic number

5. Output Arguments:

   STRXP: final stress matrix

6. Error Returns: None

7. Calling Sequence:

   (X,Z,GAMABQ,E,STRXP,WIPR,GAM,JA,EM)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Subroutine User: Plug 25

12. Subroutine Required: MPRD

1. Subroutine Name: HRAIES

2. Purpose: To generate the transformed matrix of elastic constants for asymmetric triangular cross section ring element.

3. Equations and Procedures:

   a) Generate elastic constants

   b) Generates the elastic transformation matrix.

4. Input Arguments:

   ER,ETHETA,EZ: moduli of elasticity (Young's)
   VRO,VOZ,VZR: poissons ratio
   GRZ: modulus of rigidity
   GAM: angle between material axes and element axes

5. Output Arguments:

   E: transformed elastic constant matrix

6. Error Returns: None

7. Calling Sequence:

   (ER,ETHETA,EZ,VRO,VOZ,VZR,GRZ,GAM,TEO,EM,E,EL,WIPR,GRO,GOR)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Subroutine User: Plug 25

1. Subroutine Name: HLOGEN

2. Purpose: To generate pressure load for asymmetric triangular cross section ring element.

3. Equations and Procedures: Pressure load is function of harmonic number, harmonic coefficients, and node point number.

4. Input Arguments:

   X,Z: coordinates of node points
   GAMABQ: coordinate matrix
   JA: harmonic number
   PR: harmonic coefficient for radial direction
   PZ: harmonic coefficient for axial direction
   WIPR: print control

5. Output Arguments:

   FTXP: final pressure load matrix

6. Error Returns: None

7. Calling Sequence:

   (X,Z,GAMABQ,FTXP,WIPR,JA,PR,PZ)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Subroutine User: Plug 25

1. Subroutine Name: HMASSG

2. Purpose: To generate MASS matrix for asymmetric triangular cross section ring element.

3. Equations and Procedures:

   For a given harmonic number, density constant and element area vector, an intermediate matrix BMASS is formed. This matrix is then multiplied with the coordinate transformation matrix GAMABQ to form the final mass matrix AMASS.

4. Input Arguments:

   AMASS: mass matrix
   GAMABQ: coordinate matrix
   DENS: density constant
   DELINT: area vector
   JA: harmonic number
   WIPR: print control

5. Output Arguments:

   AMASS: final mass matrix

6. Error Returns: None

7. Calling Sequence:

   (AMASS,DENS,GAMABQ,DELINT,JA,WIPR)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Subroutine User: Plug 25

12. Subroutine Required: TPRD,MPRD

1. Subroutine Name: HTHGEN

2. Purpose: To generate thermal load and thermal stress matrix for asymmetric triangular cross section ring element.

3. Equations and Procedures:

   a. For a given harmonic number AJT, harmonic coefficient PRT, thermal material coefficients and area table integral, an intermediate matrix DTT is formed. This matrix is then multiplied with coordinate transformation matrix GAMABQ to form the thermal load matrix.

   b. Thermal stress is a function of the material expansion coefficients and the material constants.

4. Input Arguments:

   GAMABQ: coordinate matrix
   E : material constants matrix
   DELINT: area integrals table
   ALFRR,ALFZZ,ALFOO: material expansion coefficient
   AJT : harmonic number
   PRT : harmonic coefficient
   WIPR : print control variable

5. Output Arguments:

   FTJ : contains thermal load element
   SZALEL: contains element thermal stress matrix

6. Error Returns: None

7. Calling Sequence:

   (GAMABQ,E,DELINT,FTJ,ALFRR,ALFZZ,ALFOO,AJT,PRT,PZT,WIPR,SZALEL)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Subroutine User: Plug 25

12. Subroutine Required: TPRD

1. Subroutine Name: PLUG26

2. Purpose: To process the required input and control the generation of membrane and flexure stiffness, mass, applied load, thermal stress and stress matrices for the high aspect ratio quadrilateral element.

3. Equations and Procedures:

A. Initialize constants.

B. Process coordinate data and form transformation matrices for different coordinate systems used in element.

C. Form: Material properties matrix in EM
Strain transformation matrix in TES
Stress transformation matrix in TESS
Displacement transformation matrix in TW

D. Compute $[EG] = [TES]^T[EM][TES]$

E. If print option equals -1, call PIPRTA to print processed input data.

F. Initialize thermal load, pressure load, thermal stress, stress and mass matrices to zero.

Membrane computations are performed as follows:

G. Store x and y coordinates in geometric system into xx and yy arrays.

H. Call GENSM to calculate the values of the mode shape and derivative functions H, HZ, HV for the NR and NS sample points.

I. Call CK26 to compute the membrane stiffness matrix and transformation matrix TGRM.

J. If requested, call CMMASS to compute membrane mass matrix in geometric system, CMM.

Call BCB to form $[AMASS] = [TGRM]^T[Cmm][TGRM]$ in system coordinates.

K. If requested, call STRS26 to compute membrane stress matrix.

I. If requested, call FT26 to compute membrane thermal load column.

M. If requested, call SZAL26 to compute membrane thermal stress column.

N. Change temperature multiplier on thermal stress.

Flexural computations are then performed in the following manner:

O. Call CK22 to add the flexural contributions to the stiffness matrix [K21S].

P. Apply transformation to global system by performing [TFM] = [TGAMB][TOGB][TGRB].

Q. If stress and/or force matrices are requested then
   (a) If input pressure not equal to 0, call CFP to formulate the pressure matrix.
   (b) The flexural contributions to the stress matrix are formulated by calling CSTF.
   (c) If flexural input temperature not equal to zero, calls CFFTS to formulate the thermal force and stress matrices.

R. If mass is requested, then
   (a) Call CFMASS to form the membrane mass matrix in local system [CMF].
   (b) The mass matrix is transformed to global system as $[AMASS] = [TGFS]^T[CMF][TGFS]$.

S. Again, if the print option is -1, intermediate element computation printout is obtained from PRT1.

4. Input Arguments:

IPL - Plug number
NNO - Number of nodes (8)
XC,YC,ZC- Coordinates of element node points
TEL - Temperature array of element node points
PEL - Pressures at element node points
NN - Number of nodes
NL - Node point numbers
KK,KN - Control for computation of matrices (see remarks)
GPAXEL - Grid point axes transformations
MAT - Array containing material properties
EXTRA - Array containing geometric properties

5. Output Arguments:

K21S - Stiffness matrix
FTEL - Element force matrix
S - Stress matrix
SZALEL - Thermal stress matrix
AMASS - Mass matrix for dynamic analysis

6. Error Returns:

a. Standard error returns by ELPLUG (NERR)
b. Sinα = 0 indicates coordinate input data error

7. Calling Sequence:

Call PLUG26(IPL, NNO, XC, YC, ZC, TEL, PEL, QS, IP, NORD, NERR, NOINK, K21S, AN1, FTEL, S, SZALEL, AMASS, DAMPV, DAMPS, NRSEL, NN, NL, NMASS, NDMPV, NDMPS, NSEL, KK, KF, K8, KTS, KM, KDS, KDV, KN, IUSEL, EPSLON, SIGZER, MAT, EXTRA, GPAXEL, NDIR, NDEG, ICONT)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total storage required is $10864_{10}/2A70_{16}$ bytes.

12. Subroutine User: ELPLUG

13. Subroutines Required:

ELTEST	PIPRTA	STRS26
NEWFT	GENSM	FT26
CDELPQ	PRT1	SZAL26
MINV	CK22	
BCB	CFP	
CK26	CSTF	
MABC	CFFTS	
CMMASS	CFMASS	

14. Remarks:

The following is a list of control indicators for PLUG26. For all indicators shown a value of one will cause the operation to be performed and a value of zero will cause the operation to be skipped.

LT1	- compute membrane contributions
LT2	- compute flexural contributions
KK	- compute element stiffness matrix
KF	- compute element force matrix (thermal and/or pressure)
K8	- compute element stress matrix
KTS	- compute element thermal stress matrix
KM	- compute element mass matrix
KDS	- not used
KDV	- not used
KN	- compute element incremental stiffness matrix.

1. Subroutine Name: GENSM

2. Purpose: Evaluate the membrane displacement function, transformation matrix and the two derivative displacement function transformation for all sample points in the high aspect ratio quadrilateral thin shell element. Also evaluate weights used for Gaussian product formula.

3. Equations and Procedures:

Loop on the number of sample points in the product quadrature formula and calculate the values of H, HV and HZ for each of the eight assumed displacement degrees of freedom. If requested, print values of H, HV, and HZ and W.

H represents displacement transformation function.

HZ represents derivative displacement $\frac{\partial}{\partial \eta}[H]$

HU represents derivative displacement $\frac{\partial}{\partial \mu}[H]$

4. Input Arguments:

NR	- Number of sample points in 'x-like' y direction
NS	- Number of sample points in 'y-like' u direction
IPRINT	- Print control

5. Output Arguments:

H	- Displacement function transformation matrix for NR*NS sample points
HZ	- Derivative displacement function transformation matrix for NR*NS sample points
HV	- Derivative displacement function transformation matrix for NR*NS sample points
W	- Weights for product Gauss NR*NS quad formula.

6. Error Returns: None

7. Calling Sequence:

Call GENSM (NR,NS,H,HZ,HV,W,IPRINT)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

2288_{10} bytes

$8F8_{16}$ bytes

12. Subroutine User: PLUG26

13. Subroutine Required: None

14. Remarks:

The displacement function and the derivative displacement functions as defined as:

$$HZ = \begin{Bmatrix} (1-u)[4z-2u-1] \\ u\,[4z+2u-3] \\ u[4z-2u-1] \\ (1-u)[4z+2u-3] \\ 4u(1-u) \\ u(-8z+4) \\ -4u(1-u) \\ (1-u)(-8z+4) \end{Bmatrix}, \quad HU = \begin{Bmatrix} z[4u-2z-1] \\ z[4u+2z-3] \\ (1-z)(4u-2z-1) \\ (1-z))4u+2y-3) \\ 4z(-2u+1) \\ 4z(1-z) \\ 4(1-z)(-2u+1) \\ -4z(1-z) \end{Bmatrix}$$

$$H = \begin{Bmatrix} z(1-u)(2z-2u-1) \\ zu(2z+2u-3) \\ (1-z)u(-2z+2u-1) \\ (1-z)(1-u)(-2z-2u+1) \\ 4zu(1-u) \\ 4zu(1-z) \\ 4(1-z)u(1-u) \\ 4z(1-z)(1-u) \end{Bmatrix}$$

1. Subroutine Name: CK26

2. Purpose: To generate the membrane stiffness matrix for the high aspect ratio quadrilateral finite element.

3. Equations and Procedures:

   1. Initialize constants and arrays.
   2. Call CTGRM to evaluate TGRM matrix which maps geometric into system reference system.
   3. Loop on the number of points in the product quadrature formulas the following calculations:
      A. Call TUDJAC to evaluate the TU and D matrices and the Jacobian for the sample point at hand.
      B. Call BCB to form the product
         $$[EO] = t|J|[TU]^T[EG][TU] \quad .$$
      C. Call BCB to form the product
         $$[K11G] = [K11G] + W_{ij}[D]^T[EO][D] \quad .$$
      D. If full print was requested, print TU, D, and K11G matrices.
   4. Call BCB to form final membrane stiffness matrix
      $$[K21S] = [TGRM]^T[K11G][TGRM] \quad .$$

4. Input Arguments:

NDIR	- Number of degrees of freedom/grid point.
NDEG	- Number of types of degrees of freedom
ICONT	- Indicator for grid point axis transformation
GPAXEL	- Array which contains grid point axis transformation
NNO	- Number of grid points
NL	- Array used to store element grid points
EEZ	- Eccentricity coefficient
AJ	- Array containing 'x-like' coefficients of 8 grid points
BJ	- Array containing 'y-like' coefficients of 8 grid points
TPRIME	- Transformation from geometric to reference degrees of freedom
T	- Membrane thickness
LT1	- Indicator of membrane thickness
EG	- Material properties matrix in geometric system
K11G	- Array used to accumulate stiffness matrix
TU	- Displacement function transformation
EO	- Matrix used for intermediate products
IFS	- Matrix used for grid point axis transformation
HZ	- Matrix which contains displacement function derivatives
HU	- Matrix which contains displacement function derivatives
D	- Matrix which contains displacement function derivatives

NR - Number of quadrature points in 'x-like' direction
NS - Number of quadrature points in 'y-like' direction
W - Array containing weights for product quad formulas
IPRINT - Print control

5. Output Arguments:

K21S - Membrane stiffness matrix in system coordinates
TGRM - Transformation from geometric to system coordinates

6. Error Returns: None

7. Calling Sequence:

```
Call CK26 (K21S,NDIR,NDEG,ICONT,GPAXEL,NNO,NL,EEZ,AJ,BJ,
           TPRIME,T,LT1,EG,TGRM,K11G,TO,EO,TFS,HZ,HU,D,
           NR,NS,W,IPRINT)
```

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: 2458_{10} bytes

12. Subroutine User: PLUG26

13. Subroutines Required:

CTGRM, AXTRA2, TUDJAC, BCB

14. Remarks: None

1. Subroutine Name: TUDJAC

2. Purpose: To calculate displacement function derivative matrix, TU matrix, and Jacobian of TU matrix for one sample point inside the quadrilateral.

3. Equations and Procedures:

$$XZ = \sum_{i=1}^{8} HZ(i)*AJ(i)$$

$$YU = \sum_{i=1}^{8} HU(i)*BJ(i)$$

$$XU = \sum_{i=1}^{8} HU(i)*AJ(i)$$

$$YZ = \sum_{i=1}^{8} HZ(i)*BJ(i)$$

RJAC = XZ*YU - XU*YZ (Jacobian)

D matrix is formed from HZ and HU arrays.

$$[TU] = \frac{1.0}{RJAC} \begin{bmatrix} YU & -YZ & 0 & 0 \\ & & -XU & XZ \\ -XU & XZ & YU & -YZ \end{bmatrix}$$

If print requested, print XZ, YU, XU, YZ, RJAC.

4. Input Arguments:

AJ	- 'x-like' coordinates of 8 grid points in element
BJ	- 'y-like' coordinates of 8 grid points in element
IPRINT	- Print control if IPRINT=1, then print
HZ	- Derivative matrix for one sample point
HU	- Derivative matrix for one sample point

5. Output Arguments:

TU	- Displacement function transformation
D	- Displacement derivative function matrix
RJAC	- Jacobian of TU matrix

6. Error Returns: None

7. Calling Sequence:

```
Call TUDJAC (AJ,BJ,HZ,HU,TU,D,RJAC,IPRINT)
```

8. Input Tapes: None

9. Ouptut Tapes: None

10. Scratch Tapes: None

11. Storage Required: 972_{10} bytes

12. Subroutine User: CK26, FT26

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: FT26

2. Purpose: To generate membrane thermal load vector for high aspect ratio quadrilateral element.

3. Equations and Procedures:

   A. Compute temperature of element for all eight grid points.

   B. If all temperatures are 0.0, set thermal load vector to 0.0 and return.

   C. Compute [AM1] = [EM][ALPHM] by calling MSB.

      Compute $[AM2] = [TES]^T[AM1]$ by calling MATB.

   D. If print is requested, print all matrices calculated so far.

   E. Loop on the number of points in the product quadrature formula:

      1. Evaluate [TU],[D] and Jacobian of [TU] for sample point.

      2. Multiply [D] matrix by temperatures at each grid point.

      3. Calculate $[IT] = [TU]^T[AM2]$ by calling MATBS.

      4. Accumulate $[FPB] = [FPB] + [D]^T[IT]$ by calling MATBS.

   F. Evaluate $[FT] = [TGRM]^T[FPB]$ by calling MATB.

4. Input Arguments:

EM	- Material properties matrix
ALPHM	- Coefficients of thermal expansion
TES	- Strain transformation matrix
TU	- Displacement function transformation
T	- Membrane plate thickness
FPB	- Array used to accumulate intermediate thermal load
TGRM	- Transformation from geometric to reference coordinates
NR	- Number of sample points in 'x-like' direction
NS	- Number of sample points in 'y-like' direction
HZ	- Displacement derivative function
HU	- Displacement derivative function
D	- Displacement derivative function
XX	- x-coordinates in geometric system
YY	- y-coordinates in geometric system
W	- Weights for product Gauss quadrature formula
TEL	- Temperature of 4 corner element grid points
H	- Displacement function transformation
TZ	- Initial temperature of structure
IPRINT	- Print control

5. Output Arguments:

FT	- Element thermal load in reference coordinates
TELE	- Effective temperature for eight grid points

6. Error Returns: None

7. Calling Sequence:

```
Call FT26 (EM,ALPHM,TES,TU,T,FT,FPB,TGRM,NR,NS,HZ,HU,D,
          XX,YY,W,TEL,H,TELE,TZ,IPRINT)
```

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

2846_{10} bytes

$B28_{16}$ bytes

12. Subroutine User: PLUG26

13. Subroutines Required:

MSB, MATB, TUDJAC, MATBS

14. Remarks: None

1. Subroutine Name: SZAL26

2. Purpose: To generate thermal stress matrix for high aspect ratio quadrilateral element.

3. Equations and Procedures:

   A. Form [AM1] = [EM][ALPHM] by calling MSB.

   B. Form [AM2] = DELTM*T $[TES]^{T_0}$[AM1] by calling MATBS.

   C. Form [SZLM] = [TESS][AM2] by calling MAB.

   D. Define elements of [SZLM] as the first three elements in each row of [SZALEL].

   E. If requested, print [TESS], [AM1], [AM2], [SZLM].

4. Input Arguments:

E	- Material properties matrix
ALPHM	- Coefficients of thermal expansion
T	- Membrane thickness
DELTM	- Membrane temperature
TESS	- Stress transformation
TES	- Strain transformation
IPRINT	- Print control

5. Output Arguments:

   SZALEL - Material thermal stress matrix

6. Error Returns: None

7. Calling Sequence:

   Call SZAL26 (EM,ALPHM,T,DELTM,TESS,SZALEL,TES,IPRINT)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: 1244_{10} bytes; $4E8_{16}$ bytes

12. Subroutine User: PLUG26

13. Subroutines Required:

    MSB, MAB, MATBS

14. Remarks: None

1. Subroutine Name: STRS26

2. Purpose: To evaluate membrane stress matrix for 5 stress point. (4 corner points plus centroid.)

3. Equations and Procedures:

   A. Form [AM1] = [TESS] [EG] by calling MBS.

   B. For each stress point

      1. Calculate YU, YZM, XUM, XZ and VJAC =1.0/(XZ+YU-XUM+YZM)

      2. Form elements in D matrix [D] = [TU] [D] based on equations in remarks section.

   C. Form [AM2] = [AM1] [D] by calling MABS.

   D. Form [AM5] = [TGRM] [AM2] by calling MAB.

   E. Store elements of AM5 as the first 3 elements in each row of S matrix.

   F. If requested, print matrices.

4. Input Arguments:

EG	- Material properties matrix (geometric system)
TESS	- Stress transformation matrix
T	- Membrane thickness
TGRM	- Transformation matrix from geometric to reference
D	- Array used in intermediate calculations
AM1	- Array used in intermediate calculations
AM2	- Array used in intermediate calculations
AM5	- Array used in intermediate calculations
AJ	- 'x-like' coordinates of eight element grid points
BJ	- 'y-like' coordinates of eight element grid points
IPRINT	- print control

5. Output Arguments:

S	- Element stress matrix

6. Error Returns: None

7. Calling Sequence:

```
Call STRS26 (EG,TESS,T,S,TGRM,D,AM1,AM2,AM5,AJ,BJ,IPRINT)
```

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: 4080_{10} bytes ; FFB_{16} bytes

12. Subroutine User: PLUG26

13. Subroutines Required: MBS, MABS, MAB

14. Remarks:

The calculation [TU] [D] is required in the expression for the stress matrix. Since the two matrices are evaluated at the 5 stress points, many of the terms for HZ and HU in the D matrix drop out. Thus we can write the expression for [TU] [D] directly.

DISPLACEMENT D.O.F. HZ	STRESS POINTS				
	(1) (1,0)	(2) (1,1)	(3) (0,1)	(4) (0,0)	(5) (⅓,⅓)
1	3	0	0	-1	0
2	0	3	-1	0	0
3	0	1	-3	0	0
4	1	0	0	-3	0
5	0	0	0	0	1
6	0	-4	4	0	0
7	0	0	0	0	-1
8	-4	0	0	4	0

DISPLACEMENT D.O.F. HU	STRESS POINTS				
	(1) (1,0)	(2) (1,1)	(3) (0,1)	(4) (0,0)	(5) (⅓,⅓)
1	-3	1	0	0	0
2	-1	3	0	0	0
3	0	0	3	-1	0
4	0	0	1	-3	0
5	4	-4	0	0	0
6	0	0	0	0	1
7	0	0	-4	4	0
8	0	0	0	0	-1

1. Subroutine Name: MBS

2. Purpose: To evaluate the matrix product B*S(SYM).

3. Equations and Procedures:

   B is of order (M,N) and dimensioned (M1, N1)
   S is (NxN) symmetric stored in lower half by rows
   AN (M,N) is the product B(M,N) * S(N,N)
   AN must have row dimension M.

4. Input Arguments:

B	- Rectangular matrix of order (M,N)
S	- Symmetric matrix of order (N,N)
M	- Number of rows in B and AN matrices
N	- Number of rows in S matrix
M1	- Row dimension of B matrix
N1	- Column dimension of B matrix

5. Output Arguments:

   AN - Matrix product of order (M,N)

6. Error Returns: None

7. Calling Sequence:

   Call MBS (B,S,AN,M,N,M1,N1)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: 770_{10} bytes ; 308_{16} bytes

12. Subroutine User: STRS26

13. Subroutines Required: None

14. Remarks: None

1. Subrcutine Name: MABS

2. Purpose: To evaluate the matrix product A * B = AN.

3. Equations and Procedures:

$$AN_{nm} = \sum_{j} A_{nj} * B_{jm} + IASSY * AN$$

4. Input Arguments:

A	- Elements of [A] matrix
B	- Elements of [B] matrix
N	- Number of rows in [A] matrix
L	- Number of columns/rows in [A] [B] matrix
M	- Number of columns in [B] matrix
N1,M1	- Dimension of [A] matrix
N2,M2	- Dimension of [B] matrix
IASSY	- Assembly control

5. Output Arguments:

AN - The matrix product

6. Error Returns: None

7. Calling Sequence:

Call MABS (A,B,AN,N,L,M,N1,M1,N2,M2,IASSY)

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

Total storage required is 798_{10} ; $31E_{16}$ bytes.

12. Subroutine User: Used by many subroutines within PLUG26.

13. Subroutines Required: None

14. Remarks:

If IASSY = 1 AN is product plus previous AN values
= 0 AN is product only.

1. Subroutine Name: MATBS

2. Purpose: Subroutine to evaluate the matrix prcduct of A transpose and B.

3. Equations and Procedures:

$$AN_{nm} = scal \sum_{e} A^{T}_{en} * B_{em} + IASSY * AN$$

where A^{T}_{en} is the transpose of A_{ne}.

4. Input Arguments:

A	- Elements of [A] matrix
B	- Elements of [B] matrix
N	- Number of rows in $[A]^T$ matrix (order)
L	- Number of columns in $[A]^T$ matrix (order)
M	- Number of columns in [B] matrix (order)
N1,M1	- Dimension of [A] matrix
N2,M2	- Dimension of [B] matrix
SCAL	- Scalar which multiplies product
IASSY	- Assembly control (see Remarks)

5. Output Arguments:

AN - Elements of matrix product

6. Error Returns: None

7. Calling Sequence:

```
Call MATBS (A,B,AN,N,L,M,N1,M1,N2,M2,SCAL,IASSY)
```

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required:

Total storage required is 928_{10} bytes; $3A0_{16}$ bytes

12. Subroutine User: Various subroutines in PLUG26

13. Subroutines Required: None

14. Remarks:

If IASSY = 1, AN = previous AN values plus new pdt.
= 0, AN - product only

APPENDIX IX

DIRECT MACHINE CONTROL IMPLEMENTATION DOCUMENT

This documentation is primarily intended for the programmer analyst or systems analyst responsible for the initial implementing and subsequent maintenance of the system.

There are five sections in this document. Special program considerations are presented in Section I (Reference 2). Included in this section is a description of internal data storage, external or peripheral data storage, and programming specifications followed. Section II deals with the operational considerations of the program. Included in this section is a discussion of the procedure to be followed in an initial implementation of the program at an installation. Data set assignments and storage limitations are discussed, and some special control cards are described (Reference 2). Section III describes how new agendum level abstraction instructions may be added to the MAGIC system. Section IV contains a catalogued procedure used for initial implementation of the MAGIC System. Section V contains a preprinted form to be used in reporting any problems with the implementation or running of the MAGIC program.

TABLE OF CONTENTS

Section		Page No.
I	SPECIAL PROGRAMMING CONSIDERATIONS	9.4
	A. Internal Data Storage	9.4
	1. Common Storage	9.4
	2. Storage of Alphameric Information. .	9.4
	3. Matrix Names	9.4
	4. Data Set Names	9.5
	5. Compression Scheme	9.5
	B. Data Set Formats	9.6
	1. Data Set Header Record	9.6
	2. Data Set Trailer Record.	9.6
	3. Matrix Header Record	9.6
	4. Matrix Trailer Record	9.7
	5. Matrix Column Record	9.7
	C. Programming Standards	9.7
II	OPERATIONAL CONSIDERATIONS.	9.9
	A. MAGIC III Data Set Philosophy	9.9
	B. Storage Limitations.	9.12
	C. Initial Implementation	9.13
	D. Machine Resource Data Card	9.16

Section		Page No.
III	AGENDUM LEVEL ABSTRACTION INSTRUCTIONS . . .	9.19
	A. Introduction	9.19
	B. Modifications to Subroutine AGENDM . . .	9.19
	C. Setting up the Agendum Library	9.20
	1. Agendum Control Cards	9.20
	2. Example of an Agendum Library . . .	9.20
	3. Example of Agendum Usage	9.21
IV	MAGIC CATALOGUED PROCEDURE	9.22
V.	MAGIC III TEST RUN REPORT FORM	9.24
	MAGIC III TEST RUN REPORT FORM (Trouble Supplement Sheet)	9.26

SECTION I

SPECIAL PROGRAMMING CONSIDERATIONS

A. INTERNAL DATA STORAGE

1. Common Storage

There are only four variables which remain in blank common at all times. These four variables are initialized when the machine resources module is compiled. The four variables are stored in the first four words of blank common and are defined as follows: (1) the first word contains an integer which is the logical number of the system input data set, (2) the second word contains an integer which is the logical number of the system output data set, (3) the third word contains an integer which is the order of the largest matrix permitted in the system, (4) the fourth word contains the number of words remaining in blank common (i.e., from word five to the end). The remaining portion of blank common, whose length is specified in word number four, is used by all FORMAT II routines as working storage.

2. Storage of Alphameric Information

All alphameric information (e.g., matrix names) is stored one character per word. Consistency is retained by reading all alphameric data into storage with an "A1" format and by compiling all alphameric data into storage using DATA statements with an "1H" format.

3. Matrix Names

The names of all matrices processed by the program are one (1) to six (6) characters in length. The first character of a matrix name must be alphabetic. The matrix names are stored one character per word. A seventh word is addended to the six words which contain the characters constituting the matrix name. This last word contains a plus (+) or minus (-) integer one (1). The sign of the seventh word indicates the sign of the matrix, (i.e., plus or minus). (Note: The characters in words two (2) through six (6) may be blank.)

4. Data Set Names

The names of master input and master output data sets follow the same rules as matrix names, with one exception. The seventh word of the data set name does not necessarily contain an integer one (1). Instead it contains an integer which is specified by the user of the program when the particular data set was created. If the data set is neither a master input data set nor a master output data set, the data set name consists of six (6) blank characters followed by an integer zero (0).

5. Compression Scheme

The columns of all matrices are stored in one of two formats, full or compressed.

If the number of zero elements in the column is greater than fifty percent, the column is stored in compressed format. When a column is compressed, it is stored as follows:

V
I
V
I
.
.
.
V
I

The V's are the non-zero values in the column and are floating point numbers. The I's are the row numbers of each of the V's and are stored as integers. The row number of any given value is denoted by the integer immediately following the value in storage.

If the number of non-zero elements is not greater than fifty percent, the column is stored in full format. When a column is full, it is stored as follows:

V
V
V
.
.
.
V

The V's are the zero and non-zero elements of the column and are floating point numbers.

B. DATA SET FORMATS

1. Data Set Header Record

The first logical record on all data sets which are processed by the program is called a data set header. The data set header is ten words long. The first word contains an integer number which is minus ten (-10). This word indicates that the record is a data set header. The second word contains an integer zero (0). This word has no significance in a data set header. The third word contains an integer seven (7). This word indicates the number of words remaining in the logical record. The remaining seven words contain an alphameric data set name if the data set is either a master input or master output data set, or contain seven (7) zeros (0) if the data set is not a master input or master output data set.

2. Data Set Trailer Record

The last logical record on all data sets which are processed by the program is called a data set trailer. The data set trailer is four (4) words long. The first word contains an integer which is minus twenty (-20). This word indicates that the record is a data set trailer. The second word contains an integer zero (0). This word has no significance in a data set trailer. The third word contains an integer one (1). This word indicates the number of words remaining in the logical record. The fourth word contains an integer zero (0). This word has no significance in a data set trailer.

3. Matrix Header Record

The first logical record in all matrices which reside on data sets which are processed by the program is called a matrix header. The matrix header is twelve (12) words long. The first word contains an integer which is minus one (-1). This word indicates that the record is a matrix header. The second word contains an integer zero (0). This word has no significance in a matrix header. The third word contains an integer nine (9). This word indicates the number of words remaining in the logical record. The next seven (7) words contain the characters which comprise the matrix name. The last two words contain integer numbers which are the number of rows and the number of columns, respectively, in the matrix.

4. Matrix Trailer Record

The last record in all matrices which reside on data sets which are processed by the program is called a matrix trailer. The matrix trailer is four (4) words long. The first word contains an integer minus two (-2). This word indicates that the record is a matrix trailer. The second word contains an integer zero (0). This word has no significance in a matrix trailer. The third word contains an integer one (1). This word indicates the number of words remaining in the logical record. The fourth word contains an integer zero (0). This word has no significance in a matrix trailer.

5. Matrix Column Records

The logical records between the matrix header record and the matrix trailer record contain the columns of the matrix, one column per logical record. The column records are variable in length. The length depends on the number of rows in the matrix and the number of non-zero elements in the column. The first word of column record contains an integer which is the column number. The second word contains an integer which is either zero (0) or one (1). A zero (0) indicates that the column is full. A one (1) indicates that the column is compressed. The third word contains an integer which indicates the number of words remaining in the logical record. Words four (4) to the end contain the matrix column elements either in full or compressed form. If a column does not contain any non-zero elements, a corresponding column is omitted from the data set.

C. PROGRAMMING STANDARDS

In the design and coding of the MAGIC II system every effort was made to keep the system machine independent. With this consideration in mind, the following rules were developed and obeyed.

a. The MAGIC II system is written entirely in FORTRAN IV.

b. No advantage was taken of the peculiarities in the FORTRAN IV language.

c. All variables are implicitly typed with the exception of logical variables which cannot be implicitly typed.

d. "EQUIVALENCE" statements were used only when they were absolutely necessary.

e. No on-line communication with the computer operator is performed.

f. Blank common is used as working storage by all routines. The size of blank common is compiled into one control section (MRES) and is made available to all routines by being stored in the fourth word of blank common. Thus by recompiling only one control section (MRES), the entire system is able to take advantage of additional core storage which may have been made available

g. All references to FORTRAN logical data sets are variable. The numbers of all the FORTRAN logical data sets available to the MAGIC II system are compiled into one control section (MRES) and subsequently made available to all routines. These data set numbers may be changed by recompilation of one control section (MRES) or via the input data.

h. All alphameric information is stored one character per word, thus no adcantage of word size was taken.

i. No advantage of the bit configuration of any character was taken.

j. The MAGIC II system is extremely modular, making additions and modifications as simple as possible.

k. In general, very straight forward and conservative coding practices were followed.

SECTION II

OPERATIONAL CONSIDERATIONS

A. MAGIC II DATA SET PHILOSOPHY

The MAGIC II system is designed to make extensive use of the input/output configuration of a given installation. Since the philosophy of MAGIC has been to keep the system completely machine independent all references to input/output devices is in terms of logical data sets. The MAGIC II data set philosophy is described in the following pages. The main topics covered are the manner in which an installation's standard FORTRAN data set configuration is communicated to the MAGIC II system, the minimum data set requirements of the MAGIC II system, and the method of selection of data sets for use as MAGIC II system utilities.

A logical data set in the MAGIC II system can take on one to four MAGIC system functions. The data may be a master input data set; that is, one which contains matrices which are required in the execution of the user's problem. The data set may be a master output data set; that is, one which is to contain matrices generated by the user's problem and which is to be saved at the end of the problem execution. The data set may be an input/output utility data set; that is, one which may be used by the MAGIC II system during both preprocessing and execution as intermediate storage. Finally the data set may be the instruction data set. The only function of this data set is to contain the executable FORTRAN instructions as generated during the preprocessing phase of a given run. This instruction data set is subsequently read during the execution phase of the same run. In addition to the four previously mentioned MAGIC system functions, there are two logical data set numbers which correspond to the system input unit and system output unit of the operating system monitoring execution of the MAGIC II system. These two data sets are used by the MAGIC II system exclusively for reading cards on or off-line and for printing on or off-line. Note that this allows batch processing.

There are two ways in which an installation's standard FORTRAN logical data sets are made available to the MAGIC II system. One of the two ways is by recompiling subroutine MRES. This subroutine must have compiled into it the logical

data set number corresponding to the system input unit and the logical data set number corresponding to the system output unit. The subroutine may have compiled into it information about some or all of the installation's standard FORTRAN logical data sets. This information consists of the five following quantities for each data set: (1) the logical data set number; (2) the MAGIC II system function of the data set; (3) the device with which the data set is associated, (e.g., tape); (4) the logical channel to which the device is attached, (e.g., A); (5) the capacity of the data set in basic machine units, (e.g., 5000 words). (A capacity of zero (0) indicates infinite capacity.) The second way the installation's four standard FORTRAN logical data sets may be made available to the MAGIC II system is by the use of the "NEW" option on the $MAGIC card and a SETUP card for each of the standard data sets. The "NEW" option has the effect of zeroing out all the data set information which has been compiled into the subroutine, with the exception of the logical data set number of the system input unit and the system output unit. Each SETUP card has the effect of re-establishing the five quantities which are associated with each data set.

Once the installation's standard configuration has been compiled into the subroutine MRES, temporary modifications may be made by the use of the "CHANGE" option on the $MAGIC card and a DELETE card or UPDATE card, depending on the particular modification to be made. The "CHANGE" option indicates that the DELETE or UPDATE cards follow the $MAGIC card. The DELETE card has the effect of zeroing out all information associated with the specified data set. The UPDATE card has the effect of changing any or all the information associated with an existing data set, that is a data set whose associated information has been compiled into subroutine MRES: or has the effect of making an additional data sets available to the MAGIC II system if the data set did not exist, that is, if the data set information associated with the specified data set has not been compiled into subroutine MRES.

There are several errors which will result from improper specification of logical data set information in the MAGIC II system. The most serious of the errors are those which the MAGIC II system, by virtue of its machine independence, cannot detect and from which it cannot recover. These errors are: (1) having specified an invalid logical data set number for either the system input unit or the system output unit in the compiled soubroutine MRES; (2) having specified the number of a logical data set as a MAGIC II system function

which is not in binary mode or which is not defined as one of the installation's standard FORTRAN logical data sets. These incorrect specifications may result from the use of a "SETUP" or "UPDATE" card, or improper use of subroutine MRES during compilation. There are other errors which result from invalid specification of logical data set information, which are internal to the MAGIC II system and hence, can be detected by the MAGIC II system. These errors are: (1) specifying a logical data set as a MAGIC II system function, when the data set is the system input or output unit; (2) specifying duplicate data set numbers on two different SETUP cards; (3) specifying a device type other than tape or disk for a data set whose function is master input or master output; (4) not specifying a sufficient number of data sets as being available to the MAGIC II system. Item number 4 in the list of errors will be clarified in the following pages.

The MAGIC II system requires that a certain number of logical data sets be available for use as system utilities during the preprocessing phase and execution phase. The function and method of selection of all required utilities is described below. The first data set selected by the preprocessor is the data set which is to contain the executable Format instructions. Since this data set is a MAGIC II system function, it may have been established at the time all the system functions were established. However, if no data set available to MAGIC has been given this system function, the preprocessor selects for this function one of the data sets that has the MAGIC II system function input/output utility. The next data set selected is used as a communication medium between the preprocessor modules. The preprocessor selects a data set which system function is input/output utility. This data set is set outside to contain any matrices which may be card input. During the allocation phase a fourth data set is selected for temporary use by the allocator. This data set is also selected from the MAGIC II input/output utilities. If no matrices have been card input it is possible that the data set selected by the allocator coincides with the data set which was to contain the card input matrices. In general, the preprocessor can function with a minimum of three (3) data sets if each one has the MAGIC II system function of input/output utility. If card input matrices exist in the user's problem, four (4) MAGIC II input/output utilities must exist.

The minimum data set requirements for the execution phase are determined during allocation. The user's problem is analyzed and the required number of master input and master output data sets is determined. Specific data sets are selected

from those whose MAGIC II system function is either master input or master output. If any data sets, whose system function is master input or master output, have not been selected for the user's problem, its system function is changed from master input or master output to input/output utility.

Associated with each valid abstraction instruction is an arithmetic module which is under control of the execution monitor. Each of these arithmetic modules requires a certain number of scratch data sets in addition to the data sets containing the matrices which are input to the instruction or created by the instruction. As a result, the allocator scans the user's abstraction instructions and determines the number of scratch data sets required by each associated arithmetic module. The maximum of these numbers is then ascertained and this number of data sets is set aside for future use as scratch data sets. These data sets are selected from the data sets whose MAGIC II system function is input/output utility. At this point the matrices occurring in the user's problem are allocated to the remaining data sets whose system function is input/output utility.

There are many errors which will occur when the minimum requirements of the MAGIC II system are not met. All of these errors are detectable by the MAGIC II system. When one occurs the appropriate error message is written. The most common error which occurs is not having enough input/output utilities available to the preprocessor at the time of the selection of an intermediate data set. When this condition arises, the error message which is written on the system output unit indicates the intermediate data set which the preprocessor has been unable to select. The condition can be corrected by specifying more data sets with the MAGIC II system function of input/output utility. An error condition also results when the user's problem requires more master input data sets than exist with the MAGIC II system function of master input. The error occurs also for an insufficient number of master output data sets.

B. STORAGE LIMITATIONS

The MAGIC II system is very flexible in its utilization of working storage, in that all subroutines use blank common as working storage and in that the allocation of blank common storage is all dynamic. The length of blank common is initialized in the COMMON statement in subroutine MRES. The size of blank common is bounded above only by the amount of

core storage available at a given installation. For large matrix problems it is desirable to have blank common as large as possible, since the MAGIC II system utilizes all of the available blank common storage. The lower bound of the size of blank common is determined by one of two things: (1) the size of a particular MAGIC II case, or (2) 708 words, whichever is greater. The size of a MAGIC II case is a function of such quantities as, the number of abstraction instructions, the size of the matrices in the system, the number of matrices in the system, etc. Since the allocation of blank common storage is all dynamic and is a function of the size of the user's problem, it is very difficult to calculate the exact number of words of blank common required. However a few guide lines will be given. In subroutine MRES a variable named KONST is initialized. This variable is the order of the largest matrix which the MAGIC II system will process. The size of blank common working storage (i.e., the WORK array) must be at least four and one half times the value of the variable KONST. Since the MAGIC II system is designed to handle matrices of order up to 3000, the usual value of KONST is 3000 and the mininum size of blank common working storage is 13500 words.

C. INITIAL IMPLEMENTATION

The following is a discussion of the procedure a system analyst should go through in initially implementing the MAGIC II system at his installation. There are several parameters which define the basic machine configuration which must be set. All these parameters are contained in subroutine MRES and are defined as follows:

a) NPIT is the FORTRAN logical data set number of the system input data set.

b) NPOT is the FORTRAN logical data set number of the system output data set.

c) NAGEND is the FORTRAN logical data set number of the data set which contains the agendum level abstraction instructions. If your installation does not make use of the AGENDUM capabilities set this variable to zero.

d) KONST is the order of the largest matrix which the MAGIC II system will process.

e) NWORK is the length of the table WORK. This length is the size of blank common less four. NWORK must be at least four and one half times KONST.

f) KONFIG is a table which describes all the FORTRAN logical data sets which are available for use by the the FORMAT II system. In the KONFIG table there are five entries for each of the available data sets. The first entry is the FORTRAN logical data set number of an available data set. Each of these data sets must be in binary mode. The second entry in the KONFIG table is the FORMAT II system function which the data set will have. At present there are four FORMAT II system functions, master input data set, master output data set, utility data set, and instruction data set. A master input data set is one which may be mounted prior to a FORMAT II execution and which may contain previously generated matrices. A code of two (2) is entered in KONFIG for this type of data set. A master output data set is one which may be saved at the end of a FORMAT II execution and which may contain matrices which are generated during a FORMAT II execution. A code of three (3) is entered in KONFIG for this type of data set. A master output data set may be used in a later FORMAT II execution as a master input data set. A utility data set is one which is used by the FORMAT II system as scratch storage. A code of one (1) is entered in KONFIG for this type of data set. The instruction data set is the data set in the information interface between the preprocessor monitor and the execution monitor. A code of four (4) is entered in KONFIG for this type of data set. The third entry in the KONFIG table for a data set is a code for the type of device which contains the data set. The codes are one (1), two (2), three (3), and four (4) indicating a device type of tape, disk, drum, and a data cell, respectively. The fourth entry in the KONFIG table is a code for the logical channel to which each device is attached. The codes are one (1) through ten (10) indicating logical channels A through J respectively. The fifth and final entry in the KONFIG table is the capacity in basic machine units (e.g., words) of the data set. A zero (0) indicates that the data set is assumed to be infinite in capacity. At present this characteristic is non-functional.

g) One final variable must be initialized in subroutine MRES. This variable is NUMR which is the number of data sets defined in the KONFIG table.

In assigning MAGIC II systems functions to the available data sets, the following rules must be followed: (1) The data set number of all available data sets (i.e., NPIT, NPOT, and all data sets defined in the KONFIG) must be unique. (2) All data sets defined in the KONFIG table must be available through the Fortran system and must be in binary mode. (3) Only one data set may be given the MAGIC II system function of the instruction data set. If the MAGIC II system function of instruction data set is not specified for any data set, one is selected from the utility data sets. (4) At least five (5) utility data sets must be specified exclusive of that which may be selected as an instruction data set. (5) Any number of master input or master output data sets may be specified.

h) The minimum machine configuration for the MAGIC System requires that eight (8) external storage units be available to MAGIC and assigned to the system via the KONFIC table in subroutine MRES. The first external unit must have the MAGIC function of instruction data set (INSTRN). The next two external units must be defined to have the MAGIC functions of master input data set (MASTRI) and master output data set (MASTRO). The last five units must be defined to have the intermediate utility data set MAGIC function (IOUTIL). Four of these utility data sets are used by the program as scratch tapes and the other one is necessary for intermediate matrix results.

It is important to note that if more than one data set is used for intermediate matrix results running time will be decreased. This can be accomplished by defining more than five external storage units to have the utility data set function (IOUTIL).

This concludes the initialization procedure in subroutines MRES. The only other area the systems analyst need be concerned with is the overlay structure of the MAGIC II system. The overlay structure on a subroutine basis is illustrated in Appendix I.

D. MACHINE RESOURCE DATA CARD

To assist the MAGIC systems analyst in initially implementing the MAGIC system or in temporarily modifying the existing logical machine configuration, several machine resources data cards are available. These cards are (1) the SETUP card, (2) the UPDATE card, and (3) the DELETE card. These cards are used in conjunction with the options on the $MAGIC card. The $MAGIC card defines the beginning of a MAGIC case. The options define the machine resources to be used during the running of the case. The form of the card is:

1	16
$MAGIC	STANDARD NEW CHANGE

Where the options are:

STANDARD - The standard machine configuration will be used for this run.

NEW - A totally new machine configuration is to be entered for this run using SETUP cards.

CHANGE - A change to the standard machine configuration is to be made for this run using either UPDATE or DELETE cards.

The machine resources data cards are defined as follows:

(1) SETUP cards are required if the NEW option has been specified on the $MAGIC card. This set of cards defines a new and temporary machine configuration. The form of the card is:

7
SETUP (n, function, device, channel, capacity)

where the arguments are:

n -the logical data set number

function-the MAGIC II system function to be assigned to this data set. This argument may be MASTRI indicating master input data set, MASTRO indicating master output data set, IOUTIL indicating intermediate utility data set, or INSTRN indicating the instruction data set.

device -the type of external storage device that the logical data set is to reside upon. This argument may be TAPE, DISK, DRUM, or CELL.

channel -the channel to which the device is attached. This argument is an alphabetic character from A thru J.

capacity-the capacity of the logical data set in basic machine units (e.g., words). A zero indicates an infinite capacity.

(2) The UPDATE card is used if the change option has been specified on the $MAGIC card. This card defines changes or additions to the standard machine configuration. The form of the card is:

7
UPDATE (n, function, device, channel, capacity)

Where the arguments are identical to those defined for the SETUP card.

(3) The DELETE card is used if the change option has been specified on the $MAGIC card. This card deletes a data set from the standard machine configuration. The form of the card is:

7
DELETE (n)

Where the argument is:

n - the logical data set number of the data set to be deleted.

The machine resources data cards immediately follow the $MAGIC card in the deck setup. For more information on the machine resources data cards refer to subroutines MRES, MRES1, MRES11, MRES2.

SECTION III

AGENDUM LEVEL ABSTRACTION INSTRUCTIONS

A. INTRODUCTION

An Agendum Level abstraction capability has been incorporated into the MAGIC System. The abstraction instructions for any type of analysis will be automatically generated for the user when he specifies the corresponding option on the $INSTRUCTION card. The Agendum library is expandable and the addition of more abstraction instruction sequences (Agendum) only requires the updating of subroutine AGENDM, and of course the Agendum Library itself. The use of an Agendum in no way restricts the user because he can include in his input deck his own abstractions to be merged with the selected Agendum.

B. MODIFICATIONS TO SUBROUTINE AGENDM

Subroutine AGENDM controls the selection from the Agendum library of the abstraction instruction sequence requested on the $INSTRUCTION card. At present, this subroutine has the capability to select six Agenduums: STATICS, STATICSC, STATICS2, DYNAMICS, DYNAMICSC and STABILITY. In order to add more options, the following variables and arrays require modification:

a. TYPE is the matrix which contains the names of the abstraction sequences in the agendum library. Increase the dimensions of this matrix and add the new Agendum names via DATA statements.

b. LTYPE is an array which contains the length of each Agendum name in the TYPE array. Increase the dimensions of this array and add the lengths of the new Agendum names via the DATA statement in sequential order corresponding to the names in the TYPE array.

c. NTYPE is the variable which defines the number of available Agendum in the library. Increase this variable to the number of names in the TYPE array.

C. SETTING UP THE AGENDUM LIBRARY

In subroutine MRES the variable NAGEND defined the FORTRAN logical unit number of the data set which contains the Agendum level abstraction instructions. Subroutine AGENDM expects the abstraction instructions in the library to have the same characteristics as card images, eighty (80) byte records.

1. Agendum Control Cards

Each sequence of Abstraction instructions must be proceeded by a control card which contains a name corresponding to a name in the TYPE array in subroutine AGENDM. For example, if the name STATICS appeared in the TYPE array then the abstraction instructions corresponding to the statics analysis would have to be proceeded by the control card $STATICS, the $ begins in card column 1 and there are no blanks allowed in the control card.

The last card signifying the end of all agendum is the $$END control card.

2. Examples of an Agendum Library

```
CC1
↓
$STATICS
{
     Statics abstraction instruction
}
$DYNAMICS
{
     Dynamics abstraction instruction
}
$STABILITY
{
     Stability abstraction instruction
}
$$END     (end of agendum library)
```

3. Examples of Agendum Usage

```
          CC1   CC7       CC16
          ↓     ↓         ↓
(a)       $MAGIC
          $RUN              GO
          $INSTRUCTION      STATICS
          $SPECIAL
           [Report Form Input Deck for .USER04. Instruction ]
          $END


(b)       $MAGIC
          $RUN
                INPUT     TAPE(OLD,1969)
                OUTPUT    TAPE(MAG,1970)
          $INSTRUCTION    DYNAMICS
                A=DYNAM.ADD.LMASS
                SAVE(MAG)DYNAM,LMASS,A
          $SPECIAL
           [Report Form Input Deck for .USER04. Instruction]
          $END
```

SECTION IV

MAGIC CATALOGUED PROCEDURE

The MAGIC Program can be executed using a catalogued procedure. For example, if the executable load module is stored in the technical library under the program name XY5630, the following catalogued procedure can be used for initial implementation.

```
//MAGIC       EXEC  PGM=XY5630
//FT01F001    DD    UNIT=SYSSQ,DISP=(NEW,DELETE),SPACE=(CYL,(5,4))
//FT02F001    DD    DDNAME=INPUT1
//INPUT1      DD    UNIT=SYSSQ,DISP=(NEW,DELETE),SPACE=(CYL,(5,4))
//FT03F001    DD    DDNAME=INPUT2
//INPUT2      DD    UNIT=SYSSQ,DISP=(NEW,DELETE),SPACE=(CYL,(5,4))
//FT04F001    DD    DDNAME=OUTPUT1
//OUTPUT1     DD    UNIT=SYSSQ,DISP=(NEW,DELETE),SPACE=(CYL,(5,4))
//FT05F001    DD    DDNAME=INPUT
//FT06F001    DD    SYSOUT=A
//FT07F001    DD    SYSOUT=B
//FT08F001    DD    DDNAME=OUTPUT2
//OUTPUT2     DD    UNIT=SYSSQ,DISP=(NEW,DELETE),SPACE=(CYL,(5,4))
//FT09F001    DD    UNIT=SYSSQ,DISP=(NEW,DELETE),SPACE=(CYL,(5,4))
//FT10F001    DD    UNIT=SYSSQ,DISP=(NEW,DELETE),SPACE=(CYL,(5,4))
//FT11F001    DD    UNIT=SYSSQ,DISP=(NEW,DELETE),SPACE=(CYL,(5,4))
//FT12F001    DD    UNIT=SYSSQ,DISP=(NEW,DELETE),SPACE=(CYL,(5,4))
//FT13F001    DD    UNIT=SYSSQ,DISP=(NEW,DELETE),SPACE=(CYL,(5,4))
//FT14F001    DD    UNIT=SYSSQ,DISP=(NEW,DELETE),SPACE=(CYL,(5,4))
//FT15F001    DD    UNIT=SYSSQ,DISP=(NEW,DELETE),SPACE=(CYL,(5,4))
//FT16F001    DD    UNIT=SYSSQ,DISP=(NEW,DELETE),SPACE=(CYL,(5,4))
//FT17F001    DD    UNIT=SYSSQ,DISP=(NEW,DELETE),SPACE=(CYL,(5,4))
//FT18F001    DD    UNIT=SYSSQ,DISP=(NEW,DELETE),SPACE=(CYL,(5,4))
//FT19F001    DD    DSNAMP=AGENDUM,DISP=OLD
//SYSABEND    DD    SYSOUT=A
```

If the problem program required any input or output tapes their definitions would be included into the procedure by overriding the DDNAMES: INPUT1, INPUT2, OUTPUT1 or OUTPUT2. For example, if one input tape and one output tape was required then the job step that envoked the catalogued procedure would be:

```
//JOB
//JOBLIB          DD      DSN=TECHNICL,DISP=SHR
//GO              EXEC    MAGIC
//MAGIC.INPUT1    DD      (Tape Definition)
//MAGIC.OUTPUT1   DD      (Tape Definition)
//MAGIC.INPUT     DD      *
```

```
          MAGIC PROBLEM DECK
//*
//END JOB
```

It is important to note that the above catalog procedure is just a sample and the actual procedure that you will implement may greatly differ from this example. The number of data sets defined and their meaning must correspond to the way the KONFIG array is defined in subroutine MRES.

SECTION V

MAGIC III

TEST RUN REPORT

Program Name MAGIC III

Date of Run _____ Report Number ____

Customer Name _____

Location _____

Machine Hours Used _____

Machine Configuration (include peripheral devices):

Type Operating System or Monitor Used (version, etc.):

(1) Objective of test run: (Discuss the routines or instructions tested and expected results.)

(2) Test run was:

_____ Satisfactory (go to Item 8)

_____ Unsatisfactory (go to Item 3).

(3) Check major reason for unsatisfactory run:

_____ Program design

_____ Program error

_____ Documentation error

_____ User error

_____ Machine failure

(4) Estimate of failure significance:

_____ Critical (preventing further progress - go to Item 5)

_____ Significant (can continue but must be corrected soon - go to Item 5)

_____ Minor (go to Item 6).

(5) Attach trouble supplement sheets to provide a discussion of run results.

(6) Has Development Team been notified of the problem prior to this report (i.e., during test session, immediately after, etc.)?

_____ No (go to Item 8)

_____ Yes -- by ____ phone; ____ memo; ____ both;

on _____ (go to Item 8).
date

(7) What action has been taken by Development Team?

(8) Additional comments, if any:

_____
Signature of Coordinator

MAGIC III- TEST RUN REPORT

(Trouble Supplement Sheet)

Program Name MAGIC III

Date of Run _____ Report Number ____

Customer Name _____

Location _____

INSTRUCTIONS:

1. Discuss run results, identify errors in program and/or documentation, include customer's comments or reactions, include supporting information such as source program, problem solution logic, memory dumps, copies of manual pages, etc.

2. Attach numbered and completed trouble supplements to appropriate MAGIC II TEST RUN REPORT, page 1. When complete, send one (1) copy to us and retain one (1) copy.

DISCUSSION:

APPENDIX X

SUBSYS CONTROL DOCUMENTATION

Table of Contents

Section		Page No.
I	INTRODUCTION and EXAMPLES	10.3
II	SUBSYS DOCUMENTATION	10.6
	A. Introduction	10.6
	B. Background	10.6
	C. Instructions for Use and Details on the SUBSYS Package	10.8
	D. Summary	10.12
III	DETAILS ON LNKSTK	10.13
	A. Introduction	10.13
	B. LNKSTK Data Card Format	10.13
	C. Examples	10.15
	D. Assembly Parameters	10.17
	E. Error Messages	10.17
	F. Dump Feature	10.18
	G. Restrictions	10.18
IV	LNKSTK AS AN EXECUTABLE SUBSYSTEM	10.19
	A. Introduction	10.19
	B. Instructions for Making LNKSTK Itself A Subsystem	10.19

Section		Page No.
V	DESCRIPTION OF THE MODIFIED .LOVRY WITH CPYLKO	10.21
VI	DESCRIPTION OF THE SEARCH ROUTINES	10.23
	A. Introduction	10.23
	B. Calling Sequence	10.23
	C. Assembly Parameter	10.24
	D. Error Message	10.24
VII	SUBSYS SUBROUTINES	10.25
	A. Introduction	10.25
	B. SUBSYS Overlay Chart	10.26
	C. List of SUBSYS Subroutine Functions . .	10.27
	D. Subroutine Documentation for SUBSYS . .	10.28

APPENDIX X

SECTION I

A. INTRODUCTION

The SUBSYS package consists of four subroutines written in MAP. The first subroutine, .LOVRY, is placed in the program deck, thus replacing the normal .LOVRY that IBSYS would have provided. The function of this altered .LOVRY is to receive control after the program has been loaded and to then copy the main link (LINK 0), which is now resident is core storage, onto a specified tape unit. Entry is then made into LNKSTK, the second SUBSYS subroutine, which will perform the function of copying LINK 0 from the tape written by .LOVRY onto another tape. Also, LNKSTK will place the overlay load file generated in the IBLDR phase and place it on the same tape as LINK 0. Upon completion of a LNKSTK execution, the entire program will be on tape in absolute load mode in two files; the first containing LINK 0 and the second containing the overlay structure. At this point the program may now be edited onto the System Library with the aid of the third SUBSYS subroutine, COPYDK, in which case it may be invoked by a $EXECUTE XXXXXX card, or the tape may be saved in its two file per program form accessible by the fourth SUBSYS subroutine, SEARCH. SEARCH has the capability of locating any program on a SUBSYS generated program tape, loading that program's LINK 0 into cord and then transferring control to it.

Usage of a SUBSYS generated program tape is accomplished by writing a FORTRAN load program that need contain only one executable statement, CALL SEARCH (6HPROGNM). This will cause SEARCH to locate the program, read the main link into core and execute the main deck. The overlay is contained in the next file and the modified .LOVRY, now resident in core with the main link, will control the loading of the overlay links. The modified .LOVRY will also substitute backspace file commands in place of rewind selections on the $ORIGIN cards in order to keep inside of the overlay file on the SUBSYS generated program tape.

A SUBSYS generated program tape may contain more than one program, each being identified and located by the name that was assigned to it by the User during the LNKSTK phase. Execution of each program is initiated by a call to SEARCH supplying the program name.

FORMAT II, with the structural Generative System insertion, is contained on one SUBSYS generated program tape as three separate programs, named AFMTII, BFMTII and .USER04., which are, respectively, the FORMAT II Preprocessor, the FORMAT II Execution Monitor and the Structural Generative System. Sequence of usage of the three programs is indicated on the following two lists, the first reflecting an application in which the .USER04. module (Structural Generative System) is accessed and the second reflecting an application in which the .USER04. module is not accessed.

B. EXAMPLES

1. .USER04. Module Accessed

The FORTRAN load program will cause the loading of --

(a) AFMTII, which upon completion of processing the input will issue a call to SEARCH to load -

(b) BFMTII, which upon encountering the .USER04. instruction will issue a call to SEARCH, to load -

(c) .USER04., which upon completion of matrix generation will issue a call to SEARCH to load -

(d) BFMTII, which upon completion of execution of the input abstraction instructions will call SEARCH to load -

(e) AFMTII, which will begin processing the next input data deck, if any.

2. .USER04. Module Not Accessed

The FORTRAN load program will cause the loading of --

(a) AFMTII, which upon completion of processing the input will issue a call to SEARCH to load,

(b) BFMTII, which upon completion of execution of the abstraction instructions will call SEARCH to load -

(c) AFMTII, which will begin processing the next input data deck, if any.

.USER04. and non-.USER04. data decks may be batched together on a single loading of the program.

Due to the fact that FORMAT II with the Structural Generative System is actually three separate programs, the necessary changes required for implementation on a given system must be made in each program. The same information must be supplied to subroutine MRES in AFMTII that was needed for direct machine control. Main programs BFMTII and .USER04. each have a subroutine RESET which must re-establish the size of blank common.

The sequence of operations to generate a SUBSYS program tape would be as follows:

1. IBSYS - start job
2. IBJOB - load AFMTII
3. LNKSTK - place AFMTII on SUBSYS program tape
4. IBJOB - load BFMTII
5. LNKSTK - place BFMTII after AFMTII on SUBSYS program tape
6. IBJOB - load .USER04.
7. LNKSTK - place .USER04. after AFMTII and BFMTII on SUBSYS program tape

It is extremely helpful, but not necessary, to the above procedure that LNKSTK be placed into IBSYS as a subsystem prior to executing the above procedure. Further examples are given in Section II, SUBSYS Documentation.

SECTION II

SUBSYS DOCUMENTATION

A. INTRODUCTION

The following section consists almost wholly of information contained in the distributed documentation supplied by SHARE regarding SUBSYS. Alterations have been made to enable one version of SUBSYS to be compatible on a stand alone 7090/94 or on a Direct Couple System 7040/7090 or 7044/7094.

Recognition for the bulk of the documentation is deserved by Mr. David E. Bluett of Westinghouse Electric Corporation, author of the original SUBSYS documentation.

This report describes a package of programs which will operate upon any FORTRAN IV program in such a way as to produce a program tape. The programs may be Overlay or non-Overlay, and the program tape may contain any number of such programs. The tape may then be used as a mounted program library (similar to a CHAIN tape in FORTRAN II) or may be edited directly onto the system tape to produce executable subsystem(s) under IBSYS.

B. BACKGROUND

The need for a package such as SUBSYS arose out of a desire to put some high-activity, high-load-time Overlay codes somewhere within the framework of IBSYS to proincreased accessibility and decreased load and peripheral times. An attempt was first made to insert a large Overlay code into IBLIB, with the intention of still going through IBLDR, but eliminating the large object deck. This method of attack ran into considerable troubles, the greatest of which was due to the limited size of the Subroutine Name and Dependency Tables when doing a Librarian edit. It became obvious that the most desirable situation would be the ability to say:

```
$EXECUTE      XXXXXX
```

thereby completely eliminating the need for input decks and any connection with IBJOB. Examination of the IBSYS manual showed that a subsystem under IBSYS should be an absolute assembly and obey certain rules. It seemed that a FORTRAN program, operating under IBJOB, already obeyed these rules

more or less by definition, since IBJØB is itself a subsystem. The only problem seemed to be the conversion of the FORTRAN code to an absolute assembly - a somewhat formidable task. However, it soon became obvious that the main link of an Overlay job (including all the Library) was itself an "absolute assembly" once it was loaded, and that the link tape, once written, was also in absolute scatter-loading format. The problem was now reduced to three parts: (1) dumping out the main link after it was loaded by IBLDR, (2) modifying the Overlay tape to correspond to proper subsystem rules, and (3) combining these two entities into one, ready for editing onto the system tape for use as a subsystem under IBSYS.

To solve part 1, a small program called CPYLKO (copy Link 0) was written which receives control immediately after execution and merely writes the main link out on tape. For convenience, this program has been made part of .LØVRY, which also had to be modified to properly control the new subsystems.

Parts 2 and 3 were solved by a separate program, LNKSTK (Link Stack), which modifies and combines the main link (as written by CPYLKO) and the Overlay tape (as written by IBLDR) to form a two-file program tape.

Tests were performed, and it was proved that the output tapes from LNKSTK could be edited onto the system tape and successfully used as subsystems under IBSYS. Even though these subsystems were placed on the system tape after IBJØB and SØRT, load time was reduced by about a factor of 4, and peripheral time (for input) reduced to essentially zero. Card shuffling errors in binary decks (a large source of lost runs) were eliminated as was a large portion of the total job setup time. Since LNKSTK has the ability to pack all of the record, execution time was usually improved, except in the cases of excessive link tape rewinding (.LØVRY now must do a "backspace file" instead of a "rewind").

Once this part of the package was operational, it was realized that the program tapes produced by LNKSTK could be mounted and operate just as well by themselves as they did as subsystems on the system tape. Since more than one complete program may reside on the program tape, all that was needed was a small loading routine to perform the functions of MYLDR, with the added feature of program selection. To provide this function, the SEARCH routine was written, and, in addition to subsystem generation, the SUBSYS package now provided the long-sought solution to the saving of Overlay tapes. It should be noted that the ability to save Overlay tapes came about essentially as a by-product of the process for subsystem generation.

C. INSTRUCTIONS FOR USE AND DETAILS ON THE SUBSYS PACKAGE

Assume that a User wishes to make a program tape from an existing PORTRAN TV Overlay program. Whether this tape will later be edited over as a subsystem or merely used as a "chain" tape is immaterial, since the technique for making the tape is the same in either case.

The special deck for .LØVRY (with CPYLKO) is inserted somewhere in the main link of the program, and the job is submitted for running in the following way:

(1) Any desired combination of $ATTACH or $SWITCH cards if needed.

(2) GØ (and any other options desired or needed) on the $IBJØB card.

(3) Only <u>one</u> link tape specified on the $ØRIGIN cards.

(4) The normal $ENTRY card (if any).

(5) No data (an end-of-file should immediately follow the $DATA card).

The program will load (the Overlay tape being written where directed by the $ØRIGIN cards) and execute by transferring to the pre-execution initialization section (PREE'). The first instruction in PREE' is TRA SYSIDR, 4, but a TTR to CPYLKO has been originally at SYSIDR in the IBSYS nucleus. The CPYLKO section of .LØVRY is thus entered immediately via SYSIDR.

The main link will now be written out as one big record on SYSCK2. If SYSCK2 is already the Overlay link tape, the output tape must be changed by altering an assembly parameter in CPYLKO. The size of the main link depends, of course on the last location used by this link, and this location is calculated in CPYLKO. The main link will be written from SYSLØC through this last word, preceded by a few communication and pointer words, and followed by an end-of-file. The output tape from CPYLKO is left un-rewound, and control returns to IBSYS via SYSRET. The cell SYSIDR in the nucleus has been altered by CPYLKO, so the next two cards in the deck (and the last two of this first phase of the job) must be:

$IBSYS

$REWTØPE

The next phase of the process is the combination of the

output tape from CPYLKO with the Overlay tape to form the final program tape. This combination is made by the Link Stack program, which will normally be the next job on the input tape. It is strongly suggested that LNKSTK (Link Stack) itself be made a subsystem under IBSYS, since this greatly simplifies the deck setup and eliminates the need for protecting the Overlay tape during the loading of LNKSTK. Instructions for making LNKSTK a subsystem are included as Appendix VIII, and this description will proceed on the assumption that this has been done.

After the $RESTØRE card, the cards are as follows:

$JØB

$EXECUTE LNKSTK

(LNKSTK data card, giving name, tapes, and options)

End-of-File card

Next may come either a return to the monitor for signing off, a system tape edit, or a test run on the new program tape using the SEARCH routine.

Once LNKSTK is loaded, it will read its data card containing the program name and the tape information required (the data card format is detailed in Appendix VII), and perform the following operations:

1. Rewind all pertinent units and read the main link as written by CPYLKO.

2. Modify this link into proper scatter-loading format and write it as one record on the specified output tape, followed by an end-of-file.

3. Read the Overlay tape, modifying the link records appropriately.

4. Write the modified links on the specified output tape, followed by an end-of-file.

5. Print a map showing input and output record counts, word counts, etc.

6. Rewind all pertinent units and exit.

The program tape is written and ready for use. It may be edited onto the system tape, loaded by means of the SEARCH program or dismounted for later use.

As a summary by way of example, assume that it is desired to make a program tape from a FORTRAN IV Overlay code called TSTJØB, edit this onto the system tape immediately following IBJØB, and then run a sample case. The deck set-up would be as follows:

```
1     8                16

$IBSYS
$JØB
$EXECUTE               IBJØB
$IBJØB jobnam          GØ,MAP, etc.
$IBLDR DECK1
     (start of decks for main link of program "TSTJØB"
     Compiles and/or assemblies may be done in this run).

.
.
.
$IBLDR .LØVRY
     (special deck of .LØVRY with CPYLKO included somewhere
     in main link).

.
.
.
$ØRIGIN  (start of link 1)

.
.
.
         (remainder of program)

.
.
.
($ENTRY card if normally included)
$DATA
End-of-file card
$IBSYS
$RESTØRE
$JØB
$EXECUTE               LNKSTK
         (Link Stack data card, explained in Appendix VII)
```

```
End-of-file card
$IBSYS
$IBEDT
      *EDIT          MAP,MØDS
      *PLACE         TSTJØB,2,1,2
      *REMARK        NØW IN NAME TABLE AS 2ND SUBSYSTEM, 2 FILES
      *REMARK        PØSITIØN TAPE AFTER IBJØB
FILE  *AFTER         IBJØB
      *REMARK        DUP IN TSTJØB FRØM SYSxxx
      *DUP           SYSxxx,SYSUT1,2
      *REMARK        ALL DØNE
End-of-file card
$IBSYS
$PAUSE               SET UP NEW SYSTEM TAPE, etc.
.
.
.
$IBSYS
$JØB                 TSTJOB MAY NOW BE USED AS A SUBSYSTEM
$EXECUTE             TSTJØB
         (sample data deck for TSTJØB)

End-of-file card
```

Obviously, any number of subsystems may be DUPed on in one edit, providing the proper *PLACE, *AFTER, and *DUP cards are used. In the IBSYS edit, the unit SYSxxx will be the LNKSTK output tape, which is one of the data card parameters.

As an alternate possibility, assume the activity of TSTJØB is not sufficiently high to warrant it inclusion as a subsystem, but that the load time is high enough to allow significant savings from the use of a program tape. The user therefore desires to make a program tape to be mounted on SYSLB2. It should be noted that program tapes produced by LNKSTK can only be mounted on one drive due to the changed structure of .LØVRY. In other words, if the program tape for TSTJØB is made to run on B5, then it must always run on B5. This so-called "running link tape" is one of the parameters on the LNKSTK data card and must be SYSLB2 for this version of SUBSYS. The following example illustrates the use of the SEARCH routine in conjunction with a program tape. The deck set-up is exactly the same as before, up through and including the EOF after the LNKSTK data card. The last three identical cards will be re-listed for continuity.

```
1    8              16

$EXECUTE            LNKSTK
```

```
          (Link Stack data card)

End-of-file card
$JØB
$EXECUTE            IBJØB
$IBJØB              GØ,MAP
$IBFTC CALL
C

          CALL SEARCH (6HTSTJØB)

C
          STØP
          END
$DATA
          (Sample data deck for TSTJØB)

End-of-file card
```

This example assumes that SEARCH has been placed on the IBJØB library (IBLIB). If this is not the case, the binary deck for SEARCH would follow the END card of the FORTRAN program above. Note that the calling sequence to SEARCH is similar to that used for CHAIN in FORTRAN II, except that the tape to be searched is omitted since it is assumed to be SYSLB2.

Search finds the specified program on SYSLB2 by name and scatter-loads it right on top of itself, leaving only enough to execute a transfer to SYSTRA which will commence execution of the desired program. The time saved when running with a mounted program tape and using SEARCH is obviously most dependent on the time used to hang the tape. The time taken to load SEARCH and its calling routine and to find and load the program is usually no more than .004 hours.

D. SUMMARY

The SUBSYS package consisting of .LØVRY with CPYLKO, LNKSTK, and SEARCH can provide considerable savings in setup, peripheral, and main-frame time when used with 7090, 7094, 7094/2 FORTRAN IV Overlay and non-Overlay codes.

Since no modifications are involved to IBSYS or IBJØP, SUBSYS should be more "version independent" than other packages available which do involve system mods. SUBSYS has been tested on both version 12 and version 13 installations. This is a tape-oriented package, and its value to a disk-oriented user is questionable. It is left to the disk user to make such an evaluation.

SECTION III

DETAILS ON LNKSTK

A. INTRODUCTION

The information needed by LNKSTK to produce a program tape is supplied by two sources: the communication words passed on by CPYLKO, and the LNKSTK data card. The communication words are obtained by LNKSTK when it reads the main link from tape, and are described in Appendix IX.

B. LNKSTK DATA CARD FORMAT

Field	Columns	Contents
1	1 - 6	The program name as it will appear in the first record of the program and on the $EXECUTE card or SEARCH argument. The name must be BCD, 6 character max., left adjusted in the field with trailing blanks if less than 6 characters. If this program is to become a subsystem, the name must be different from any other system record name.
2	8 - 13	Input tape on which LNKSTK may expect to find the main link as written by CPYLKO. This unit must be specified as SYSxxx, and would be SYSCK2 if running with the distributed version of CPYLKO which uses SYSCK2 for its output.
3	15 - 20	Input tape containing the Overlay links as written by IBLDR. This unit, which must also be specified by its SYSUNI name, is the tape presently containing the Overlay links, regardless of what SYSUNI it may have been (due to $ATTACH and $SWITCH cards) when the program was loaded. If this is not an Overlay job, the word "NØLINK" must be inserted in this field.
4	22 - 27	This field is another SYSUNI name which specifies the "running link tape", or the unit on which the program tape must be mounted when running with the SEARCH program and must be SYSLB2 for the distributed version of LNKSTK and SEARCH.

Field	Columns	Contents
		If this is not an Overlay job, the word "NØLINK" may be inserted in this field.
5	29 - 34	Output tape for LNKSTK, also a SYSUNI name. This name may be the same as that in Field 2, but may not be the same as the Overlay link tape in Field 3. It may be, but is not necessarily the same as the "running link tape" in Field 4.
6	36 - 39	If the record packing option is desired, this field should contain the word "PACK". If PACK is specified, all records for each Overlay link (written as 464 words by IBLDR) will be combined to form one long record. The .LRECT table generated by the loader is modified by LNKSTK to reflect the new positions of the links on tape. So called "remote sections" specified by $INCLUDE cards cannot be handled by LNKSTK*. This feature means that considerably less tape is used for the link section of the program, due to fewer record gaps. Link loading is considerably faster, usually resulting in an overall improvement in execution time. If this option is not specified, the records produced will be 465 words, a BCD name being added to each record (standard system record format). This option is meaningless for a non-Overlay job.
7	41 - 45 (if Field 6 was present)	Rewind options applying to the LNKSTK output tape. Either RB and/or RA, in either order, may occupy this field, or the field may be null.

* See Version 13 IBJØB manual, C28-6389-0, page 43.

Field	Columns	Contents
7	36 - 40 (if Field 6 was absent)	To permit the stacking of more than one program on the output tape, the rewinds are strictly controlled by these options. If RB (rewind before writing this program) is specified, LNKSTK will perform a rewind on the output tape immediately before it attempts to write the modified main link. If RA (rewind after writing this program) is specified, LNKSTK will finish writing the last Overlay link, write an EØF, and write a 3 word trailer record containing the word "ENDTPE". It will then rewind the output tape. This trailer record will cause the word ENDTPE, to scatter-load into SYSFAZ, enabling the SEARCH routine to recognize the end of the program tape. RB must be specified for the first (or only) program to be put on the output tape, while RA must be specified for the last (or only) program. If more than two programs are to be stacked on the output tape, any "middle" programs would have neither option specified to insure that no rewinds are performed.

All fields are separated by commas (or any other non-blank delimiter). The remainder of the card after col. 45 is available for comments. Fields 1 - 5 must be present in the columns assigned, while the last two fields are optional.

C. EXAMPLES

```
col. 1
     *
     TSTJØB,SYSCK2,SYSUT2,SYSLB2,SYSUT5,PACK,RB,RA
     SIFT  ,SYSCK2,SYSLB3,SYSLB2,SYSCK2,PACK
     SMALJB,SYSCK2,NØLINK,NØLINK,SYSLB2,RB
     BIGJØB,SYSUT3,SYSCK2,SYSLB2,SYSUT7,PACK,RA
```

The setup for stacking more than one program on the output tape is merely an extension of the case for one program. The order of jobs in the deck would be similar to the following:

```
First Program
LNKSTK run (with RB on the data card)
Second Program
LNKSTK run (with no rewind options)
      .
      .
      .
(nth Program)
(nth LNKSTK run, no rewinds)
      .
      .
      .
Last Program
LNKSTK run (with RA on the data card)
```

The output tape would, of course, be the same on all these LNKSTK data cards, while the other options may be as desired. Overlay and non-Overlay programs may be stacked on the same tape. A double EØF will follow a non-Overlay program, so that each program will be 2 files for the SEARCH routine (see Appendix X). If the system rewinds the LNKSTK output tape between jobs or job segments, these rewinds must be circumvented if more than one program is to be stacked on a given output tape.

In addition to writing the main link in scatter-load format, LNKSTK provides entries for the following communication cells:

1. Location 2 - TTR .LXSTR
2. Location 10_8 - TTR .FPTRP
3. Location 230_8* - A corrected skew-check mask.
4. SYSTRA - TTR PREEX (start of pre-execution initialization)
5. SYSGET - "IBSXEC"
6. SYSFAZ - program name from data card

* A "feature" has been added in IBSYS Version 13 such that any IØCP with a word count greater than 37777_8 which enters SYSTCH causes the record to be treated as if it were redundant. Entry 3 above corrects this, but is only done if LNKSTK is assembled for Version 13. See "Assembly Parameters".

7. SYSLØC - zero

8. .JLIN (line ctr.) - zero

9. SYSCUR - name of each record (main or Overlay as it is loaded

The program name enters SYSFAZ and SYSCUR when the main link is loaded, and the name remains in SYSFAZ throughout the run. Each link record stores its name in SYSCUR as it is loaded, so that the contents of SYSCUR will always represent the last record read.

The link record name is a combination of the program name and the link number if record packing is in effect, or the program name, link number, and record number if packing is not in effect.

Examples from "TSTJØB":

Packing: TSTJ04 (Link 4)
No Packing: TST721 (Link 7, Record 21)

All the link and record numbers will be BCD. The link will occupy 2 characters if it becomes greater than 9.

D. ASSEMBLY PARAMETERS

1. VRSION - assembled as 13 by a "SET". Pertains to the existence of SYSUT5-SYSUT9 and to skew-mask correction. See Appendix X, since the same parameter is contained in SEARCH to control I-Ø table assembly.

2. UNIT - assembled as SYSCK2. This is the output unit on which LNKSTK will dump itself if entered by a $ENTRY CPYLNK card. It must therefore be specified as the input unit on the LNKSTK data card when producing a program tape from LNKSTK itself (see Appendix VIII).

E. ERROR MESSAGES

If any error is detected during a LNKSTK run, a message:

ERRØR IN LINK STACK AT RELATIVE LØC XXXXX ØCTAL (SEE LISTING). CANNØT PRØCEED. is printed off-line. Examination of the comments on the listing will reveal the nature of the error. The message:

ERROR IN LINK STACK. FLUSH ANY REMAINING PARTS ØF THIS JØB HIT START FØ DUMP

ØPERATØR ACTIØN PAUSE

is printed on-line. Depressing the START key will cause a core dump via SYSDMP (AC, MQ, etc. are saved), but the operator is responsible for flushing the rest of the run.

F. DUMP FEATURE

If User modifications are made to LNKSTK, or if there seems to be trouble during a LNKSTK run, it may be desirable to obtain a core dump immediately after LNKSTK is through with its processing. To provide this facility, a feature has been added to LNKSTK such that the console entry keys are examined before LNKSTK returns to IBSYS via SYSRET. If any prefix key (S, 1, or 2) or any combination of prefix keys is down, LNKSTK will exit via SYSDMP rather than via SYSRET.

The operator must, of course, be informed that the key(s) are to be set before the termination of the LNKSTK run.

G. RESTRICTIONS

The fact that LNKSTK cannot handle "remote" sections specified on $INCLUDE cards has already been mentioned, as has the fact that only one link tape may be called for on the $ORIGIN cards.

Other problems may arise from certain record size limitations are imposed by SUBSYS and the systems which it must use. LNKSTK has a buffer size of 28000_{10} words (66540_8), and this represents the maximum size of any link record (the main link would usually be the largest record, since it contains all the library routines and possibly some named CØMMØN). When running strictly from a program tape, the SEARCH routine can load a record in excess of the LNKSTK maximum (actually 28246_{10} or 67126_8). However, things are not so simple when using the system editor. At the time of this writing, no documentation of any record size limitation has been found in either the coding for EDITØR or the IBSYS manual, but examination of the actual I-Ø command in EDITOR shows the following limits:

IBSYS Ver. 12 (EDITOR Ver.5) - 24607_{10} or 60037_8
IBSYS Ver. 13 (EDITOR Ver.6) - 23840_{10} or 56440_8

Analysis of a LØGIC or MAP will show whether a program is within these limits. Insertion of one or two redundant $ØRIGIN cards in the main link is usually all that is needed to bring the program back into line with LNKSTK and EDITØR.

In IBSYS Ver. 13, SYSLDR has been changed to check for skew-errors by insisting that no bits enter bit position 3, 19, and 20 of any scatter load IØCP. This has been corrected by the skew-mask described previously, which effectively allows SYSLDR to load a record of any size. Regardless of the method used, the practical size limit is still SYSEND-SYSØRG.

SECTION IV

LNKSTK AS AN EXECUTABLE SUBSYSTEM

A. INTRODUCTION

If LNKSTK is loaded from a binary deck, not only is the deck setup for making a program tape somewhat more complicated, but certain SYSUTx files (which might contain the Overlay links) must be protected during the loading of LNKSTK. The ideal situation is to have LNKSTK reside on the system tape as an executable subsystem. This adds no appreciable "bulk" to the system tape, since LNKSTK is only one file, consisting of one 2200 word record, and the deck setup of:

```
$JØB
$EXECUTE    LNKSTK
   (LNKSTK data card)
```

is certainly as compact and simple as could be desired.

B. INSTRUCTIONS FOR MAKING LNKSTK ITSELF A SUBSYSTEM

LNKSTK has its own built-in equivalent of CPYLKO, called CPYLNK, which may be entered in the case where it is desired to have LNKSTK operate on itself. Since this entry is not the general case, it is not made automatically as it is in the CPYLKO section of .LØVRY, but must be made by a $ENTRY card. The deck setup to make a program tape from LNKSTK itself and edit it over to the system tape immediately after IBJØB is as follows:

```
1       8     16

$IBSYS
$JØB
$EXECUTE      IBJØB
$IBJØB        GØ,MAP
$IBLDR LNKSTK
  (LNKSTK binary deck)
$DKEND LNKSTK
$ENTRY        CPYLNK
$DATA
```

```
                1   8        16
data card:      LNKSTK,SYSCK2,NØLINK,NØLINK,SYSCK2,RB,RA
                End-of-file card
                $IBSYS
                $JØB
                $IBEDT
                     *EDIT     MAP,MØDS
                     *PLACE    LNKSTK,1,1,2
                FILE *AFTER    IBJØB
                     *DUP      SYSCK2,SYSUT1,1
                End-of-file card
                $IBSYS
```

In this example, the $ENTRY card will cause LNKSTK to write itself out on SYSCK2 (this tape is an assembly parameter in LNKSTK) before transferring to its normal entry. It will then read its data card and proceed as it would on any non-Overlay job. Note the following on the data card:

1. The program name is specified as LNKSTK (similarly on the *PLACE card), and this is the name that must be specified on the $EXECUTE card when using the subsystem.

2. The input tape for the main (and only) link is specified as SYSCK2.

3. The NØLINK feature is specified in place of the normal link tape designations, signifying that this is not an Overlay job.

4. SYSCK2 is also used for the output tape, illustrating the fact that the output tape for LNKSTK may be the same as the main link onput tape.

5. The PACK option is not specified, since this would be meaningless for a non-Overlay job.

6. Since this is the only program to be put on the output tape, both the RB (rewind before) and RA (rewind after)

The new system tape, containing LNKSTK as the 2nd subsystem under IBSYS, will be produced on whatever unit is attached as SYSUT1.

SECTION V

DESCRIPTION OF THE MODIFIED .LØVRY WITH CPYLKO

The standard IBM routine .LØVRY, whose function is the loading of Overlay links, has been somewhat modified for use with the SUBSYS package. The largest change, of course, is the addition of the CPYLKO routine, which is discussed elsewhere in this write-up. Other changes are as follows:

1. The table of legal link tapes (UNITAB) has been reduced to one location, since now only one link tape is used, whether running as a subsystem or as a program tape. All general references to UNITAB (as a table) have been removed, and the UNITAB index in the .LRECT table is no longer examined. The single UNITAB cell in .LØVRY is now set by LNKSTK during its processing of the main link, the desired "running" link tape being specified on the LNKSTK data card. Since only one link tape may now be used, certain codes which have an extremely high activity of link loading and link tape rewinding may run considerably longer under this system, possibly enough to negate its worth. This is something that is best determined empirically.

2. All disk and hypertape coding has been removed for simplicity, since SUBSYS is a tape oriented package.

3. The IBSYS Version 13 Mod. which adds the skew error check is not included, since this has not proved to be troublesome in our installation. It may easily be inserted by the User if desired.

4. The subsystem (or program tapes) are now two files, the main link being one, and the Overlay links the second. .LØVRY must then skip over the EØF after the main link on the first entry and after each BSF. BSF's now replace rewinds when a rewind is requested by REW on the $ØRIGIN card.

5. If the PACK option is specified on the LNKSTK data card, all records for one Overlay link will be packed into one long record, thereby reducing the length of tape needed for the program and shortening the time for link loading. However, the .LRECT table produced by the loader will no longer reflect the correct record counts and tape positions for each link. This table is automatically modified in LNKSTK to reflect the true "one record per link" status of the link file on the tape. No change to .LØVRY is involved here.

Aside from these changes, .LØVRY is essentially the same. The number of words removed is about the same as the number of words added by the addition of CPYLKO. In the process of writing the main link from SYSLOC through its last word, CPYLKO also passes on to LNKSTK:

1. The address of PREEX
2. The addresses of .LASTR and .FPTRP
3. The address and length of the .LRECT table.
4. The address of UNITAB in .LØVRY.

All other information needed by LNKSTK is present on the data card.

The length of the main link is calculated at execution time in CPYLKO. A search is performed from SYSEND-1000 backward (towards location 0), looking for the first word that is not an STR 0,,0. This is assumed to be the last word of the main link. This is reliable as long as IBLDR performs as it is supposed to in its final section, and this method is certainly preferable to using an assembly parameter as was formerly done.

The standard error message in .LØVRY is written on first entry if the UCB's for the unit specified in UNITAB and SYSLB1 show both these units at load point.

SECTION VI

DESCRIPTION OF THE SEARCH ROUTINE

A. INTRODUCTION

The general function of the SEARCH routine has been described earlier in this manual. The scatter-load and redundancy-checking routine is origined at 72000_8 to prevent it from being destroyed as a large main link is scatterloading in. The initialization and table sections of the program will be destroyed in this process, since they are needed only once. The search for the program is dependent on the BCD name supplied in the calling sequence. The program name from tape will be scatter-loaded into SYSFAZ. When SYSFAZ becomes non-zero, SEARCH compares its contents with the name from the CALL. If they are the same, the scatter-load is allowed to continue, and, if not redundant, control then passes to the main link via SYSTRA. If the names are not the same, the scatter-load is immediately terminated and 2 files are skipped. The process is then repeated until either the program is found or the word "ENDTPE" enters SYSFAZ, signifying the end of the tape. If this trailer label is encountered, an error message is printed and the program exits via SYSDMP.

B. CALLING SEQUENCE

In FORTRAN or MAP: CALL SEARCH (Arg1)

where Arg1 is the program name as 6Hxxxxxx

It is strongly suggested that SEARCH be edited onto the IBJØB library as soon as it has been reassembled for the particular installation.

C. ASSEMBLY PARAMETERS

1. VRSIØN is assembled as 13 by a "SET", and represents the version of IBSYS in use. It pertains only to the existence of SYSUT5 - SYSUT9 and is used with IFT's and IFF's to control the assembly of the I-Ø tables.

2. BCDTAB - table of BCD SYSUNI names.

3. SYSTAB - table of SYSUNI indices.

4. RDSTAB - tables of read selects.

All of these I-Ø tables must be examined and made to conform to the installation I-Ø configuration.

D. ERROR MESSAGES

Due to a number of possible causes such as illegal tape designation, the word "ENDTPE" entering SYSFAZ, etc,, the message:

PROGRAM 'XXXXX' IS NOT ON SYSLB2 . . . SORRY

is printed on-line, followed by a dump. If the arguments look all right, the cell SYSFAZ should be examined.

If the main link record is still redundant after 10 tries, the message:

REDUNDANCY READING SYSLB2 . . . SEARCH DISCONTINUED is printed on-line, followed by a dump.

SECTION VII

SUBSYS SUBROUTINES

A. INTRODUCTION

The following is an example of how SUBSYS was implemented. It describes the subroutines which were added to the MAGIC System for SUBSYS control. The Overlay chart, B, should replace Figure I.7 in Appendix I for SUBSYS control.

STRUCTURAL SYSTEM OVERLAY CHART
(when using SUBSYS)

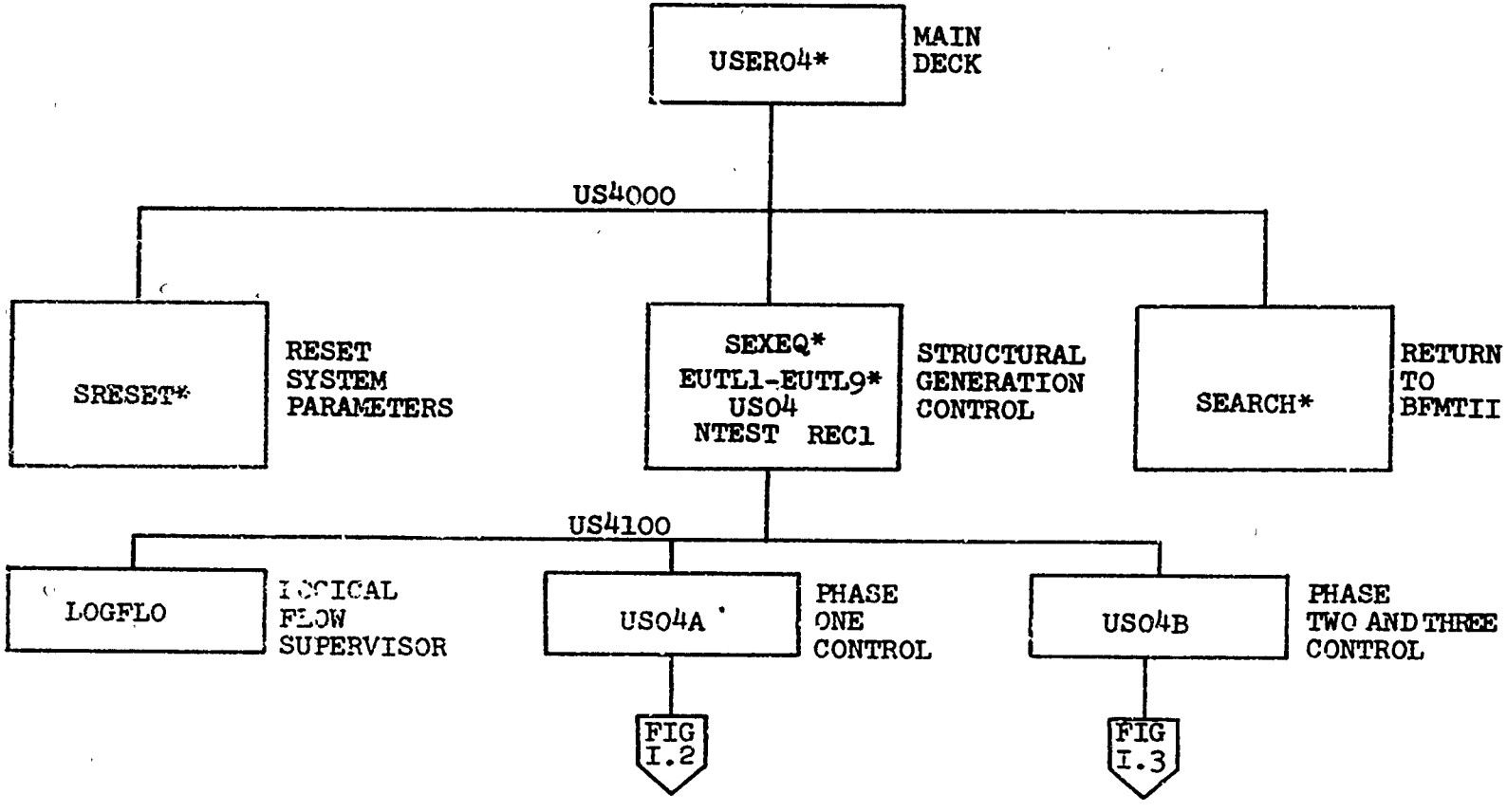


FIGURE I.1 CONTROL SECTION

C. LIST OF SUBSYS SUBROUTINE FUNCTIONS

.USER04.	(Main deck) Control reset of system parameters, call SEXEQ and return control to BFMTII
SRESET	Reset system input unit, system output unit, maximum matrix limit, size of work area, print control and re-establish blank common area
SEXEQ	Read and interpret .USER04. instruction and pass control to US04

D. SUBROUTINE DOCUMENTATION FOR SUBSYS

1. Subroutine Name: USERO4 (Main Deck)

2. Purpose: Provide main deck control under SUBSYS implementation

3. Equation and Procedures: Logical variable ERROR is set to false. Subroutine SRESET is called to reset system parameters. Subroutine SEXEQ is then called to execute the .USERO4. abstraction instruction. SUBSYS subroutine SEARCH is then called to return to the BFMTII program.

4. Input Argument: None

5. Output Argument: None

6. Error Returns: If logical variable ERROR is found to be true after performing subroutine SEXEQ, then an error message to this effect is printed and continuation of execution is attempted.

7. Calling Sequence: None

8. Input Tapes: None

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total storage is 120_8 (80_{10}).

12. Subroutine User: None

13. Subroutines Required: SRESET
    SEXEQ
    SEARCH

14. Remarks: None

1. Subroutine Name: SRESET

2. Purpose: Reset system parameters under SUBSYS implementation

3. Equations and Procedures: There are seven system parameters which must be reset due to operating under SUBSYS. They are:

   (1) NPIT : System input unit number
   (2) NPOT : System output unit number
   (3) KONST : Maximum matrix order capability
   (4) NWORK : Number of storages in work area
   (5) IPRINT : Output print control
   (6) WORK : Dimensioned work storage area (to be in blank common)
   (7) NINST : Unit number containing instructions

   NINST is defined to have a value of one.
   NPIT, NPOT, KONST, NWORK and IPRINT are reset by reading them from the return instruction on NINST. NINST is searched until the return instruction is located, then NINST is backspaced and the return instruction is read again, this time the required system parameters are read, thus resetting their values. The work storage area, WORK, is allocated into blank common by a COMMON statement in SRESET.

4. Input Arguments: None

5. Output Arguments:

   NINST : Fortran logical unit number containing instructions
   IPRINT : Output print control
   NPO1 : System output unit number

6. Error Returns: None

7. Calling Sequence: (NINST, IPRINT, NPO1)

8. Input tape: NINST - Abstraction instruction input tape.

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total storage is 163_8 (115_{10}).

12. Subroutine User: USER04

13. Subroutines Required: None

14. Remarks: None

1. Subroutine Name: SEXEQ

2. Purpose: Extract and separate the required information from the USERO4 instruction on the instruction tape

3. Equation and Procedure: The USERO4 instruction is read from the instruction tape into the common work storage area. From information contained in the first six words of the instruction record the succeeding data in the record is separated into its component sections and placed into the calling sequence to US04.

4. Input Arguments:

   NINST : Instruction tape number
   IPRINT : Output print control

5. Output Arguments:

   ERROR : Error condition indicator

6. Error Returns: None

7. Calling Sequence: (NINST, IPRINT, ERROR)

8. Input Tape: NINST - Abstraction instruction tape

9. Output Tapes: None

10. Scratch Tapes: None

11. Storage Required: Total Storage is 217_8 (136_{10}).

12. Subroutine User: USERO4

13. Subroutine Required: US04

14. Remarks: None

APPENDIX XI

DOCUMENTATION FOR ELEMENT INSERTION INTO THE MAGIC SYSTEM

TABLE OF CONTENTS

Section		Page No.
I	FINITE ELEMENT MATRIX SUBROUTINES DEFINITION RULES	11.4
	A. Subroutine Name	11.4
	B. Purpose	11.4
	C. Equations and Procedures	11.4
	D. Input Arguments	11.6
	E. Output Arguments	11.8
	F. Error Returns	11.9
	G. Calling Sequence	11.9
	H. Storage	11.9
	I. Subroutine User	11.9
	J. Subroutines Used	11.9
II	INSERTION OF FINITE ELEMENT MATRICES INTO MAGIC	11.10
	A. Revisions to .USER04. MODULE	11.11
	B. Revisions to the STRESS Module	11.12
	C. Revisions to the FORCE Module	11.13
	D. Revisions to the EPRINT Module	11.14
	E. Revisions to OVERLAY.	11.16
	F. Plugs and Subroutine Changes	11.17
	G. Checklist Tables for Use In Insertion	11.18

Section		Page No.
III	REVISIONS TO ELEM AND FELEM	11.22
	A. Revisions to FELEM.	11.22
	B. Revisions to ELEM	11.22
IV	EQUIVALENCE OF LOCAL WORK ARRAYS	11.27
	A. Work Array Equivalences for Plug Subroutines	11.27

SECTION I

FINITE ELEMENT MATRIX SUBROUTINES DEFINITION RULES

A subroutine must be generated which may be used by ELPLUG in order to generate the element matrices required for finite element analysis in MAGIC. This module may be written and checked out independent of MAGIC. The checked out routines may then be added to MAGIC by following the "INSERTION OF FINITE ELEMENT MATRICES INTO MAGIC", page 11.10.

For purposes of clarification, the standard subroutine writeup format is used in describing the necessary rules. This format is similar to the subroutine writeup format used in the Volume III Programmer's Manual.

A. SUBROUTINE NAME

Any subroutine name may be chosen. Later, when the module is inserted into MAGIC, the name may be changed to satisfy ELPLUG rules.

B. PURPOSE

To generate the finite element matrices required to generate statics, stress dynamics, or stability analysis, this module must be suitable for insertion into MAGIC.

C. EQUATIONS AND PROCEDURES

1. Equations

   a) Equations must satisfy the requirements and assumptions of displacement method finite element analysis.

   b) All matrices must be generated with system degrees of freedom ordered according to grid point, that is: $u_1, v_1, w_1, u_2, v_2, w_2, w_3, v_3, w_3$, etc. where $u_1 = u$ for grid point 1, $u_2 = u$ for grid point 2, etc.

c) The total number of degrees of freedom = NORD = number of grid points x number of degrees of freedom per point. For example, if an element has u,v,w, θ_x,θ_y,θ_z for each grid point and has three grid points, then NORD = 6 x 3 = 18.

2. Procedures

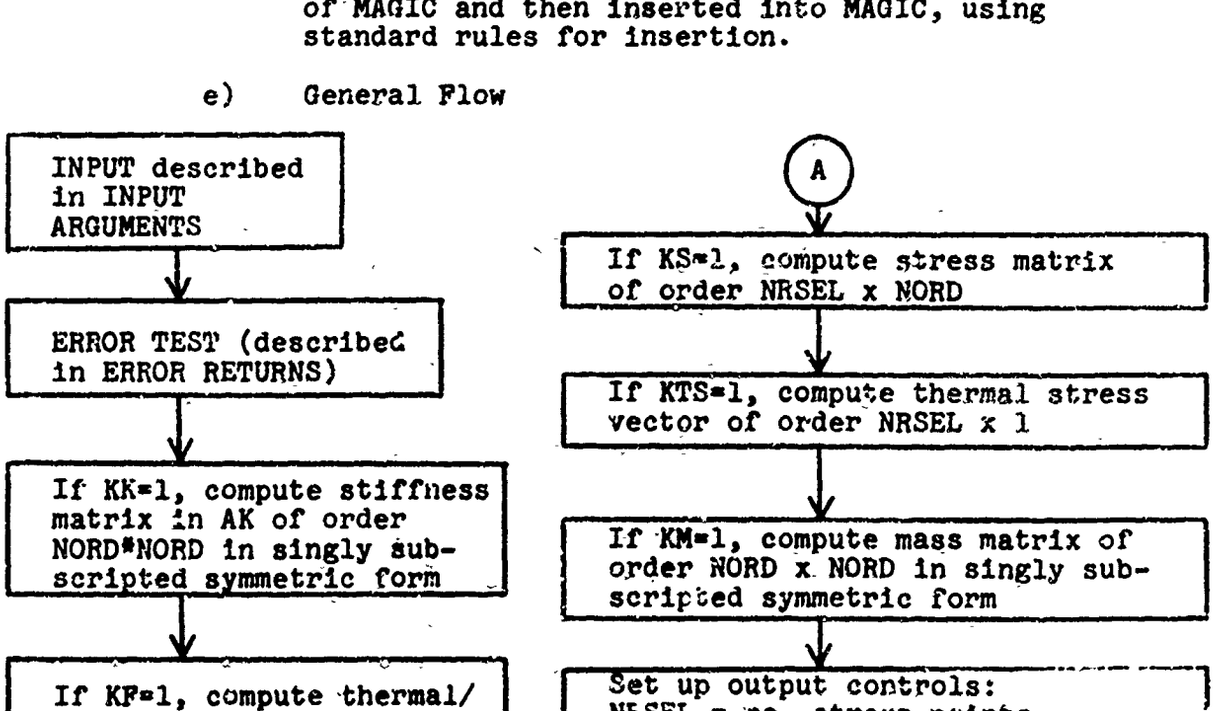
a) Element material properties, element grid point data and geometric properties are supplied as input through the argument list. The form of this input is described under "INPUT ARGUMENTS".

b) Using matrix methods, all element matrices must be generated in system coordinates. That is, all transformations required must be performed internal to the subroutine. A selection of matrix computations must be supplied, based on input selection controls.

c) Output matrices are supplied to the MAGIC system through the calling sequence, described under "OUTPUT ARGUMENTS".

d) This module should be checked out independently of MAGIC and then inserted into MAGIC, using standard rules for insertion.

e) General Flow



D. INPUT ARGUMENTS

1. Control Information

All suppression controls should be tested for value = 0 or 1. If the value = 0, do not compute the appropriate matrix. If the value = 1, do compute the matrix.

IPL = Internal element identification number

KK = Suppression control for element stiffness matrix

KF = Suppression control for element thermal and pressure load matrices

KS = Suppression control for element stress matrix

KTS = Suppression control for element thermal stress matrix

KM = Suppression control for element mass matrix

KN = Suppression control for element incremental stiffness matrix

2. Dimension Information

NNO = Number of grid points on element

NORD = Total number of degrees of freedom = order of stiffness matrix

3. Gridpoint Coordinate Data

XC = X coordinates of element gridpoints of length NNO

YC = Y coordinates of element gridpoints of length NNO

ZC = Z coordinates of element gridpoints of length NNO

TC = Grid point temperatures for element of length NNO

PC = Grid point pressures for element of length NNO

4. Material Property Input

These properties are input temperature interpolated element related material properties stored in a one-dimensional array: MAT. This array was generated in ELEM by computing the effective element temperature and then interpolating the material file tables for necessary values.

a) MAT Array - MAT(1) contains temperatures at which variables will be interpolated:

i) Elastic Properties

MAT(2)	E_x	Young's Modulus
MAT(3)	E_y	
MAT(4)	E_z	
MAT(5)	γ_{xy}	Poisson's Ratio
MAT(6)	γ_{yz}	
MAT(7)	γ_{zx}	
MAT(8)	α_x	Thermal Coefficients of Expansion
MAT(9)	α_y	
MAT(10)	α_z	
MAT(11)	G_{xy}	Shear Modulus
MAT(12)	G_{yz}	
MAT(13)	G_{zx}	

ii) MAT(14) - MAT(21) is reserved for future use.

iii)Other Parameters

MAT(22)	Mass Density - DENSTY
MAT(23)	Option for element print - WIPR
MAT(24)	Initial temperature - TZERO

E. OUTPUT ARGUMENTS

All symmetric arrays are stored such that only the lower half is stored by rows in single subscripted form.

1. Control Information

NERR = Error set control (described under ERROR RETURNS).

2. Dimension Information

NOINK = Number of words in symmetric matrix NORD x (NORD+1)/2

NRSEL = Number of stress components = number of rows in stress matrix

NSEL = Number of words in stress matrix = NRSEL * NORD

NMASS = Number of words in mass matrix.

3. Output Element Matrices

a) Stiffness Matrix - AKEL = Singularly subscripted array which represents storage of length (NORD x (NORD+1))/2. Elements of lower half of symmetric matrix of order NORD x NORD must be stored in system coordinates. The computation of this matrix should be suppressed if KK = 0.

b) If KF = 1, compute FTEL = thermal + pressure element load matrix column of order NORD x 1.

c) If KS = 1, compute SEL = stress matrix of order (NRSEL x NORD) where NRSEL = number of stress components.

d) If KTS = 1, compute thermal stress matrix = STEL = NRSEL x 1 matrix column.

e) If KM = 1, compute MASS matrix in same form as stiffness matrix.

f) NOINK = number of storages for stiffness matrix.

g) NRSEL = number of stress points on stress matrix.

h) NMASS = number of storages for mass matrix, if no mass matrix exists, set NMASS = 1.

4. EXTRA

EXTRA = a total of 5 input element properties is possible as input to MAGIC. Element thicknesses or other geometric properties are obtained from this array.

F. ERROR RETURNS

Set NERR = 0 if no error
Set NERR = 1 if finite element number (IPL) is incorrect
Set NERR = 2 if number of Nodes (NNO) is incorrect
Set NERR = 3 if order of matrix = NORD is incorrect.

G. CALLING SEQUENCE

```
Call (Subroutine Name) (IPL,NNO,NORD,KK,KF,KS,KTS,KM,
     KN,XC,YC,ZC,TC,PC,MAT,EXTRA,NOINK,NRSEL,NSEL,NMASS,
     AKEL,FTEL,SEL,STEL,AMASS,NERR)
```

H. STORAGE

All singular subscript arrays should be dimensioned (1). SEL must be dimensioned (NRSEL,NORD).

I. SUBROUTINE USER

ELPLUG must be updated to accept this routine. "INSERTION OF FINITE ELEMENT MATRICES INTO MAGIC" should be consulted for the changes necessary to ELPLUG and MAGIC.

J. SUBROUTINES USED

Any subroutines may be used and written for use in this routine.

SECTION II

INSERTION OF FINITE ELEMENT MATRICES INTO MAGIC

As MAGIC is a General Purpose Structural Analysis Program, certain sections can be considered as modules. Revisions to the program is accomplished by insertion of new subroutines or modules. This concept of inserting or "plugging" finite element matrices into a program was originally a concept in 1956 of Turner of Boeing. Thus the term "plug" means inserting different finite element equations into MAGIC.

Revisions to MAGIC Include the Following:

A. Revisions to .USER04. Module

B. Revisions to STRESS Module

C. Revisions to FORCE Module

D. Revisions to EPRINT Module

E. Revisions to OVERLAY of Program

F. Revisions to the Plug Subroutines Themselves.

A. REVISIONS TO .USER04. MODULE

1. Subroutine ELEM

   a) Revise the "Table of Contents" of elements, if necessary. Consult current listing for present form.

   b) Refer to "REVISIONS TO ELEM AND FELEM" in order to update the DATA and DIMENSION statements for the arrays:

      NUMOLD
      IPLNO
      NDSEL

   c) Increase the value of NUMPLG by +1 for each plug added.

2. Subroutine FELEM

   a) Update the "Table of Contents" of elements, if necessary. See Table I.

   b) Revise the data statement for the NEWNUM array, if necessary. Refer to "REVISIONS TO ELEM AND FELEM", page 551.

3. Subroutine ELPLUG

   a) Modify the computed GOTO statement so that control passes to statement number MN00 when IPL assumes the value MN. (NOTE: MN is the one or two digit plug number.)

   b) Insert the CALL PLUGMN with appropriate calling sequence at statement number MN00.

   c) Insert instructions to bypass the grid point axis transformation, if necessary. These transformations must be skipped in all plugs which handle grid point axis transformation inside the plug itself.

B. REVISIONS TO THE STRESS MODULE

1. Subroutine STRES2

a) Increase the DIMENSION of the array PLUGS. This array contains the plug number of all the element types available to the MAGIC system.

b) Update the variable NPLUGS. This variable should be the same as the dimension of the PLUGS array.

c) Add the new element type plug number to the DATA statement which defines the PLUGS array.

d) Update the GOTO statement which transfers control to the WRITE statement which writes the heading for the stresses of that particular element type. The statement number to transfer to is calculated by IPL*1000.

e) Add the WRITE statement with format number IPL*1000+1. Then define the following variables:

i) NSC = the number of stress components for this element
ii) IFMT = the updated value of NPLUGS
iii) KFMT = 1,2,3,4 or n depending on the format needed to write out the stress values. The actual format will be discussed in Section V.

f) Add the statement GOTO 320.

g) The heading printed out for the stresses of this element should conform to the format of all the other headings.

C. REVISIONS TO THE FORCE MODULE

1. Subroutine FORCE2

a) Increase the DIMENSION of the array PLUGS. This array contains the plug number of all the element types available to the MAGIC system.

b) Update the variable NPLUGS. This variable should be the same as the dimension of the PLUGS array.

c) Add the new element type plug number to the DATA statement which defines the PLUGS array.

d) Update the GOTO statement which transfers control to the WRITE statement which writes the heading for the forces of that particular element type. The statement number to transfer to is calculated by IPL*1000.

e) Add the WRITE statement with format number IPL*1000+1. Then define the following variables:

i) NFC = the number of components of force for this element
ii) IFMT = the updated value of NPLUGS*100
iii) KFMT = 1,2,3,4 or n depending on the format needed to write out the force values. The actual format will be discussed in Section V.

f) Add the statement GOTO 320.

g) The heading printed out for the forces of this element should conform to the format of all other headings.

D. REVISION TO THE EPRINT MODULE

1. Subroutine EPRINT

a) Increase the DIMENSION of the array PLUGS. This array contains the plug number of all the element types available to the MAGIC system.

b) Update the variable NPLUGS. This variable should be the same as the dimension of the PLUGS array.

c) Add the new element type plug number to the DATA statement which defines the PLUGS array.

d) Update the GOTO statement which transfers control to the WRITE statement which writes the headings for the stresses or forces of that particular element. If IPRT=1, then net element stresses are to be written. The statement number to transfer to is calculated by 600+IPL. If IPRT=2 then net element forces are to be written. The statement number to be transferred to is calculated by 700+IPL.

e) Add the WRITE statement with format number 800+IPL for stresses and 900+IPL for forces. For both stresses and forces define the variables:

i) NC = number of stress or force components
ii) IFMT = the updated value of NPLUGS for stresses and for forces it equals NPLUGS*100.
iii) KFMT = 1,2,3,4 or n depending on the format needed to write out the stress and force values. The actual format is discussed under STRPRT revisions below.

f) Add the statement GOTO 200.

g) The headings to be printed should be exactly the same as those written in subroutines STRES2 and FORCE2.

2. Subroutine STRPRT

Subroutine STRPRT is called by STRES2, FORCE2, and EPRINT.

a) This routine contains the format statements necessary to write the stress or force values. At present, there are four different formats available, defined by FMT1, FMT2, FMT3 and FMT4. The value of KFMT as defined in STRES2, FORCE2 and EPRINT will point to one of these formats. If any of the present formats are not applicable for the printing of the values of a new element type then the following must be done.

   i) Define a new format statement in a DATA statement. Give it the name FMTn. Set KFMT=n.

   ii) DIMENSION this format.

   iii) Update the GOTO statement which transfers to the WRITE statement which uses the new FORMAT FMTn. Calculate the statement number by KFMT*100.

b) The column headings that are to be printed for the new stresses and forces must also be added to this routine. Update the GOTO statement which transfers control to the correct WRITE statement. For stresses, the statement number is IPL*1000 and the format number of the write statement is IPL*1000+1. For forces the statement number is IPL*1000+3 and the format number is IPL*100+2.

c) The FORMAT statement which contains the headings for the columns should follow a format similar to those already included in the routine.

E. REVISIONS TO OVERLAY

The overlay will have to be revised whenever new subroutines are added to MAGIC. This overlay structure may be a function of the particular version on a particular machine. There is no standard procedure but a general guideline is available: NEWPLUGS may be placed on new links which are on the same level as existing plugs since only one plug will be necessary in core at one time.

F. PLUGS AND SUBROUTINE CHANGES

1. Obtain listing of PLUG which has been written and checked out by following the rules under "Finite Element Matrix Subroutine Definitions Rules", page

2. Equivalence all working dimensions to WORK storage by referring to "Equivalence of Local Work Areas in MAGIC", page

3. Insert this card immediately after the subroutine PLUGMN statement:

   ```
   COMMON NPIT,Φ,KONST,DUMMY(7097),
   WORK(NLAST)
   ```

   When NLAST is defined as the last location of the WORK storage array referenced in Item (2) above.

4. REPEAT(3) above for every subroutine used by PLUGMN.

G. CHECKLIST TABLES FOR USE IN INSERTION

This table contains all of the revisions listed. These tables should be used in order to be sure that all steps have been completed.

When revised item has been completed, write an X in the space provided.

A. .USER04.

1. Subroutine ELEM

a. Revise the "Table of Contents" ____

(1) REVISED ____

(2) No Revision Necessary ____

b. Revise NUMOLD

(1) DATA Statement ____

(2) DIMENSION Statement ____

Revise IPLNC

(1) DATA Statement ____

(2) DIMENSION Statement ____

Revise NDSEL

(1) DATA Statement ____

(2) DIMENSION Statement ____

c. Increase NUMPLUG by +1 ____

2. Subroutine FELEM

a. Table of Contents Revision ____

b. NEWNUM Array Revision ____

3. Subroutine ELPLUG

a. Computed GOTO Statement NO. _____ ____

b. Call PLUGMN - Plug No. _____ ____

c. Grid Point Axis Transformation

(1) Included Inside PLUG ____

(2) Not Included Inside PLUG ____

B. STRESS MODULE

1. Subroutine STRES2

a. Increase dimension of PLUGS _____

b. Update NPLUGS _____

c. Update PLUGS DATA Statement _____

d. Update GOTO Statement for Element Stress Headings

GOTO Statement No. _____ _____

e. Add WRITE Statement _____

Redefine:

(1) NSC _____

(2) IFMT _____

(3) KFMT _____

f. Add statement GOTO 320 _____

g. Insert New Heading for Stress Print _____

C. FORCE MODULE

1. Subroutine FORCE2

a. Increase Dimension of PLUGS _____

b. Update NPLUGS _____

c. Update PLUGS DATA Statement _____

d. Update GOTO Statement for Element Force Headings

GOTO Statement No. _____ _____

e. Redefine:

(1) NFC _____

(2) IFMT _____

(3) KFMT _____

f. Add Statement GOTO 320 _____

g. Insert New Heading for FORCE Print _____

D. EPRINT MODULE

1. Subroutine EPRINT

   a. Increase Dimension of PLUGS ____
   b. Update NPLUGS ____
   c. Update PLUGS DATA Statement ____
   d. Update GOTO Statements for Element Stress and Force Headings ____
   e. Add WRITE Statements ____

      Redefine:

      (1) NC ____
      (2) IFMT ____
      (3) KFMT ____

   f. Add Statement GOTO 200 ____
   g. Insert Headings which are same as for STRES2 and FORCE2 ____

2. Subroutine STRPRT

   a. Define FMTn DATA Statement ____

      Set KFMT=n ____

      Dimension FMTn ____

      Update GOTO Statement
      GOTO (KFMT*100) _____ ____

   b. Update FORMAT and GOTO Statements for Print of Column Headings

      STRESSES:

      Format No. (IPL*1000+1) _____ ____
      GOTO (IPL*1000) _____ ____

      FORCES:

      Format No. (IPL*2000+2) _____ ____
      GOTO (IPL*1000+3) _____ ____

E. REVISIONS TO OVERLAY

1. Revise OVERLAY of Program ____

F. PLUGS AND SUBROUTINE CHANGES

1. Set up and Checkout PLUG Subroutines ____
2. Equivalence WORK Storages ____
3. Insert: COMMON NPIT,Φ,KONST,DUMMY,WORK in all Subroutines ____

SECTION III

REVISIONS TO ELEM AND FELEM

A. REVISIONS TO FELEM

1. Defining NEWNUM (contained in FELEM)

The logical grouping of elements selected for MAGIC is shown in Table I. The "plug" numbers are shown in Table I also. Using Table I as a reference, the MAGIC numbering system is arranged in ascending order, inserting a zero for an unidentified element. This results in data for a NEWNUM array shown in Table II. Referring to Table I and Table II, let I = plug number, J = MAGIC (NEWNUM). Then the array NEWNUM is defined by: NEWNUM(J) = I. NUNUM only must be revised if new group is added.

B. REVISIONS TO ELEM

1. Defining NUMOLD

At a given point in time, NUMOLD is shown in Table III. It is defined by the following: NUMOLD(I) = J. When I and A above have the same meaning as in (A) above, NUMOLD must be revised where a new plug is added.

2. Defining IPLNO

IPLNO represents the group number of existing MAGIC elements and must be extended for any new element matrix set. This array represents the NUMOLD array after zeros have been deleted.

3. Defining NDSEL

NDSEL represents the number of stress points coded for existing elements in MAGIC. This number is the one actually coded in the plug and corresponds to NRSEL described under reference "Definition of Calling Sequence for ELEMENT Matrix Subroutines."

For example, referring tc Table I, if a sandwich plate is to be added, I = 18, J = 28; that is, PLUG 18 representing group No. 28 is to be added. Suppose that only 5 stress points are considered for this element. Then the revised statements and arrays are shown in Table III(A).

TABLE I

TABLE OF CONTENTS FOR FELEM ELEMENT DESCRIPTION

I PLUG	J MAGIC (NEWNUM)	NUMBER NODES	DESCRIPTION OF ELEMENT
1	21	8	Quadrilateral Shell
2	20	6	Triangular Shell
3	22	3	Triangular Plate of Constant Stress
4	23	4	Quadrilateral Plate of Constant Stress
5	30	2	Torodial Ring
6	40	3	Triangular Ring
7	11	3	Frame
8	41	4	Trapezodial Ring
9	42	1	Core (Ring)
10	50	4	Tetrahedron
11	24	4	Shear Panel (Translation Only)
12	26	3	Sandwich Plate
13	51	6	Triangular Prism
14	25	4	Shear Panel (Translation and Rotation)
15	10	2	Axial
16	12	3	Stiffener
17	27	3	Triangular Plate
18	28	4	Quadrilateral Plate
19	43	2	Truncated Cone
20	52	8	Rectangular Prism
22	13	-	Incremental Frame

Example for the "Quadrilateral Plate":

I = Plug No. = 18

J = Group No. = 28

TABLE II

NEWNUM DATA STATEMENT IN
SUBROUTINE FELEM

```
DIMENSION NEWNUM(5a)
DATA NEWNUM/
1       0,0,0,0,0,0,0,0,0
2       15,7,16,22,0,0,0,0,0,0
3       2,1,3,4,11,14,12,17,18,0
4       5,0,0,0,0,0,0,0,0,0
5       6,8,9,19,0,0,0,0,0,0
6       10,13,20,0,0,0,0,0,0,0/
```

TABLE III

DATA STATEMENTS IN SUBROUTINE ELEM

These tables represent MAGIC with the following plug numbers: 1,2,4,5,6,14,17,18

```
(a)   DIMENSION NUMOLD (17)
          DATA NUMOLD        /21,20,0,0,30,40,11,0,0,
                              0,0,0,0,25,0,0,27,/

(b)   DIMENSION IPLNO (7)
          DATA IPLNO         /21,20,30,40,11,25,27/

(c)   DIMENSION NDSEL (7)
          DATA NDSEL         /40,32,15,4,12,1,8/
                             NUMPLG = 7
```

TABLE III(a)

Represents NUMOLD, IPLNO and NDSEL after addition of quadrilateral plate (example):

```
(a)   DIMENSION NUMOLD (18)
          DATA NUMOLD        /21,20,0,0,30,40,11,0,0
                              0,0,0,0,25,0,0,27,28/

(b)   DIMENSION IPLNO (8)
          DATA IPLNO         /21,20,30,40,11,25,27,28/

(c)   DIMENSION NDSEL (8)
          DATA NDSEL         /40,32,15,4,12,1,8,5/
                             NUMPLG = 8
```

SECTION IV

EQUIVALENCE OF LOCAL WORK ARRAYS

The MAGIC System uses a large area of blank common to store all temporary and work arrays for the .USER04. module. The array is set up in routine ELPLUG and modified in each of the plugs. All local arrays used by subroutines called by a plug may be defined in this large common area by an equivalence statement in the plug. Thus no additional storage is required after the common work array has been defined.

NWORK = the maximum number of WORK storages available to the MAGIC System. The value of this parameter is set in the MAGIC routine MRES.

NLAST = NWORK - 7096 = total number of work storages available for equivalence of local arrays.

A. WORK ARRAY EQUIVALENCES FOR PLUG SUBROUTINES

1. Obtain "plug" listing, with array map.

2. Check argument list of plug and determine dimensional arrays which appear in argument list. These arrays are not local to plug and therefore should not be equivalenced.

3. Remaining arrays must now be equivalent to work array in plug. All these arrays are local to plug itself.

4. Check dimension statement and equivalence of all these local arrays successively.

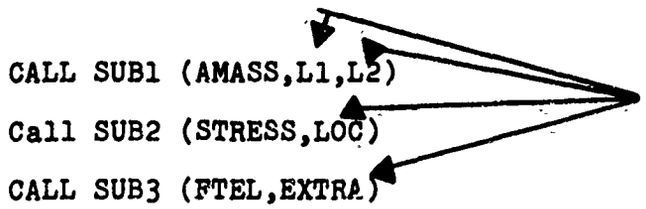
5. Now search through all array maps of subroutines called by plug and place all arrays local to the called subroutines (which are dimensioned 10 or above) in the argument list. Equivalence these arrays to the work array in the plug itself. Enter the appropriate dimension statement in the plug.

6. Now check equivalence storage map and equivalence the longest number of each equivalence group to the next available location of WORK. Enter dimension statements if necessary. Leave the original equivalence statements in.

7. See the following example where subroutines SUB1, SUB2, SUB3 with local arrays L1, L2, LOC, EXTRA are to be called by PLUGX. For this example NWORK = 13000. NLAST = 1300 - 7096 = 5900.

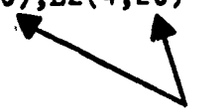
```
PLUGX(AMASS,STRESS,FTEL,ETC.  )

COMMON NPIT,NPOT,KONST,DUMMY(7097),WORK(5904)
DIMENSION L1(50), L2(420), LOC(100), EXTRA(300)
EQUIVALENCE (WORK(1),L1(1))
    "       (WORK(51),L2(1))
    "       (WORK(131),LOC(1))
    "       (WORK(231),EXTRA(1))
    "       (WORK(531), -----  )
```



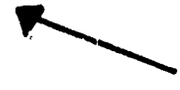
Search all subroutines called by PLUGX for local arrays and put the dimension and equivalence in PLUGX to conserve storage.

```
SUBROUTINE SUB1(AMASS,L1,L2,...)
DIMENSION AMASS(...),L1(50),L2(4,20)
```

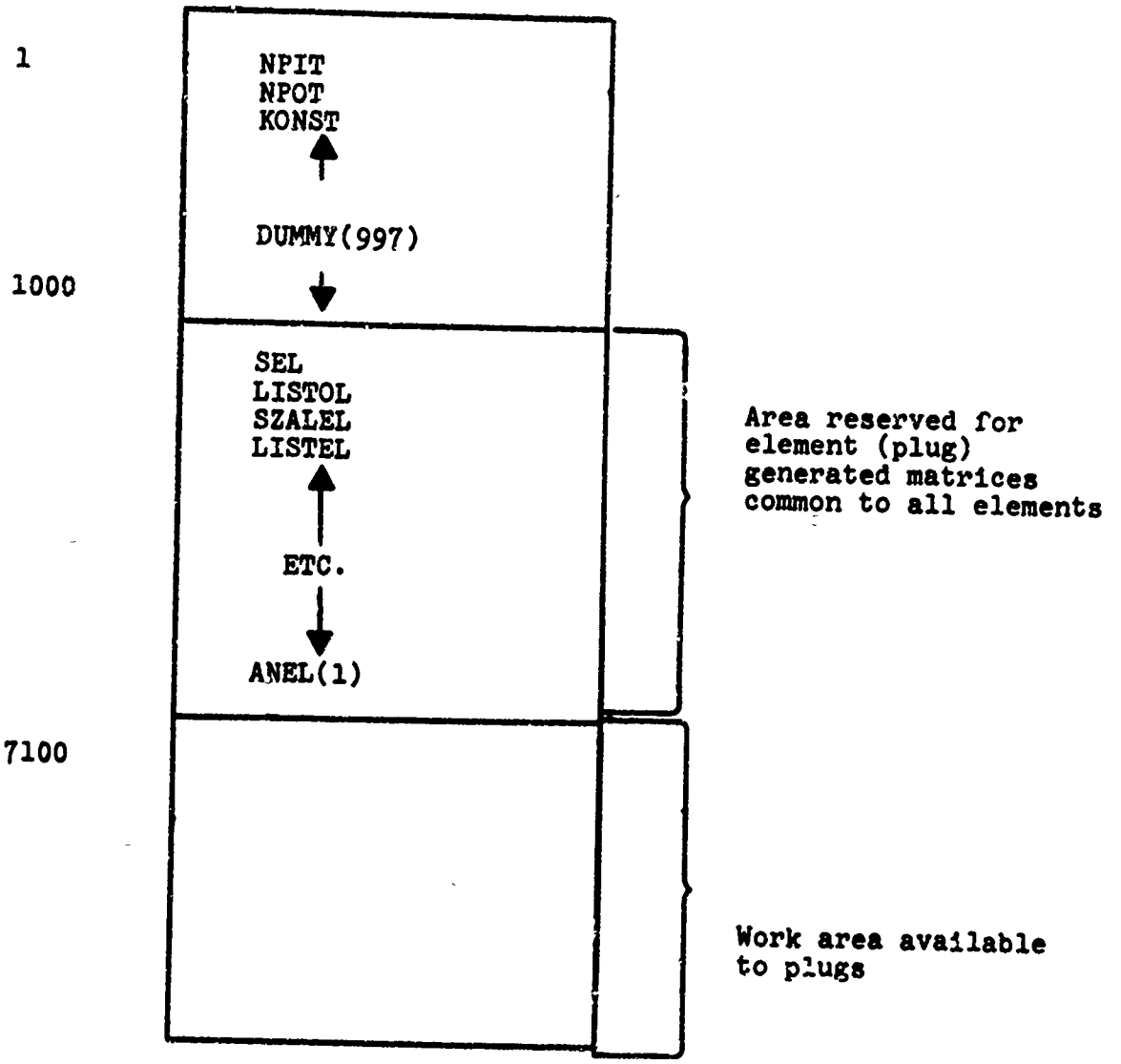



```
SUBROUTINE SUB2(STRESS,LOC,...)
DIMENSION STRESS(...),LOC(100)
```

```
SUBROUTINE SUB3 (FTEL,EXTRA,...)
DIMENSION FTEL(...),EXTRA(300)
```



COMMON WORK AREA FOR .USER04.



REFERENCES

1. DeSantis, Daniel, "MAGIC: An Automated General Purpose System for Structural Analysis; Volume III: Programmer's Manual," Report No. AFFDL-TR-68-56, Volume III, Air Force Flight Dynamics Laboratory, Wright-Patterson Air Force Base, Ohio, March 1968.

2. Cogan, J.P., "FORMAT II - Second Version of FORTRAN Matrix Abstraction Technique; Volume II: Description of Digital Computer Program," Report No. AFFDL-TR-66-207, Volume II, Air Force Flight Dynamics Laboratory, Wright-Patterson Air Force Base, Ohio, December 1966.