USAAMRDL-TR-75-39D

DESIGN CRITERIA FOR ELASTOMERIC BEARINGS

Volume IV - Programmer's Manual

Thiokol/Wasatch Division
A Division of Thiokol Corporation
Brigham City, Utah 84302

March 1976

Final Report

Approved for public release;
distribution unlimited.

Prepared for

EUSTIS DIRECTORATE
U. S. ARMY AIR MOBILITY RESEARCH AND DEVELOPMENT LABORATORY
Fort Eustis, Va. 23604

EUSTIS DIRECTORATE POSITION STATEMENT

The data contained in this report are the results of an effort designed to improve the state of the art of elastomeric bearing design for helicopter rotor head applications. The products of this effort are a design manual and a computer program based on finite-element techniques. The results of this program are contained in the following four volumes.

Volume I - Final Report
Volume II - Design Manual
Volume III - Program User's Manual
Volume IV - Programmer's Manual

Volume I contains the development and background information used in producing the design manual.

Volume II presents design considerations and procedures, bearing applications, methods of analysis, and techniques for predicting bearing performance.

Volumes III and IV contain the computer code and examples of problems showing sample inputs and outputs.

The products of this effort provide a good foundation for building a comprehensive manual and computer code for the design and analysis of elastomeric bearings for helicopter rotor head applications. It was recognized at the onset of this program that both the manual and the code would be first editions. The results of this effort were expected to define areas requiring further development. Further investigations coupled with feedback from users and/or evaluators are expected to provide material for upgrading the content and format of the manual and codes.

Mr. John Sobczak of the Military Operations Technology Division served as project engineer for this effort.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: USAAMRDL-TR-75-39D
2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle): DESIGN CRITERIA FOR ELASTOMERIC BEARINGS, Volume IV. Programmer's Manual.
5. TYPE OF REPORT & PERIOD COVERED: PROGRAMMER'S MANUAL
6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s): DON H. LEE
8. CONTRACT OR GRANT NUMBER(s): DAAJ02-73-C-0091

9. PERFORMING ORGANIZATION NAME AND ADDRESS: Thiokol/Wasatch Division, A Division of Thiokol Corporation, Brigham City, Utah 84302
10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 62205A119 01 013EK

11. CONTROLLING OFFICE NAME AND ADDRESS: Eustis Directorate, U.S. Army Air Mobility Research and Development Laboratory, Fort Eustis, Virginia 23604
12. REPORT DATE: March 1976
13. NUMBER OF PAGES: 59

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)
15. SECURITY CLASS. (of this report): UNCLASSIFIED
15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

DA-1-F-162205-A-119

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Volume IV of a 4-volume report

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Elastomeric Bearings
Helicopter Rotor
Natural Rubber
Finite Element Analysis
Stability
Service Life

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This document provides information necessary to the programmer to support and modify program S3359. The general philosphy of the program and detail on some sections are contained herein.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

TABLE OF CONTENTS

INTRODUCTION

This program was developed at Thiokol Corporation to assist in the design of elastomeric bearings.

It is a simplified version of a larger system. Because of this, it has several "vestigial organs" in it. For example, the material property arrays and routines are fully set up for orthotropic materials even though there is no orthotropic capability in the program. Other instances of this will be noticed in the program. These vestigial functions have some advantage in that the capability of the program can be expanded more easily if and when desired.

The program has been designed to be modular in nature. Thus particular functions are restricted to certain modules. This also aids in maintenance and modification. Each of the modules is described in later sections.

In general non-FORTRAN I/O has been used in the program. This has two main advantages. First, it is faster, in general two to four times faster, than binary FORTRAN I/O. Second, some things are possible in the special routines that can not be done in FORTRAN. For example, subroutine TDRR uses the NOTE/POINT capability to retrieve large records directly into their place in core in an order different from that in which they were written.

MODULE DESCRIPTIONS

The basic philosophy of the program has been to try to separate functions into modulues. Following are descriptions of each of the modules in the program. These are not intended to be comprehensive but, along with the flow charts, to give an introduction to the system.

THE CONTROL SECTION

This part of the program is the driver that controls the logic flow through each of the modules.

The program begins execution in a small FORTRAN routine whose only function is to call the GTMAIN routine. The GTMAIN routine uses a conditional getmain to get a large block of core storage to be used as a work area by the program and then passes control to subroutine CONTRL.

Subroutine CONTRL actually controls the logic of the program. It reads the title and the general input section and calculates basic dimensions. A set of logical variables contained in common CNTRL specifies which modules will be entered and how the program will loop. These logicals are initialized in a block data routine and then are changed according to program options.

THE INPUT MODULE

This module controls the reading of the input data from cards. All input to the program is via subroutine FREFRM which allows data to be entered in a free field format. Each of the routines in this module calls FREFRM and then interprets the string of data returned to it.

Subroutine IIMOD is the first routine called. It sets up the basic arrays out of the work area and then passes control to subroutine INPUT.

Subroutine INPUT checks the flag record against a table of valid flags and then calls the proper routine to handle that section of input. At the end of the input it calls subroutine PRGEOM, which prints the geometry and creates the geometry and load files (IGEOM and ILOAD).

This module uses for scratch files two pairs of "flip/flop" files, that is, the data is passed back and forth between the two files with the file names being interchanged (flip/flopped) between each pass. One set is used for the integer code information, and one set is used to hold the load information. This technique avoids the use of large in-core arrays to hold all of this information. It is a trade-off between core size and run time.

THE SOLUTION MODULE

This module creates the element stiffness matrices and force vectors, assembles them, and solves for the displacements. The solution scheme used is a Gaussian scheme, and only the semi-bandwidth for a row of elements is in core at any one time.

The control routine for this module is SOLMOD. It sets up the arrays and calls subroutine FORSOL for the forward part of the solution, and then reallocates the arrays and calls subroutine BAKSUB for the backward part of the solution, and the extraction of the displacements. If a large deformation iteration is involved, then subroutine LDCTRL is called to control that function.

Subroutine FORSOL loops through for each element in the system. The geometry and load files are read for each row of nodes, and the loads are adjusted if there is an incremental loading situation. Material properties are calculated for each element and interpolated for if there is strain or thermal dependence. A logical array, LSTP, is set up which contains a reference of flags giving the degrees of freedom on each node; this array is used in the assembly of the solution matrix and the extraction of the roots. The displacement boundary conditions and nodal point forces are applied to each element stiffness matrix. The solution matrix is assembled and partially reduced for a row of elements at a time and the equations are written out on a data file.

A special I/O routine, TDRR, is used to handle the equations. It writes large records directly from core without using buffers. It also has the capability of recovering the records in any order. In this program a record number is assigned to each record as it is written in FORSOL and then we use the record numbers in the inverse order to recover the records in the inverse order in BAKSUB.

Subroutine BAKSUB controls the extraction of the displacements. It reads the equations from the data file in the reverse order to the way they were written and completes the Gaussian reduction solution scheme. The array LSTP is used to tell what displacements from the solution vector go into which displacement array. At the end the displacements are placed in the proper order and written on the displacement file, IDSP. If this is an asymmetric loading run, they will also be written on the accumulation file, IACM.

During the forward solution, the element loads are written on the load print file, IELOAD, for later use. The element material properties are written on the material property print file, IEMAT. The element stiffness matrices are written on a scratch file, ISCR3, and passed to the stress module for use in the accuracy check, energy, and reaction force calculations. The stiffness matrix and force vector are also written on file ISTFF for use in the stress calculations.

THE STRESS MODULE

This module calculates stresses and strains for each element. Calculation of energy, reaction forces, and accuracy check is also done here. If we have an incremental loading run, subroutine INCLDC is entered to control that function.

Subroutine STSMOD is the module driver; it calculates dimensions for the basic arrays and passes control to subroutine STRDRV.

Subroutine STRDRV calls the particular routines needed to calculate the stresses and strains and then calls the routines for energy, reaction forces, and accuracy check calculations.

THE PRINT MODULE

This module prints the data produced by the other modules. Various routines are called to print the specific sets of data.

The data contained on files IELOAD, IEMAT, IDSP and IPRNT are all printed as requested. The element loads are on IELOAD. The element material properties are on IEMAT. The nodal displacements are on IDSP. Stresses and strains, energy, accuracy check, and reaction forces are on IPRNT.

Common SPRINT is used by several of the routines as a work area. Its only function is to order the variables.

THE PLOT-ACCUMULATION MODULE

This module has two functions. First, it produces deformed geometry plots on command from the input. Second, it accumulates and prints the data from the asymmetric loading run. These two functions may be interspersed so that plots can be made of the accumulated data.

All plots are made from data on scratch file ISCR1. If this file has not been generated by the accumulation function, it must be created by subroutine CISCR, which is there for that purpose.

Two sets of flip-flop files are used in the accumulation. One set is for displacements and the other set is for stresses and strains. As each set of data is read in, the old summation is read, the new data is multiplied by the proper trig function and added to it, and then the accumulated data is written out. At the end of each pass, the files change roles.

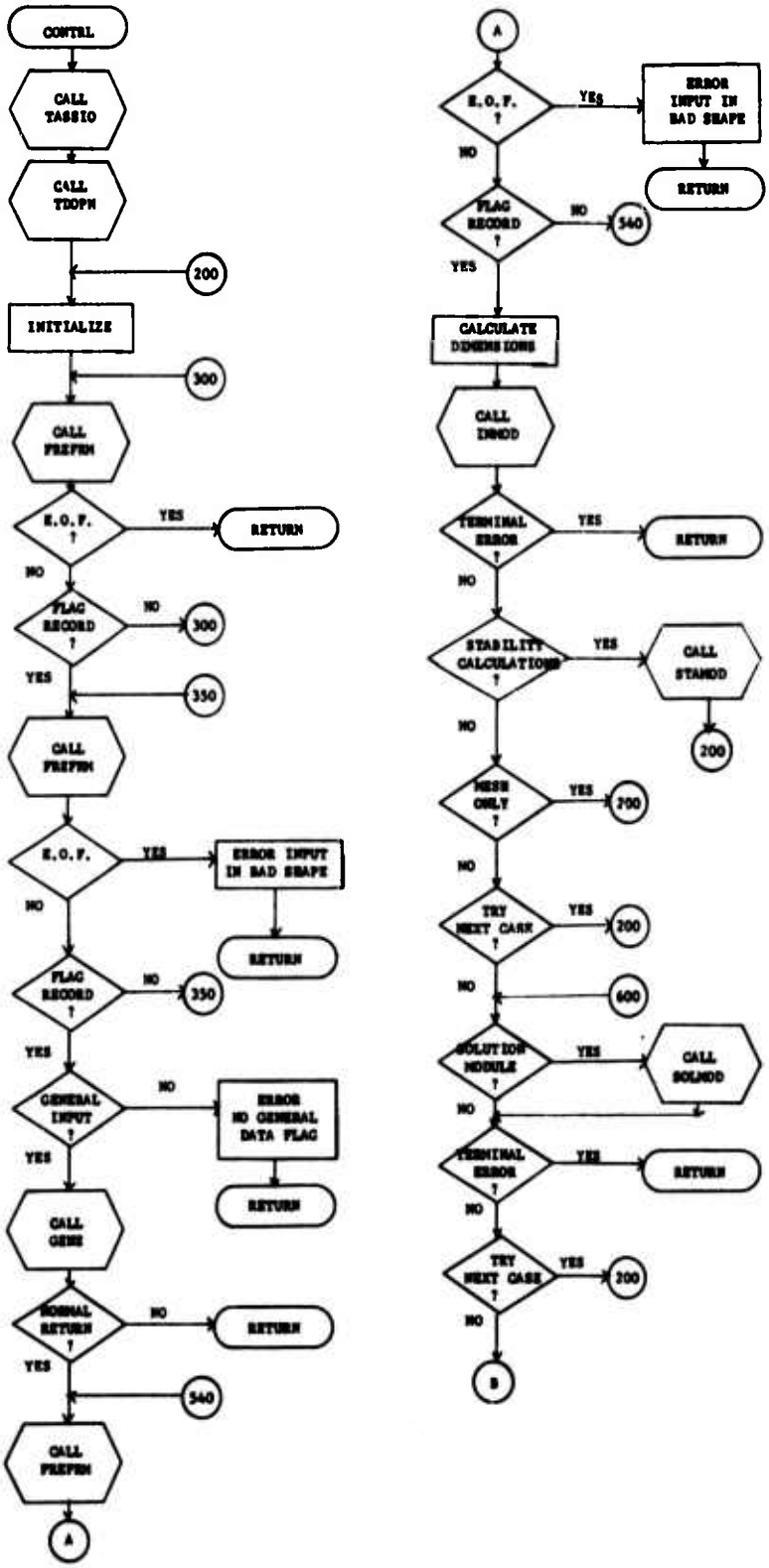
When making a transverse cut through the body, the points are not necessarily picked up in the desired order and must be sorted. The routines that do this are so written that they will make use of all available core to avoid unnecessary I/O operations. When merging is necessary, the number of required merges is written out on the audit file.

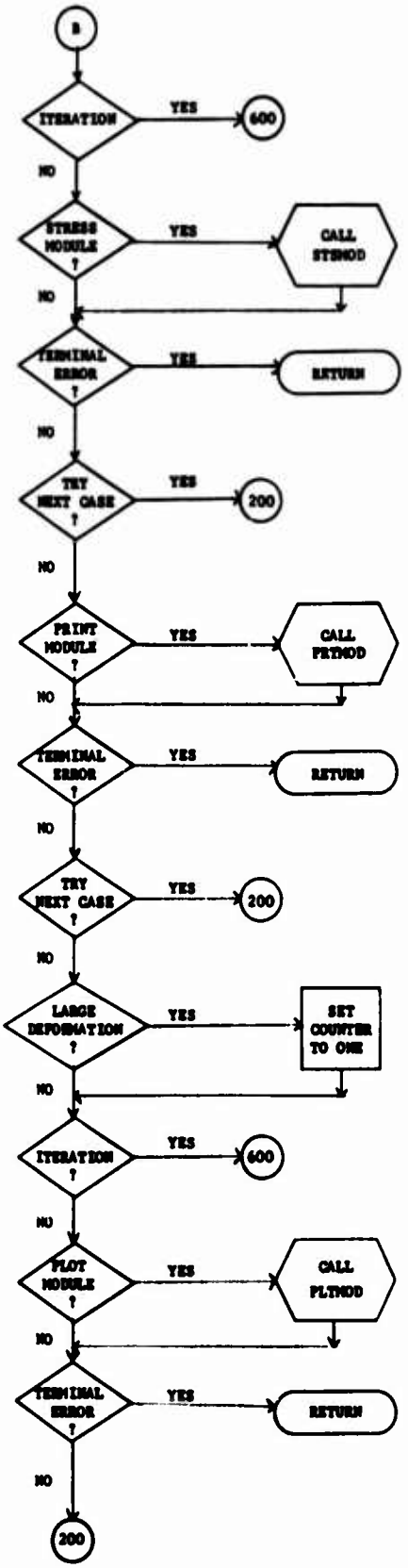
THE STABILITY MODULE

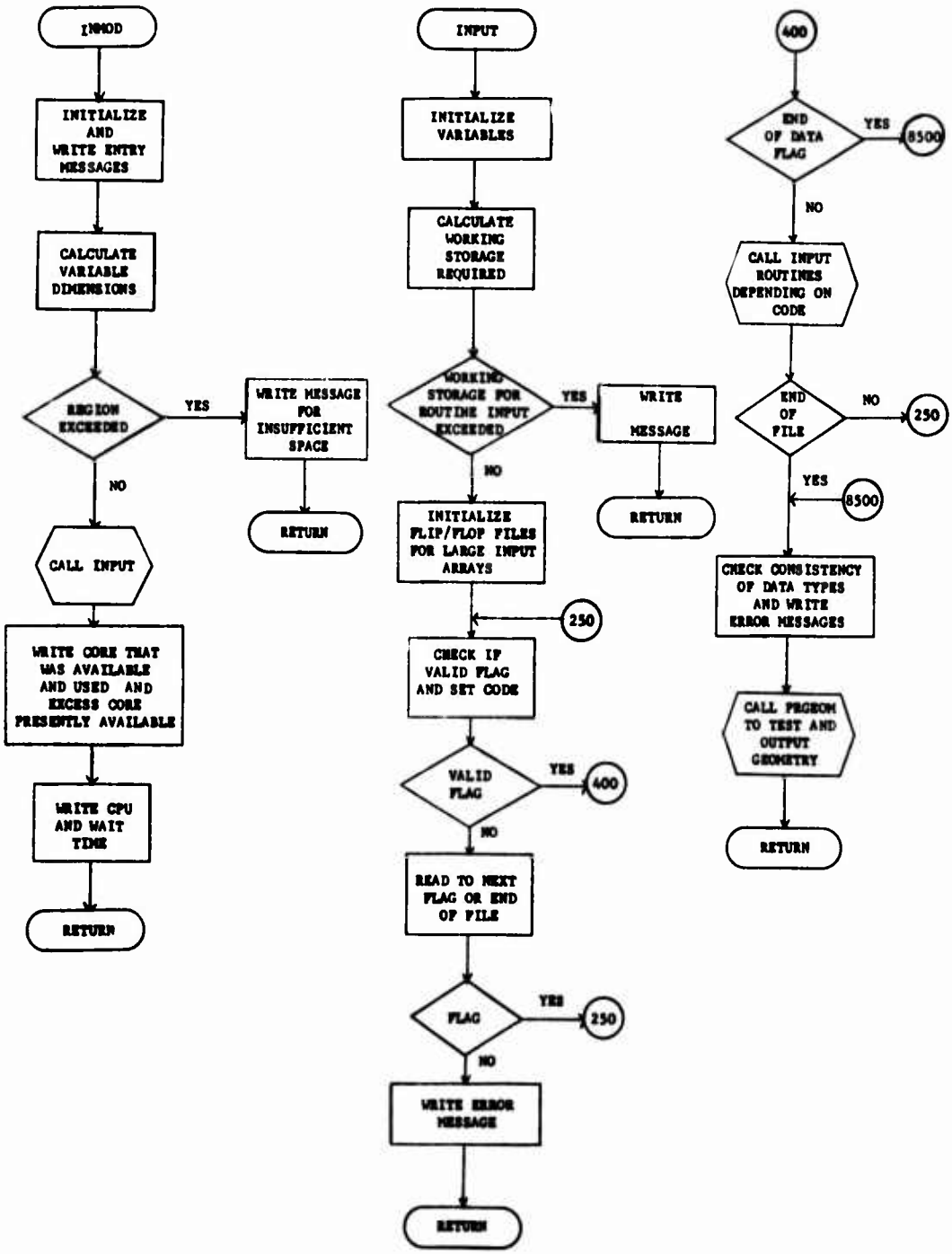
This module consists of only one routine. Due to the simplicity of the calculations, the lack of a requirement for large amounts of core, and its different nature from the rest of the program, this routine simply prints as it calculates. This allows the program to run in a minimum region when this option is used.

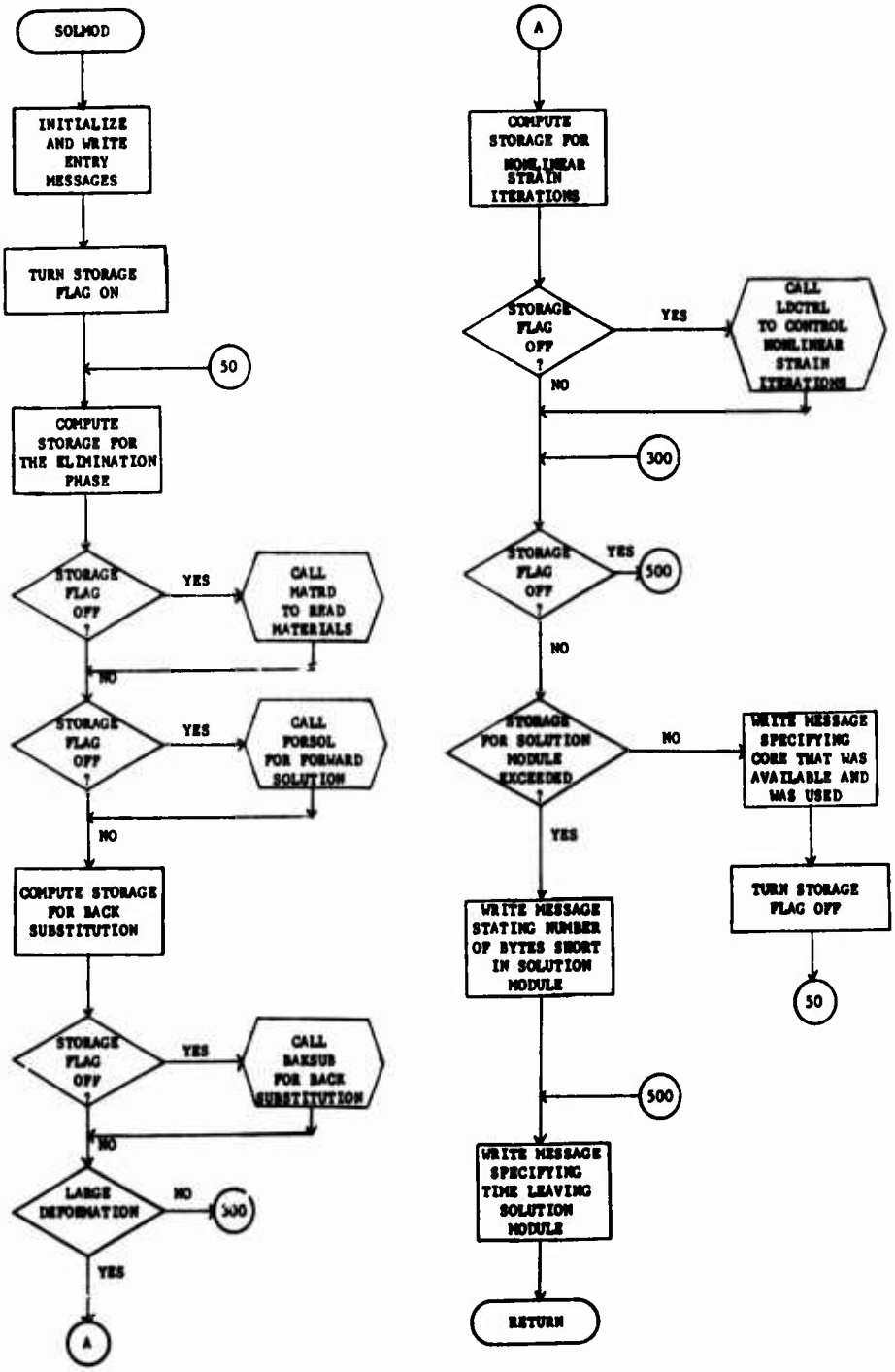
FLOW CHARTS

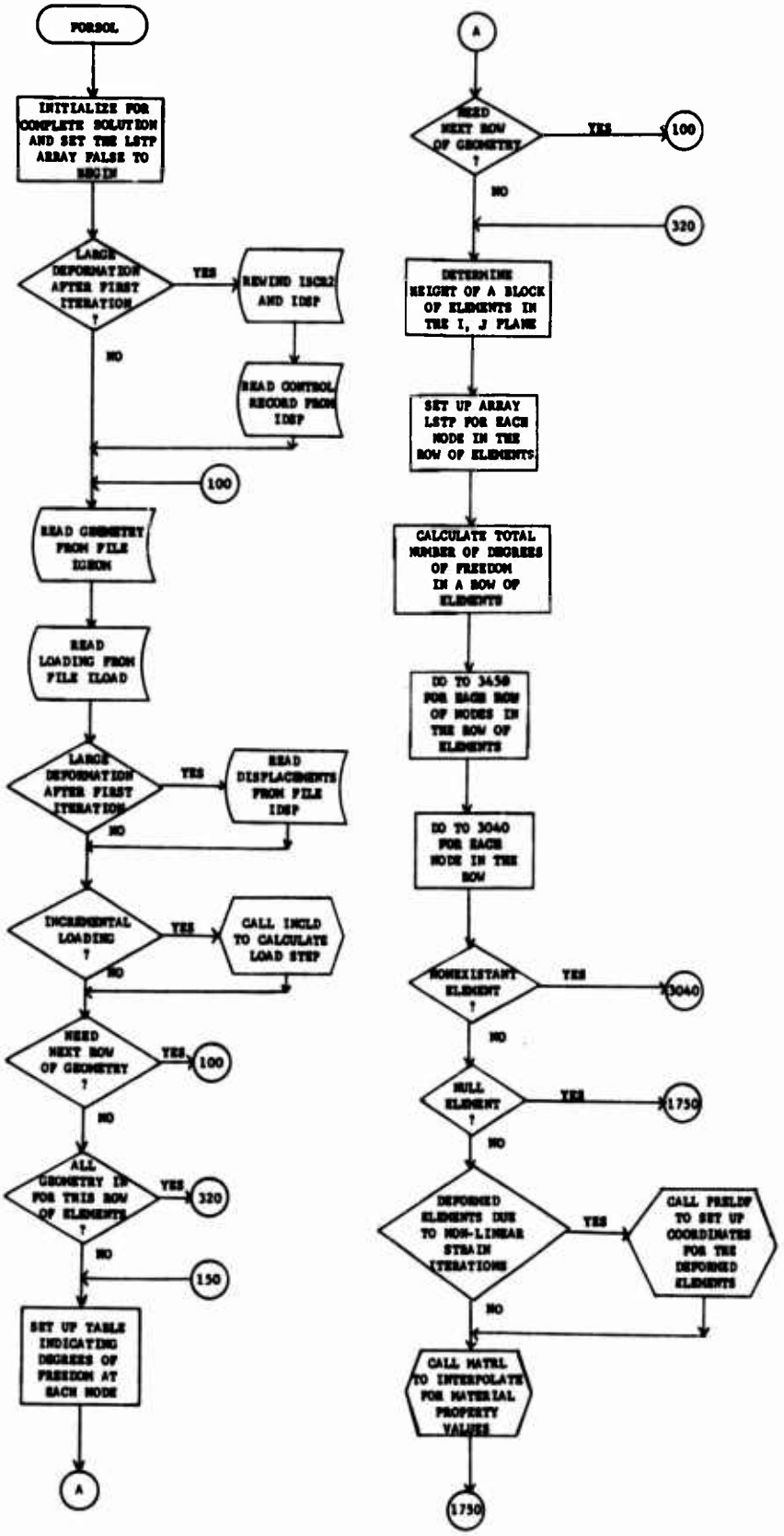
Following are flow charts of the basic program modules. Subroutine CONTRL is the main control routine and calls the other module drivers. Flow charts of the module drivers and main logic routines follow in order.

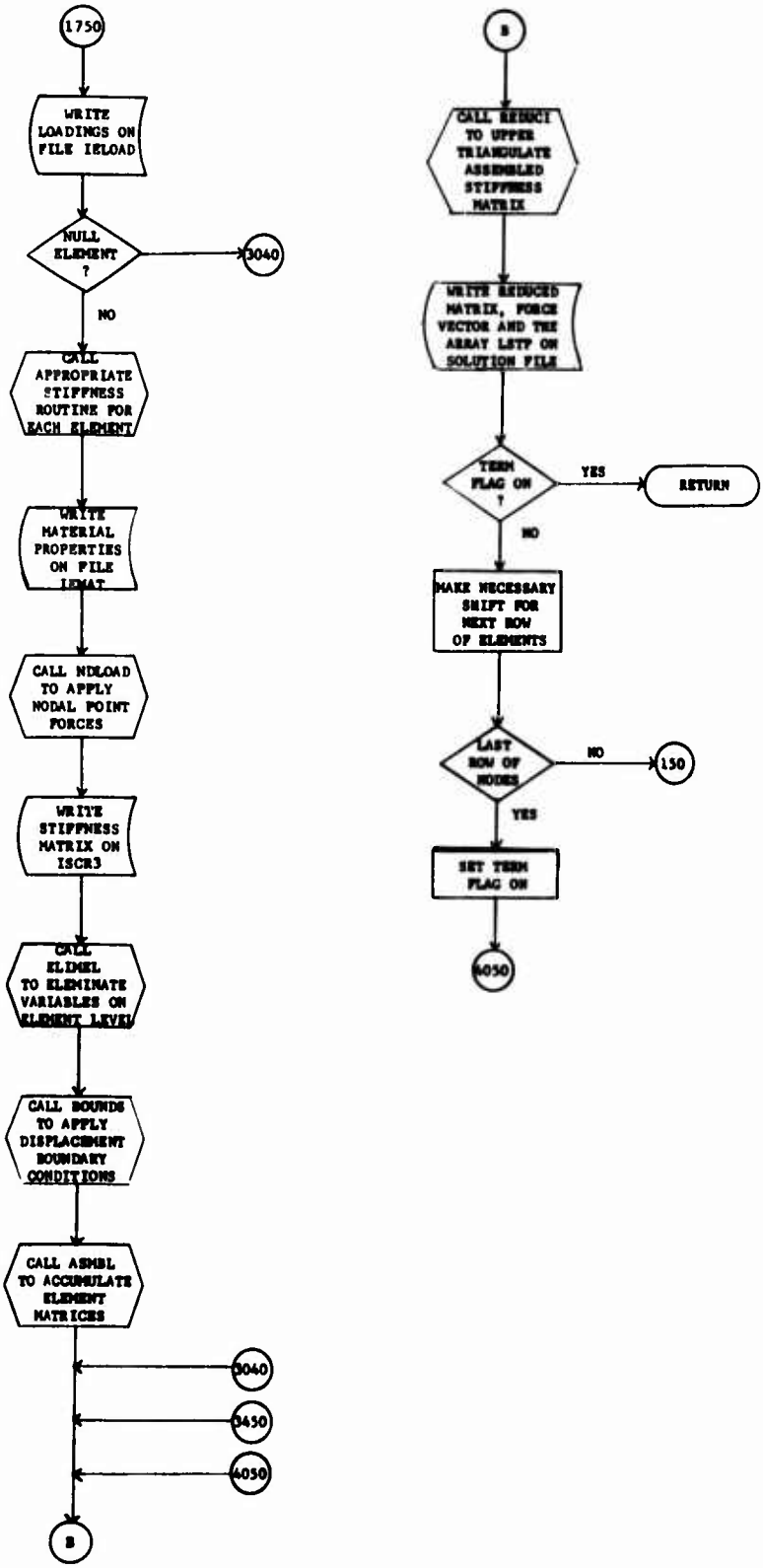

CONTRL
CALL TASSIO
CALL TDOPN
200
INITIALIZE
300
CALL PREPRN
E.O.F. ?
YES
RETURN
NO
FLAG RECORD ?
NO
300
YES
350
CALL PREPRN
E.O.F.
YES
ERROR INPUT IN BAD SHAPE
NO
RETURN
FLAG RECORD ?
NO
350
YES
GENERAL INPUT ?
NO
ERROR NO GENERAL DATA FLAG
YES
RETURN
CALL GENE
NORMAL RETURN ?
NO
RETURN
YES
540
CALL PREPRN
A
A
E.O.F. ?
YES
ERROR INPUT IN BAD SHAPE
NO
RETURN
FLAG RECORD ?
NO
540
YES
CALCULATE DIMENSIONS
CALL INMOD
TERMINAL ERROR ?
YES
RETURN
NO
STABILITY CALCULATIONS ?
YES
CALL STAMOD
200
NO
MESH ONLY ?
YES
200
NO
TRY NEXT CASE ?
YES
200
NO
600
SOLUTION MODULE ?
YES
CALL SOLMOD
NO
TERMINAL ERROR ?
YES
RETURN
NO
TRY NEXT CASE ?
YES
200
NO
B

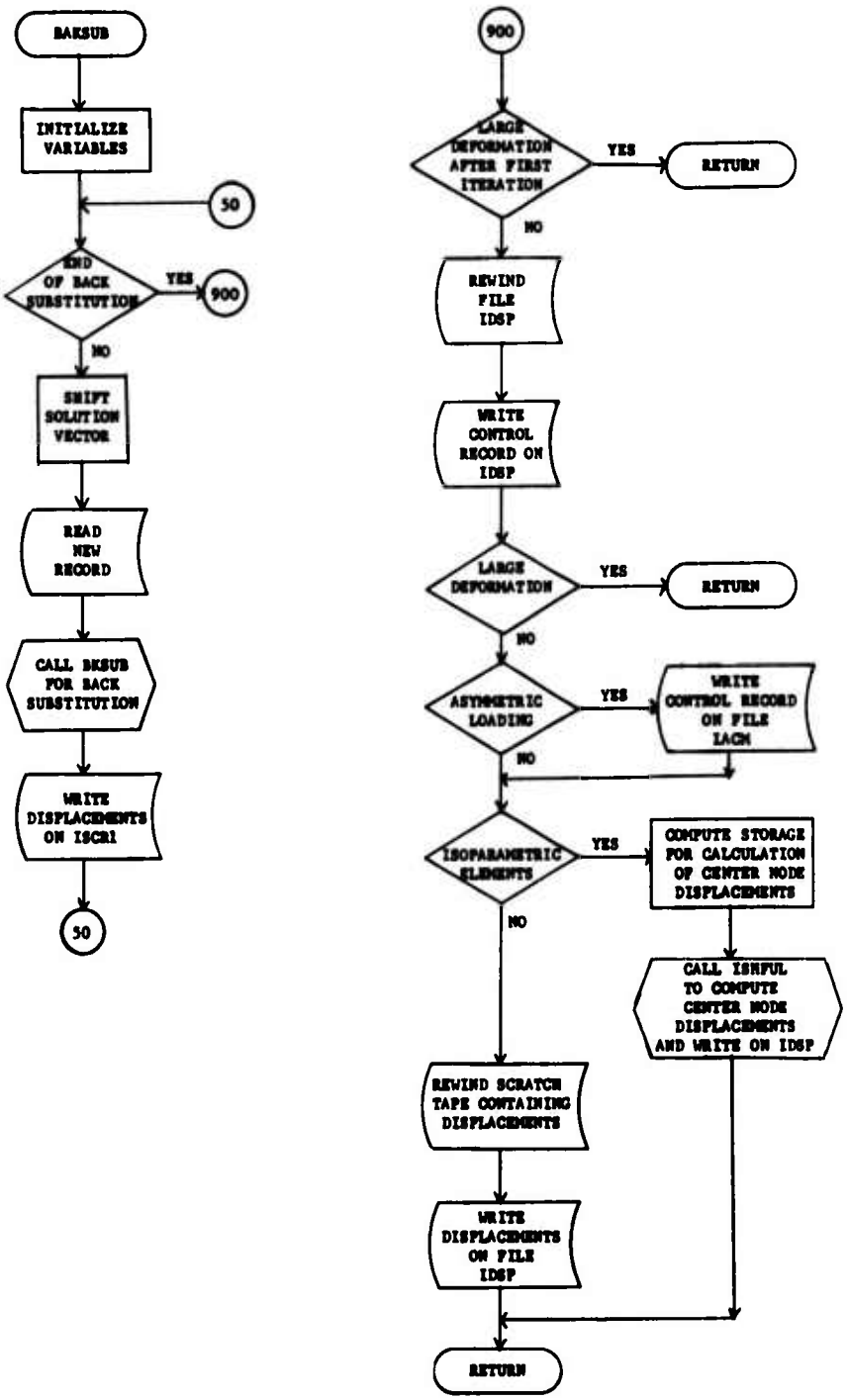

B
ITERATION
YES
600
NO
STRESS MODULE ?
YES
CALL STRMOD
NO
TERMINAL ERROR ?
YES
RETURN
NO
TRY NEXT CASE ?
YES
200
NO
PRINT MODULE ?
YES
CALL PRTMOD
NO
TERMINAL ERROR ?
YES
RETURN
NO
TRY NEXT CASE ?
YES
200
NO
LARGE DEFORMATION ?
YES
SET COUNTER TO ONE
NO
ITERATION ?
YES
600
NO
PLOT MODULE ?
YES
CALL PLTMOD
NO
TERMINAL ERROR ?
YES
RETURN
NO
200

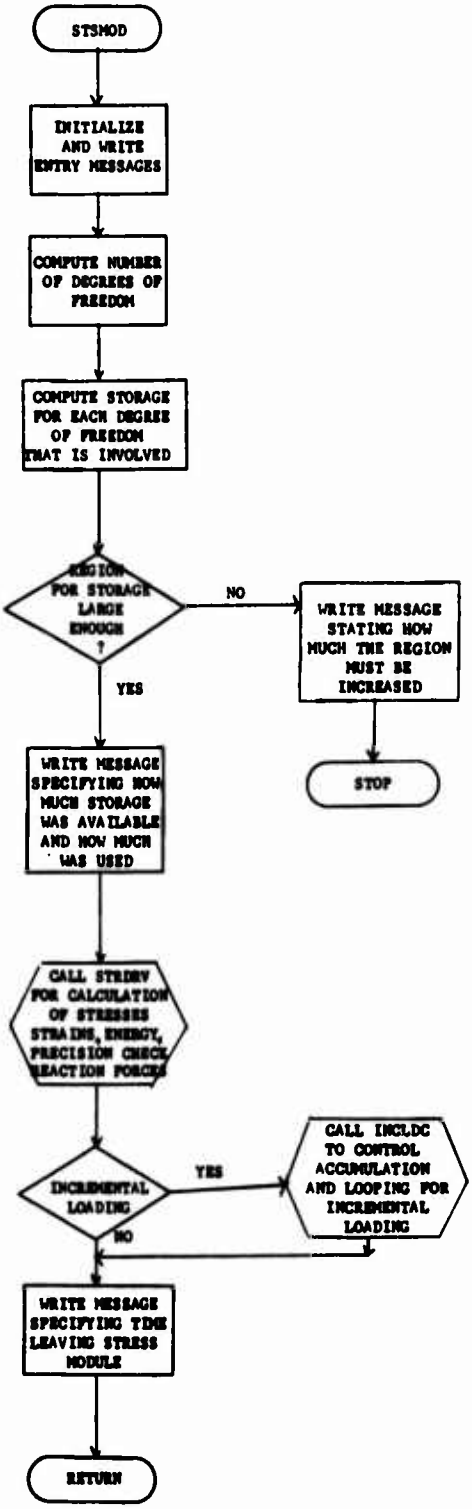

INMOD
INITIALIZE AND WRITE ENTRY MESSAGES
CALCULATE VARIABLE DIMENSIONS
REGION EXCEEDED
YES
WRITE MESSAGE FOR INSUFFICIENT SPACE
RETURN
NO
CALL INPUT
WRITE CORE THAT WAS AVAILABLE AND USED AND EXCESS CORE PRESENTLY AVAILABLE
WRITE CPU AND WAIT TIME
RETURN
INPUT
INITIALIZE VARIABLES
CALCULATE WORKING STORAGE REQUIRED
WORKING STORAGE FOR ROUTINE INPUT EXCEEDED
YES
WRITE MESSAGE
RETURN
NO
INITIALIZE FLIP/FLOP FILES FOR LARGE INPUT ARRAYS
250
CHECK IF VALID FLAG AND SET CODE
VALID FLAG
YES
400
NO
READ TO NEXT FLAG OR END OF FILE
FLAG
YES
250
NO
WRITE ERROR MESSAGE
RETURN
400
END OF DATA FLAG
YES
8500
NO
CALL INPUT ROUTINES DEPENDING ON CODE
END OF FILE
NO
250
YES
8500
CHECK CONSISTENCY OF DATA TYPES AND WRITE ERROR MESSAGES
CALL PROGRAM TO TEST AND OUTPUT GEOMETRY
RETURN

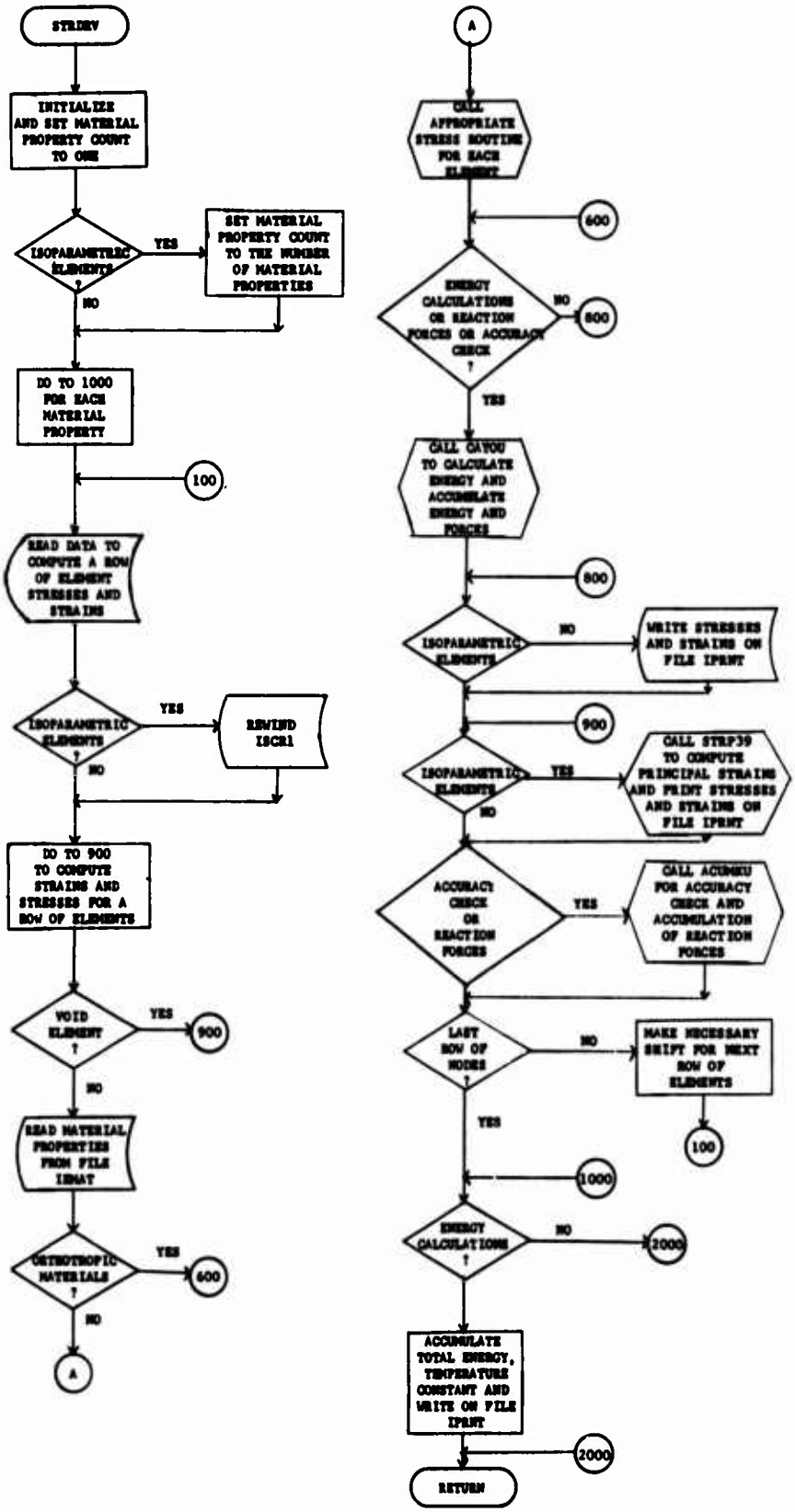

SOLMOD
INITIALIZE AND WRITE ENTRY MESSAGES
TURN STORAGE FLAG ON
50
COMPUTE STORAGE FOR THE ELIMINATION PHASE
STORAGE FLAG OFF ?
YES
CALL MATRD TO READ MATERIALS
NO
STORAGE FLAG OFF ?
YES
CALL FORSOL FOR FORWARD SOLUTION
NO
COMPUTE STORAGE FOR BACK SUBSTITUTION
STORAGE FLAG OFF ?
YES
CALL BAKSUB FOR BACK SUBSTITUTION
NO
LARGE DEFORMATION
NO
300
YES
A
A
COMPUTE STORAGE FOR NONLINEAR STRAIN ITERATIONS
STORAGE FLAG OFF ?
YES
CALL LDCTRL TO CONTROL NONLINEAR STRAIN ITERATIONS
NO
300
STORAGE FLAG OFF ?
YES
500
NO
STORAGE FOR SOLUTION MODULE EXCEEDED ?
NO
WRITE MESSAGE SPECIFYING CORE THAT WAS AVAILABLE AND WAS USED
TURN STORAGE FLAG OFF
50
YES
WRITE MESSAGE STATING NUMBER OF BYTES SHORT IN SOLUTION MODULE
500
WRITE MESSAGE SPECIFYING TIME LEAVING SOLUTION MODULE
RETURN

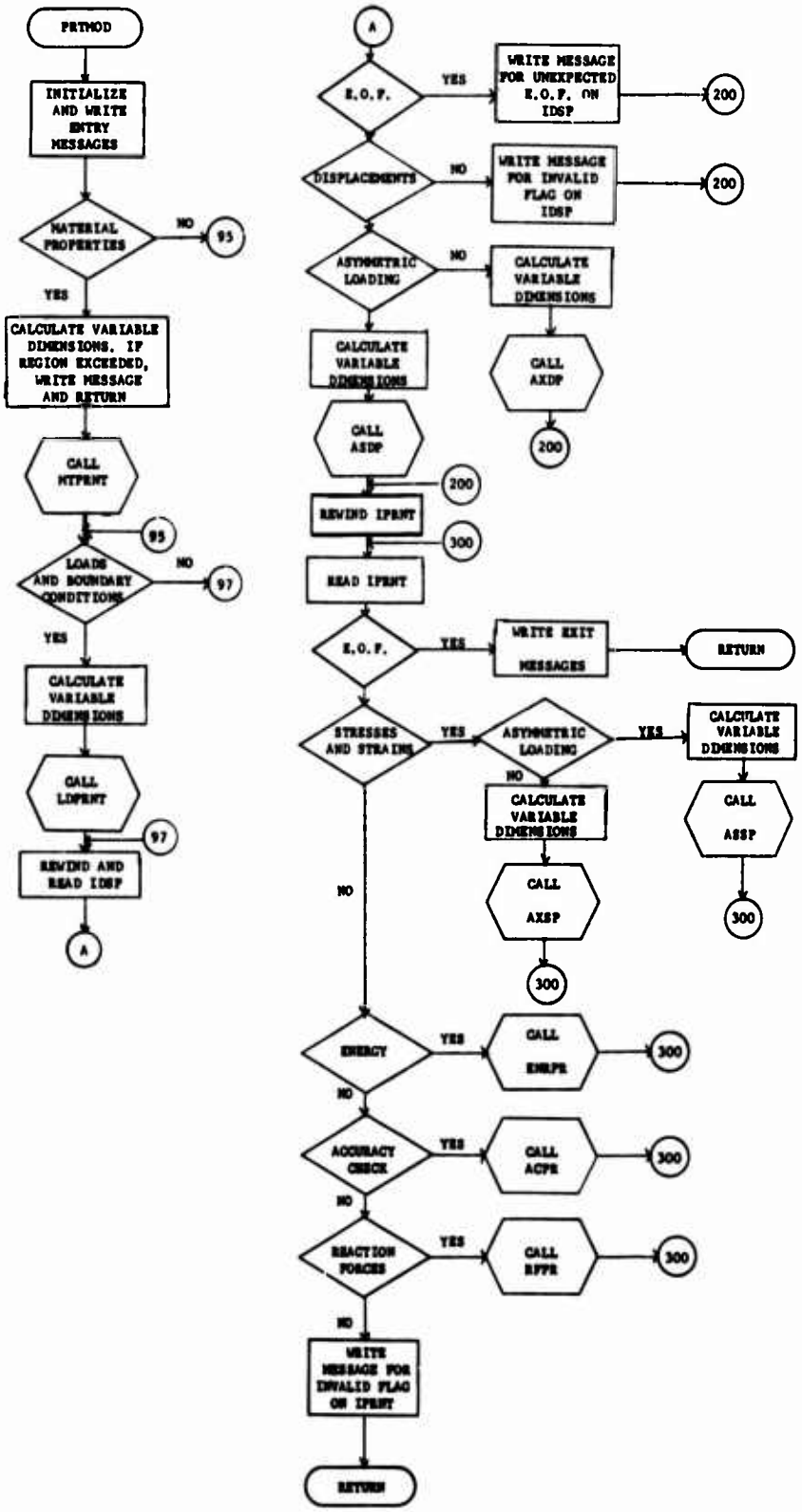

FORSOL
INITIALIZE FOR COMPLETE SOLUTION AND SET THE LSTP ARRAY FALSE TO BEGIN
LARGE DEFORMATION AFTER FIRST ITERATION ?
YES
REWIND ISCR2 AND IDSP
NO
READ CONTROL RECORD FROM IDSP
100
READ GEOMETRY FROM FILE IGEOM
READ LOADING FROM FILE ILOAD
LARGE DEFORMATION AFTER FIRST ITERATION
YES
READ DISPLACEMENTS FROM FILE IDSP
NO
INCREMENTAL LOADING ?
YES
CALL INCLD TO CALCULATE LOAD STEP
NO
NEED NEXT ROW OF GEOMETRY ?
YES
100
NO
ALL GEOMETRY IN FOR THIS ROW OF ELEMENTS ?
YES
320
NO
150
SET UP TABLE INDICATING DEGREES OF FREEDOM AT EACH NODE
A
A
NEED NEXT ROW OF GEOMETRY ?
YES
100
NO
320
DETERMINE HEIGHT OF A BLOCK OF ELEMENTS IN THE I, J PLANE
SET UP ARRAY LSTP FOR EACH NODE IN THE ROW OF ELEMENTS
CALCULATE TOTAL NUMBER OF DEGREES OF FREEDOM IN A ROW OF ELEMENTS
DO TO 3450 FOR EACH ROW OF NODES IN THE ROW OF ELEMENTS
DO TO 3040 FOR EACH NODE IN THE ROW
NONEXISTANT ELEMENT ?
YES
3040
NO
NULL ELEMENT ?
YES
1750
NO
DEFORMED ELEMENTS DUE TO NON-LINEAR STRAIN ITERATIONS
YES
CALL PRELDF TO SET UP COORDINATES FOR THE DEFORMED ELEMENTS
NO
CALL MATRL TO INTERPOLATE FOR MATERIAL PROPERTY VALUES
1750

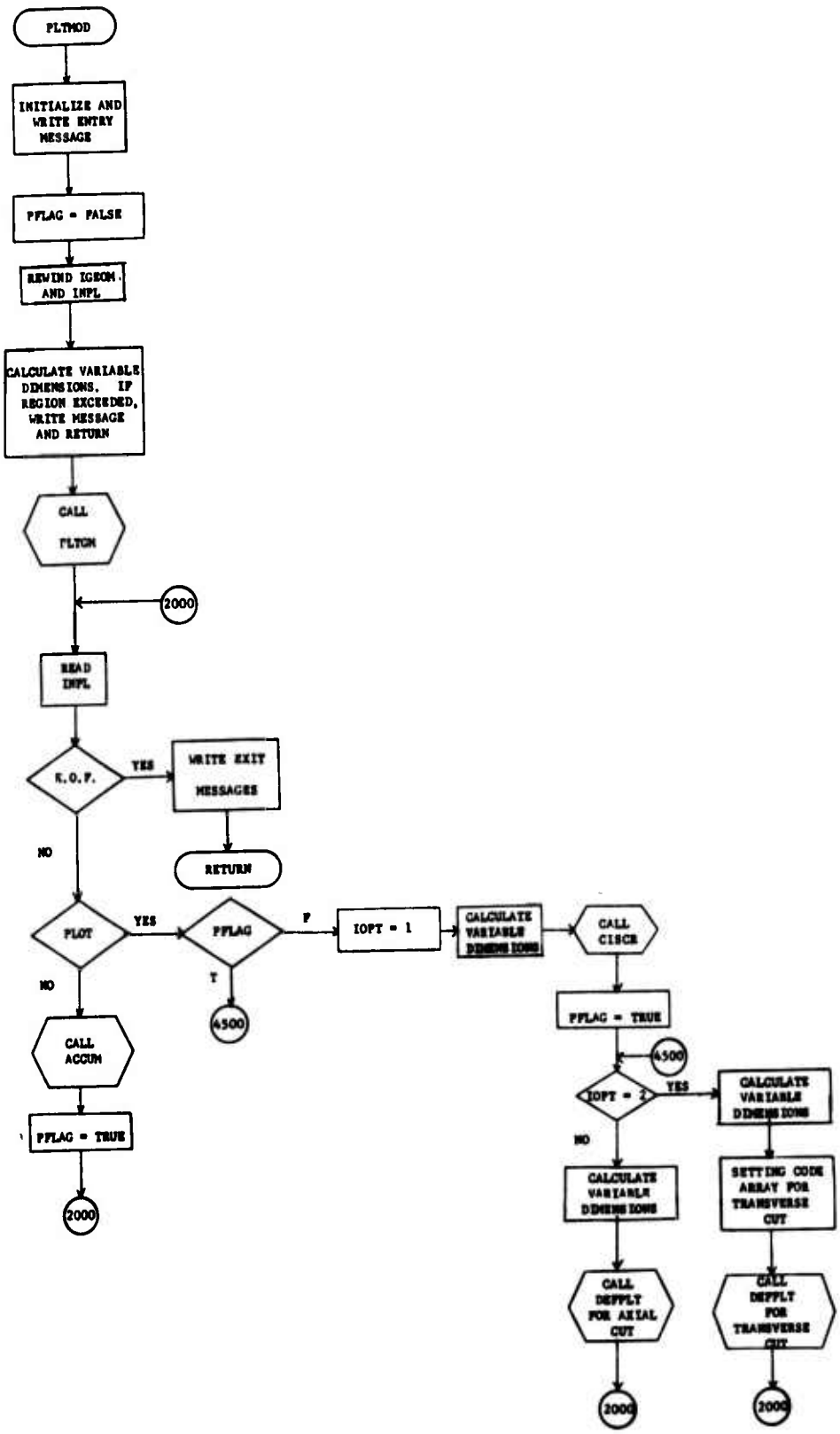

1750
WRITE LOADINGS ON FILE ISLOAD
NULL ELEMENT ?
3040
NO
CALL APPROPRIATE STIFFNESS ROUTINE FOR EACH ELEMENT
WRITE MATERIAL PROPERTIES ON FILE IEMAT
CALL NDLOAD TO APPLY NODAL POINT FORCES
WRITE STIFFNESS MATRIX ON ISCR3
CALL ELIMEL TO ELEMINATE VARIABLES ON ELEMENT LEVEL
CALL BOUNDS TO APPLY DISPLACEMENT BOUNDARY CONDITIONS
CALL ASMBL TO ACCUMULATE ELEMENT MATRICES
3040
3450
4050
B
B
CALL REDUCI TO UPPER TRIANGULATE ASSEMBLED STIFFNESS MATRIX
WRITE REDUCED MATRIX, FORCE VECTOR AND THE ARRAY LSTP ON SOLUTION FILE
TERM FLAG ON ?
YES
RETURN
NO
MAKE NECESSARY SHIFT FOR NEXT ROW OF ELEMENTS
LAST ROW OF NODES
NO
150
YES
SET TERM FLAG ON
4050


BAKSUB
INITIALIZE VARIABLES
50
END OF BACK SUBSTITUTION
YES
900
NO
SHIFT SOLUTION VECTOR
READ NEW RECORD
CALL BKSUB FOR BACK SUBSTITUTION
WRITE DISPLACEMENTS ON ISCR1
50
900
LARGE DEFORMATION AFTER FIRST ITERATION
YES
RETURN
NO
REWIND FILE IDSP
WRITE CONTROL RECORD ON IDSP
LARGE DEFORMATION
YES
RETURN
NO
ASYMMETRIC LOADING
YES
WRITE CONTROL RECORD ON FILE LAGN
NO
ISOPARAMETRIC ELEMENTS
YES
COMPUTE STORAGE FOR CALCULATION OF CENTER NODE DISPLACEMENTS
NO
CALL ISHFUL TO COMPUTE CENTER NODE DISPLACEMENTS AND WRITE ON IDSP
REWIND SCRATCH TAPE CONTAINING DISPLACEMENTS
WRITE DISPLACEMENTS ON FILE IDSP
RETURN


STSMOD
INITIALIZE AND WRITE ENTRY MESSAGES
COMPUTE NUMBER OF DEGREES OF FREEDOM
COMPUTE STORAGE FOR EACH DEGREE OF FREEDOM THAT IS INVOLVED
REGION FOR STORAGE LARGE ENOUGH ?
NO
WRITE MESSAGE STATING HOW MUCH THE REGION MUST BE INCREASED
STOP
YES
WRITE MESSAGE SPECIFYING HOW MUCH STORAGE WAS AVAILABLE AND HOW MUCH WAS USED
CALL STRDRV FOR CALCULATION OF STRESSES STRAINS, ENERGY, PRECISION CHECK, REACTION FORCES
INCREMENTAL LOADING
YES
CALL INCLDC TO CONTROL ACCUMULATION AND LOOPING FOR INCREMENTAL LOADING
NO
WRITE MESSAGE SPECIFYING TIME LEAVING STRESS MODULE
RETURN


STRDRV
INITIALIZE AND SET MATERIAL PROPERTY COUNT TO ONE
ISOPARAMETRIC ELEMENTS ?
YES
SET MATERIAL PROPERTY COUNT TO THE NUMBER OF MATERIAL PROPERTIES
NO
DO TO 1000 FOR EACH MATERIAL PROPERTY
100
READ DATA TO COMPUTE A ROW OF ELEMENT STRESSES AND STRAINS
ISOPARAMETRIC ELEMENTS ?
YES
REWIND ISCR1
NO
DO TO 900 TO COMPUTE STRAINS AND STRESSES FOR A ROW OF ELEMENTS
VOID ELEMENT ?
YES
900
NO
READ MATERIAL PROPERTIES FROM FILE IMAT
ORTHOTROPIC MATERIALS ?
YES
600
NO
A
A
CALL APPROPRIATE STRESS ROUTINE FOR EACH ELEMENT
600
ENERGY CALCULATIONS OR REACTION FORCES OR ACCURACY CHECK ?
NO
800
YES
CALL GAYOU TO CALCULATE ENERGY AND ACCUMULATE ENERGY AND FORCES
800
ISOPARAMETRIC ELEMENTS
NO
WRITE STRESSES AND STRAINS ON FILE IPRNT
900
ISOPARAMETRIC ELEMENTS
YES
CALL STRP39 TO COMPUTE PRINCIPAL STRAINS AND PRINT STRESSES AND STRAINS ON FILE IPRNT
NO
ACCURACY CHECK OR REACTION FORCES
YES
CALL ACUMKU FOR ACCURACY CHECK AND ACCUMULATION OF REACTION FORCES
LAST ROW OF NODES ?
NO
MAKE NECESSARY SHIFT FOR NEXT ROW OF ELEMENTS
100
YES
1000
ENERGY CALCULATIONS
NO
2000
ACCUMULATE TOTAL ENERGY, TEMPERATURE CONSTANT AND WRITE ON FILE IPRNT
2000
RETURN


PRTMOD
INITIALIZE AND WRITE ENTRY MESSAGES
MATERIAL PROPERTIES
NO
95
YES
CALCULATE VARIABLE DIMENSIONS. IF REGION EXCEEDED, WRITE MESSAGE AND RETURN
CALL MTPRNT
95
LOADS AND BOUNDARY CONDITIONS
NO
97
YES
CALCULATE VARIABLE DIMENSIONS
CALL LDPRNT
97
REWIND AND READ IDSP
A
A
E.O.F.
YES
WRITE MESSAGE FOR UNEXPECTED E.O.F. ON IDSP
200
DISPLACEMENTS
NO
WRITE MESSAGE FOR INVALID FLAG ON IDSP
200
ASYMMETRIC LOADING
NO
CALCULATE VARIABLE DIMENSIONS
CALL AXDP
200
CALCULATE VARIABLE DIMENSIONS
CALL ASDP
200
REWIND IPRNT
300
READ IPRNT
E.O.F.
YES
WRITE EXIT MESSAGES
RETURN
STRESSES AND STRAINS
YES
ASYMMETRIC LOADING
YES
CALCULATE VARIABLE DIMENSIONS
CALL ASSP
300
NO
CALCULATE VARIABLE DIMENSIONS
CALL AXSP
300
NO
ENERGY
YES
CALL ENEPR
300
NO
ACCURACY CHECK
YES
CALL ACPR
300
NO
REACTION FORCES
YES
CALL RFPR
300
NO
WRITE MESSAGE FOR INVALID FLAG ON IPRNT
RETURN


PLTMOD
INITIALIZE AND WRITE ENTRY MESSAGE
PFLAG = FALSE
REWIND IGEOM AND INPL
CALCULATE VARIABLE DIMENSIONS. IF REGION EXCEEDED, WRITE MESSAGE AND RETURN
CALL PLTGN
2000
READ INPL
E.O.F.
YES
WRITE EXIT MESSAGES
RETURN
NO
PLOT
YES
PFLAG
F
IOPT = 1
CALCULATE VARIABLE DIMENSIONS
CALL CISCR
T
4500
NO
CALL ACCUN
PFLAG = TRUE
2000
PFLAG = TRUE
4500
IOPT = 2
YES
CALCULATE VARIABLE DIMENSIONS
NO
SETTING CODE ARRAY FOR TRANSVERSE CUT
CALCULATE VARIABLE DIMENSIONS
CALL DEFFLT FOR AXIAL CUT
CALL DEFFLT FOR TRANSVERSE CUT
2000
2000

SPECIAL ROUTINES

There are several assembler language routines in the program which require some special attention. These are detailed below:

Subroutine TDRR

The purpose of this routine is to provide a means to write large records on disk and then retrieve them in any order.

The routine is written in assembler language and uses BSAM reads and writes with note and point to accomplish its I/O.

When the open entry is called, a getmain is issued to get core for a table. This table will be used to hold the beginning address of each logical record. The index passed with each read or write is used to reference this table either to store the track address for a write or to retrieve it for a read.

Logical records may be any size, and the routine will break them up into blocks if necessary. The track address of the first block in any one record is all that will be saved.

The maximum physical block size is coded into the routine as the variable LIM and must always reflect the track size of the disk drive being used. No buffers are used since the routine reads and writes directly from core.

There are three entries into the routine:

CALL TDOPN ('FTXX', CODE, MAX)

This entry must be called to initialize activity on the data set. The XX in FTXX must be replaced with the two digits from the DDNAME for the data set. Code must be 1 at present. MAX is the maximum number of logical records to be written. CODE and MAX are both integers.

CALL TDWR (REC, K, L)

This entry will write a logical record from array REC for K contiguous words. L is the index number for this record; it must be unique, between 1 and MAX, and used to request the record when it is to be read. K and L are integers, while REC may be of any type.

CALL TDRD (REC, K, L)

This entry will cause the record assigned index number L in the write command to be read into array REC for K words.

This routine will handle only one data set at present.

Subroutine TASSIO

The purpose of this routine is to rapidly read and write data from scratch data sets in binary form. The routine is written in assembler language and uses bsam reads and writes to accomplish its I/O. The records are blocked into physical records of 800 bytes by the routine, and a control word is attached to each record, giving the length of both the preceding and the following records. This control word is used to check record lengths on reads to verify the data, and to control the backspacing of logical records.

The maximum logical record allowed is 32767 words. Records will automatically be spanned over several physical records where necessary.

A maximum of 50 files with FT numbers 1 through 50 may be specified, and any number may be in use at one time. A getmain is issued for 1744 bytes for each open file, and the area thus obtained contains the DCB, DECB and buffers along with the buffer control parameters. Either disk or tape may be used.

The routine has six entry points. In the following descriptions, file, code and count are integers while the arrays may be any type.

CALL TASSIO (FILE, CODE)

This entry must be called to initiate activity on each file. File is the unit number and must be from 1 to 50. Code is 1 if the file is to be used for output or 2 if 1 is to be opened for input.

CALL TASWR (FILE, ARRAY1, COUNT1,...,ARRAYN, COUNTN)

This entry will write as many logical records as there are pairs of arrays and counts. File is the unit number. Array is the area to be written from. Count is the number of 4-byte words to be written. Count must be no larger than 32767.

CALL TASRD (FILE, IRTN, ARRAY1, COUNT1,...,ARRAYN, COUNTN)

This entry will read as many logical records as there are pairs of arrays and counts. File is the unit number. Array is the area to be read into. Count is the number of 4-byte words to be read. Input a -1 in IRTN to have TASSIO dump on an end of file. If IRTN is 0 on entry, it will be left 0 on a normal return and will be set to +1 if an end of file is encountered.

CALL TASBCK (FILE, IRTN, ARRAY1, COUNT1,...,ARRAYN, COUNTN)

This entry will read as many logical records as there are pairs of arrays and counts. The records will be retrieved in inverse order to the way they were written. This accomplishes the same thing as a backspace - read - backspace in FORTRAN. All parameters are the same as for TASRD.

CALL TASREW (FILE, CODE)

This entry rewinds the file and positions it as specified by CODE. If CODE is 1, the file is opened for output. If CODE is 2, the file is opened for input and the first block is read and checked and the second block has a read issued on it.

CALL TASCLS (FILE, CODE)

This entry closes the file as specified by CODE. If CODE is 1, the file will be rewound and all control and buffer space will be freed; TASSIO must be called for this file before it can be used again. If CODE is 2, the file is completed but left open and positioned to the end so that it can be read using TASBCK. TASCLS must be called with a CODE of 2 before TASBCK is called.

Subroutine GETMAIN

This routine does a conditional getmain for a very large section of core. Then when the system returns and tells what is available, it frees the whole thing, reduces the amount available by a specified amount, and does a second getmain for the reduced region. This allows us to get all available core in our region and still leave the system what it needs.

Subroutine FREFRM

This routine is designed to read cards in a free-field format. The input specifications are given in the user's manual.

Two things should be mentioned here that are not given elsewhere.

First, the routine uses what is referred to as a control string. There is one special control string built into the program in the material property input routine.

A control string may be used to simplify the input of a sequence of logical records. Each digit in the control string represents a parameter in the logical record and has the following values:

0 - no input for this parameter.

1 - this parameter will be input in each record.

2 - input will be made in the first record and will be considered constant thereafter.

For example, if a ten value record is available and the 2nd value is constant, the 4th and 6th vary with each record, and no other values will be input; a control string of C0201010000 may be set up. The first record would then contain 3 values and each following only 2.

Second, the routine begins scanning on a record at the end of the previous record, not at the beginning of a card. The difference then between title and flag records is that title records are treated as cards and columns 1-72 are moved directly into the flag area, while all leading blanks are stripped from a flag record. This insures that the first word in the flag area can be compared to a sequence of valid flags.

Two FORTRAN routines are used by this routine: FRERD to read file FT05 and FRERR to put error messages on FT06.

Subroutine DATIM

This routine is used to get the date as a Julian integer and the time in centiseconds. It does this by issuing a TIME macro in assembler language.

Subroutine DATE

This routine gives the calendar date as an alpha string.

Subroutine TYME

This routine gives the time of day in hours, minutes and seconds as an alpha string.

Subroutine XFPTRP

Literally X-floating-point-trap. This routine traps certain floating point errors and takes corrective action. When an underflow occurs, no error message is given; the result is simply set to zero and execution continues.

When a division by zero occurs, the result is set to zero and execution continues. When an overflow condition occurs, the program terminates with a core dump and a user completion code of 444.

This routine calls a FORTRAN routine CTNPWT to write its error messages.

Subroutine ELPTIM

This routine obtains the CPU and wait time from the system. Both times are returned as integers in hundredths of seconds.

Subroutine BLKLD

This routine is used to initialize a block of core to a given value. The routine goes from one address to, but not including, a final address.

Subroutine ABORTA

This routine creates an abend with a core dump and user completion code of 3276 anytime it is called.

DATA FILES

There are several basic data files whose format will be defined in the following paragraphs. Not all files will be defined here since several are scratch files or their format and usage are relatively obvious from the program coding.

The Geometry File

This has the name IGEOM. It contains geometry and flag data row-wise for use throughout the program. It is created in subroutine PRGEOM.

The record set for each row of nodes is as follows:

1. The J index of the row.
2. The minimum I index for the row.
3. The maximum I index for the row.
4. All of the R coordinates for the row.
5. All of the Z coordinates for the row.
6. All of the ICODE data for the row. For each node this consists of 16 half-words whose usage is:

    1 Print flag

    2 Element type

    3 Material code

    4 R boundary condition flag

    5 Z boundary condition flag

    6 Not used at present

    7 θ boundary condition flag

    8 Sliding boundary condition flag

    9 Not used at present

    10 Not used at present

11 Isoparametric order of integration

12-16 Not used at present

The Load File

This file has the name ILOAD. It contains load and boundary condition data row-wise. It is created in subroutine PRGEOM.

The record set for each row of nodes is as follows:

1. The J index of the row

2. The load array. For each node this consists of 38 double words as follows:

1 Temperature of the element

2-5 Pressure on faces 1-4 in that order

6-9 Shear on faces 1-4

10-13 Radial traction (PR) on faces 1-4

14-17 Theta traction (PTH) on faces 1-4

18-21 Axial traction (PZ) on faces 1-4

22 Radial body force

23 Theta body force

24 Axial body force

25 Not used at present

26 Radial boundary condition value

27 Axial boundary condition value

28 Not used at present

29 Theta boundary condition value

30 Sliding boundary condition value

31-38 Not used at present

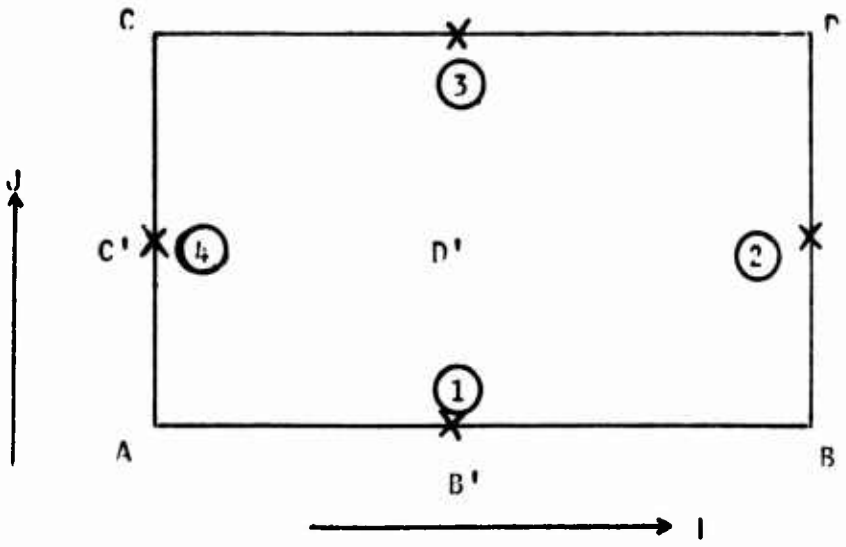


Figure 1. Pressure and Shear Conventions

Figure 1 shows the face designations for pressure and shear loads. The numbers in circles give the face codes. Note the direction of increasing I and J.

If we have an isoparametric element with extra nodes indicated by the X's in Figure 1, the loads will be applied at the corners rather than on the faces. The loads for node A are in the first and fourth load locations of the data for the linear element A. The loads for node B are in the first and second load locations for linear element B'. Thus the location in the sequence gives the direction of the load, the first location being in the direction normal to face one, etc.

The Material File

This file has the name MATFL. It contains the material property tables. It is written in subroutine MATP and read in subroutine MATRD.

The form of the table is as follows for each material:

1. A material header record consisting of 5 double words as follows:

1 The material number as a floating point number.

2 The number of records (N) in this material table as a floating point number.

3-5 A 24-character material name.

2. The material property detail record consisting of 15 double words. There will be N of these for each material. Note that this is an orthotropic record.

1-3 The three values of Young's modulus. The isotropic modulus is in location 1.

4-9 The six values of Poisson's ratio. The isotropic ratio is in location 4.

10 The orthotropic shear modulus.

11-13 The three values of alpha. The isotropic alpha is in location 11.

14 Temperature if temperature dependent; otherwise zero.

15 Strain if strain dependent; otherwise zero.

The Displacement File

This file has the name IDSP. It contains the displacements as written during the backsubstitution.

The file begins with a flag record of the following form:

1. The number 1, indicating displacements.

2. The mode if asymmetric loading.

3. A bit flag defining the degrees of freedom in the problem.

One record of the following form is written for each row of nodes.

1. The J index of the row.

2. The displacements for the row as a series of arrays U, V, W, ROT & H, any of which may be zero length. For example, for the axisymmetric problem, only U and W actually exist.

The Print File

This file has the name IPRNT. It contains stresses and strains, energy, accuracy check and reaction forces to be printed by the print module. Any or all of the groups may be present depending on the requested options. Each group is layed out as follows.

The stresses and strains are preceded by a flag record:

1. The number 2, indicating stresses.

2. The mode number if asymmetric.

3. A bit flag giving the degrees of freedom.

The flag is followed by one record of the following form for each set of stresses and strains to be printed:

1. The I index

2. The J index

3. A record containing the values. This varies in length and content from element to element. See the respective stress or print routines for the detail for a given element.

The energy is preceded by a flag record similar to the one for stresses except that the flag for energy is the number 3. The flag is followed by one record of the form:

1. The mechanical energy term.

2. The temperature constant.

3. The sum of the two which is the total energy.

The accuracy check data is preceded by a flag record similar to the one for stresses except that the flag is the number 4. The flag is followed by ten records of the form:

1. An I index

2. A J index

3. A component code

4. The value of KU for the component and node

5. The value of F

6. The difference KU-F

The reaction force data is preceded by a flag record similar to the one for stresses except that the flag is the number 5. The flag is followed by one record for each node with boundary conditions on it. The last record is followed by a flag record of all hexadecimal 01234567 words to mark the end. The logical record is of the form:

1. The I index

2. The J index

3. Three double words giving the reaction forces in the coordinate directions:

   Radial

   Theta

   Axial

The Service Life File

This file has the name ISLF. It is written using FORTRAN binary I/O. It is usually a tape and contains data needed for the service life calculations. Several sets of data may be stacked together on the file. Each set of data has the following form.

A flag record giving:

1. The date as a Julian date in a one-word integer.

2. The time of day in centiseconds as a one-word integer.

3. The 18-word title of the run producing the data.

This is followed by one record of the following form for each element:

1. The I index

2. The J index

3. The thermal load

4. The radial strain

5. The theta strain

6. The axial strain

7. The RZ shearing strain

8. The Rθ shearing strain

9. The Zθ shearing strain

10. The radial stress

11. The theta stress

12. The axial stress

13. The RZ shearing stress

14. The Rθ shearing stress

15. The Zθ shearing stress

The last record is followed by a flag record which indicates the end of that set. It is composed of 28 words of a hexidecimal 01234567.

COMMONS

Common /$BEAR/

Usage: This common stores the bearing geometry to be passed to the subroutine BGRID.

Modules: Inmod

Parameters

name	Type	Usage
IMIN	I	Minimum I value in the bearing.
IMAX	I	Maximum I value in the bearing.
JMIN	I	Minimum J value in the bearing.
JMAX	I	Maximum J value in the bearing.
IVEC	I	Vector flag
ID11	I	Dummy for word alignment
VANG	R*8	Vector angle (degrees)
ITF	I	Not used at present
IEQU	I	If this is nonzero, the nodes will be equally spaced across the bearing in the I direction instead of having the first and last interval half the others.
ID1	I	Not used at present
ISC	I	+0 spherical bearing +1 Conical bearing
RI	R*8	Inside radius of bearing
RF	R*8	Outside radius of bearing
RII	R*8	Radial coordinate of a corner node

RIF	R*8	Radial coordinate of a corner node
RFI	R*8	Radial coordinate of a corner node
RFF	R*8	Radial coordinate of a corner node
ZII	R*8	Axial coordinate of a corner node
ZIF	R*8	Axial coordinate of a corner node
ZFI	R*8	Axial coordinate of a corner node
ZFF	R*8	Axial coordinate of a corner node
TS	R*8	Thickness of shim
NS	I	Number of columns in one shim
ISN	I	Number of shims
TE	R*8	Thickness of elastomer
NE	I	Number of columns in one elastomer
IEN	I	Number of elastomers
ID2(7)	I	Not used at present
MS	I	Material number for shim
ME	I	Material number for elastomer
ID5(19)	I	Not used at present
SHIMT(50)	R*8	Shim thickness table
ELAST(50)	R*8	Elastomer thickness table
MATS(50)	I	Shim material tables
MATE(50)	I	Elastomer material tables

COMMON /$BIND/

Usage: This common contains the indices which define the region to be filled in by the subroutine BEARNG.

Modules: Inmod

Parameters:

Name	Type	Usage
IB	I	Beginning I index
IE	I	Ending I index
JB	I	Beginning J index
JE	I	Ending J index

COMMON /ACCOM/

Usage: This common provides information necessary to accumulate and print displacements, stresses and strains for an asymmetrically loaded body.

Modules: Pltmod

Parameters:

Name	Type	Usage
TORZ	R*8	Theta or Z cut location
NN	I	Maximum number of node points in radial direction on transverse cut.
MODE	I	Mode number in the Fourier series.
FLAG	L	First time flag for data read in accumulation.
IDS	I	ID-1
JDS	I	JD-1

KK	I	Accumulation record counter
IOPT	I	Option flag for axial or transverse cut
KD	I	Actual number of node points in radial direction on transverse cut

COMMON /ASYM/

Usage: This common stores data necessary for the asymmetric loading option.

Modules: Driver, Solmod, Stsmod

Parameters:

Name	Type	Usage
MODE	I	Mode number
LMOD	I	Highest mode allowed in the problem

COMMON /CNTRL/

Usage: The general data is stored in this common.

Modules: Inmod, Driver, Pltmod, Prtmod, Solmod, Stsmod
Parameters:

Name	Type	Usage
TITLE(18)	R*4	Title for program
LRUN(20)	L	Run parameter flags (in same order as run parameters in input document)
LSOL	L	Logical control
LSTRS	L	parameters to
LPRNT	L	specify which
LPLOT	L	module to enter

LOOP1	L	Iteration control logical
LOOP2	L	Iteration control logical
LDUM(5)	L	Not used at present
IMIN	I	Minimum I
JMIN	I	Minimum J
IMAX	I	Maximum I
JMAX	I	Maximum J
NTTPS	I	Not used at present
NREGN	I	Not used at present
JSHEL	I	Not used at present
AXSTRA	R*8	Not used at present
AXSTRS	R*8	Not used at present
BTEMP	R*8	Base temperature

COMMON /CORMAX/

Usage: This common stores necessary information to compute the amount of core used in a particular module.

Modules: Inmod, Pltmod

Parameters:

Name	Type	Usage
KMAX	I	The maximum number of words allocated from the array A and used in one of the subroutines of the module
KJL	I	Number of words allocated from the array A and used in the driver of the module

COMMON /DIMEN/

Usage: This common stores dimensions necessary to run the program.

Modules: Inmod, Driver, Pltmod, Prtmod, Solmod, Stsmod

Parameters:

Name	Type	Usage
ID	I	The I dimension of the problem
JD	I	The J dimension of the problem
ND	I	Number of rows of geometry necessary to completely define a row of elements
MD	I	Number of single precision words required for material property tables
LL	L	Words remaining in array A after words have been allocated to working routines

COMMON /DSPRC/

Usage: This common provides work space for those routines which accumulate or print displacements for an asymmetrically loaded body.

Modules: Pltmod

Parameters:

Name	Type	Usage
TH	R*4	Theta or Z coordinate
R	R*4	Radial coordinate
DR	R*4	R displacement
DT	R*4	Theta displacement
DZ	R*4	Z displacement

COMMON /ELPTR/

Usage: This common stores pointers, etc, to assemble the element stiffness matrix into the master stiffness matrix.

Modules: Solmod

Parameters:

Name	Type	Usage
NU(3,3)	I	U pointers
NW(3,3)	I	W pointers
NROT(3,3)	I	Rotation pointers
NV(3,3)	I	V pointers
NH(3,3)	I	H pointers
INDOF(3,3)	I	Degrees of freedom
MTXSIZ	I	Element matrix dimension
LIM	I	Number of degrees of freedom eliminated from element matrix

COMMON /FILES/

Usage: The common contains all file designations except that for the TDRR file, which uses FT01F001, and that for the service life file, which has its own common. All I/O is to use the symbolic names as given here.

Name	Ft. Number	Usage
IOUT	6	Printer
IPUN	7	Card punch
IAUD	8	Printer - audit file
INPL	12	Plot and accumulation control records
INEIS	13	Not used at present
MATFL	14	Material properties
IGEOM	10	Geometry file
ILOAD	11	Load file

IDSP	18	Displacement file
IPRNT	19	Print file for all but displacements
ISTFF	20	Stiffness data passed from stiffness to stress routines
IEMAT	21	Element material properties
ISCR1	22	Scratch file
ISCR2	23	Scratch file
ISCR3	24	Scratch file
ISCR4	30	Scratch file
ISCR5	31	Scratch file
ISCR6/ISD1	32	Scratch file also used in incremental loading option
ISCR7/ISD2	33	Scratch file also used in incremental loading option
ISCR8	34	Scratch file
ISCR9	35	Scratch file
IELOAD	25	Element loads
IACM	26	Displacements and strains for asymmetric accumulation

Note: Due to the fact that some pairs of file names are interchanged in the program, the names will not always have the value given here. Also, different forms of this common will be found in some areas of the program; the names may change, but the use of the files is as given here.

COMMON /INCNTL/

Usage: This common provides information about the correctness and the type of input being read by the subroutine FREFRM.

Modules: Inmod, Driver

Parameters:

Name	Type	Usage
IFLG(18)	I	Flag record
IERR	I	Error return code
ICNT	I	Return code which tells excess number of values read

COMMON /INCRLD/

Usage: This common stores the information necessary for the control of the incremental loading option.

Modules: Inmod, Driver, Solmod, Stsmod

Parameters:

Name	Type	Usage
ILKT	I	Counter
ILLIM	I	Total number of increments
ILPT	L	Print flag

The following 8 parameters are flags to indicate which loads are applied incrementally.

ILPR	L	Pressure
ILSH	L	Shear
ILTMP	L	Temperature
ILXBF	L	R body force
ILYBF	L	Z body force
ILXNL	L	R nodal load
ILYNL	L	Z nodal load
ILND	L	Nodal displacement
ILDUM	L	Not used at present

FACS(100)	R*8	Incremental loading factors

COMMON /INCSV/

Usage: This is an incremental loading save common.

Modules: Stsmod

Parameters:

Name	Type	Usage
IG	I	File number for IGEOM
IP	L	Print flag (LOUT(1))
IFLG	L	First time flag. Set TRUE in the block data and then set FALSE in INCLDC.

COMMON /ISMAT/

Usage: This common is used to pass material properties for the stress calculation on isoparametric elements.

Modules: STSMOD

Parameters:

Name	Type	Usage
MU	R*8	E/(1+NU)
NU	R*8	Poisson's ratio

COMMON /ISOMAT/

Usage: This common provides the material counts and material numbers for isoparametric stress calculations.

Modulus: Solmod, Stsmod

Parameters:

Name	Type	Usage
MTCNT	I	Number of materials in problem
MATS(50)	I	Material numbers in problem

COMMON /KOMSTF/

Usage: This common stores the stiffness matrix and force vectors for the various elements in the solution module.

Modules: Solmod

Parameters:

Name	Type	Usage
MFULL	I	Dimension of stiffness matrix
LIMNAT	I	Number of rows to be eliminated by elimination routine
S	R*8	Stiffness matrix, force vectors, etc., depending on the element
ALFDLT	R*8	Thermal load

COMMON /LDCNTL/

Usage: This common stores the information needed to control the large deformation option.

Modules: Inmod, Driver, Solmod, Stsmod

Parameters:

Name	Type	Usage
LDKT	I	Iteration counter
LDLIM	I	Number of last iteration to be allowed
LDPT	L	Flag for print-out of displacements
LDCONV	L	Flag for convergence
CNVFCI	R*8	Convergence factor
URF(100)	R*8	Under relaxation factors for iterations 1-100

COMMON /LDKOM/

Usage: This common is used to pass the data required for the large deformation routines.

Modules: Solmod, Stsmod

Parameters:

Name	Type	Usage
ALBR11	R*8	Nonlinear
ALBR22	R*8	correction
ALBR33	R*8	terms
ALBR13	R*8	
CC	R*8	
CBAR	R*8	
EP11	R*8	Strain
EP33	R*8	components
EP13	R*8	
EP22	R*8	
EP1	R*8	Principal
EP2	R*8	strains in R-Z
ANGL	R*8	Angle of principal strain
ADIVA	R*8	Ratio of undeformed area to deformed area
H	R*8	Herrman's variational
RR(22)	R*8	Element coordinates
UU(22)	R*8	Element displacements

COMMON /LMAT/

Usage: This common contains the material properties for an element.

Modulus: Solmod

Parameters:

Name	Type	Usage
ELMAT(19)	R*8	Material properties in orthotropic form
ANGL	R*8	Orthotropic angle of rotation
AREA	R*8	Area of element
TEMP	R*8	Temperature of the element

COMMON /MATPR/

Usage: This common provides work space for the ordering of the parameters (material properties) within the subroutine.

Modules: Inmod

Parameters:

Name	Type	Usage
XMAT	R*8	Material number
E(3)	R*8	Young's moduli
XNU(6)	R*8	Poisson's ratios
G	R*8	Shear modulus
ALP(3)	R*8	Alpha (these are the thermal loading coefficents)
TEMP	R*8	Temperature
STRN	R*8	Strain
ANM(3)	R*8	Material name

COMMON /NRGEE/

Usage: This common contains total energy and variables used for accuracy check.

Modules: Stsmod

Parameters:

Name	Type	Usage
TCON	R*8	Element temperature constant
ENTOT	R*8	Total energy
TOTCON	R*8	Total temperature constant
IAC(10)	I	I indices of elements with the greatest difference (KU-F)
JAC(10)	I	J indices of elements with the greatest difference (KU-F)
KAC(10)	I	Flags which indicate the degree of freedom for which the largest difference occurred
ABSR(10)	R*8	KU-F
CAYU(10)	R*8	KU
EF(10)	R*8	F

COMMON /OUTPUT/

Usage: This common contains the flags for output options with respect to the entire output set.

Modules: Inmod, Driver, Pltmod, Prtmod, Stsmod

Parameters:

Name	Type	Usage
LOUT(20)	L	These appear in the same order and have the same meaning as the parameters in the input document.

COMMON /PAGE/

Usage: The data stored in this common controls the printing of page headings.

Modules: Inmod, Driver, Pltmod, Prtmod, Solmod

Parameters:

Name	Type	Usage
NPGS	I	Pages of output required for this print
ILNS	I	Lines left on page
IPAGE	I	Current page number
LNKNT	I	Line counter

COMMON /PLTCOM/

Usage: This common stores data necessary for plotting.

Modules: Inmod, Driver, Pltmod

Parameters:

Name	Type	Usage
IMN	I	Indices which define
JMN	I	the region over
IMX	I	which we are
JMX	I	plotting
DSPM	R*8	Displacement multiplier
XSCL	R*8	X-scale factor
YSCL	R*8	Y-scale factor
XMIN	R*8	X-minimum
YMIN	R*8	Y-minimum

XMAX	R*8	X-maximum
YMAX	R*8	Y-maximun
XPS	R*8	Paper size, X-direction
YPS	R*8	Paper size, Y-direction
XTL	R*8	Coordinates for title
YTL	R*8	location
ITL	R*8	Title size
IXY	R*8	Orientation factor
PTITL(6)	R*8	Plot title

COMMON /PRMC/

Usage: The first parameter in this common is a flag to determine which of the subroutines in Prtmod are entered.

Modules: Prtmod

Parameters:

Name	Type	Usage
NFLG(1)	I	Flag
		1-displacements
		2-stress
		3-energy
		4-accuracy check
		5-reaction forces
NFLG(2) or Mode	I	Mode number if asymmetric
NFLG(3)	I	Bit flag giving the degrees of freedom in the problem

COMMON /ROWEL/

Usage: This common is used in the solution module to pass indices.

Modules: Solmod, Stsmod

Parameters:

Name	Type	Usage
IMN(3)	I	Minimum I index for the rows of nodes in core
IMX(3)	I	Maximum I index for the rows of nodes in core
I	I	I index
J	I	J index for a row of elements in core
J6	I	J index of row of elements in core
LTYPE	I	Element type
IWD	I	Number of nodes across an element
JWD	I	Number of rows of nodes required for the J6th row of elements
I2	I	Same as IWD for this version
J2	I	Same as JWD for this version
NUMPAR	I	Maximum number of degrees of freedom in problem

COMMON /SLFILE/

Usage: This common contains the designation for the tape storing the information needed in the service life routine.

Modules: Driver, Pltmod, Prtmod

Parameters:

Name	Type	Usage
ISLF	I	Service life file. Set to 9 for FT09F001 in the block data

COMMON /SOLDIM/

Usage: This common stores the important dimensions for the solution module.

Modules: Solmod

Parameters:

Name	Type	Usage
K4	I	Maximum number of equations to be reduced at one time
K5	I	Semi-bandwidth
K6	I	Number of equations for a row of elements
IH	I	Number of equations to be reduced at one time for a row of elements
NF	I	Column limiter for reduction
NCOL	I	Semi-bandwidth for a row of elements
NS	I	Same as K6 at present
NREC	I	Record count on the solution file
IL	I	Minimum I for a row of elements
IU	I	Maximum I for a row of elements
K3	I	Required size of solution vector
K9	I	Number of degrees of freedom on a row of nodes
NDOF	I	Number of degrees of freedom on a node

COMMON /SPRINT/

Usage: This common provides work space for the ordering of the parameters (stresses and strains) within the subroutine.

Module: Prtmod

See the using routines for structure.

COMMON /STIMOD/

Usage: This is the common for the stability module.

Modules: Inmod, Stamod

Parameters:

Name	Type	Usage
X	R*8	These variables
Y	R*8	are the input to
A	R*8	the stability
B	R*8	module
Z	R*8	
C	R*8	
CONVF	R*8	
MAXITR	I	
ID	I	Dummy for boundary alignment
XNOS	R*8	Number of shims
OSF	R*8	Old shape factor

COMMON /STOUT/

Usage: This is the stress output common.

Modules: Stsmod

Parameters:

Name	Type	Usage
LEN	I	The number of values beginning with ENGR to be written on IPRNT
INK	I	The index into the array where the incremental loading accumulation begins
ENRG	R*8	Element energy
		The remainder of this common will vary in length and content according to the stress routine involved.

COMMON /STSDIM/

Usage: This common stores an array of parameters needed by the stress module.

Modules: Stsmod

Parameters:

Name	Type	Usage
NDOF	I	Number of degrees of freedom
IDOF	I	A bit flag to indicate the degrees of freedom in the problem
K9	I	Number of displacements on a row of nodes
IPTR(60)	I	An array of pointers used to give the relation between the element stiffness matrix and the assembled stiffness matrix

COMMON /STSMAT/

Usage: This common contains the material properties for an element.

Modules: Stsmod

Parameters:

Name	Type	Usage
ELMAT(19)	R*8	The material properties in orthotropic form
ANG	R*8	Orthotropic angle of rotation
AREA	R*8	Area of element
TEMP	R*8	Temperature

COMMON /STSSTF/

Usage: This common stores the stiffness matrix and force vectors for the various elements.

Modules: Stsmod

Parameters:

Name	Type	Usage
MFULL	I	Dimension of stiffness matrix
LIMNAT	I	Number of rows to be eliminated by the elimination routine
S(20,24)	R*8	The stiffness matrix, force vectors, etc., depending on the element

COMMON /SRTCOM/

Usage: This common stores information relative to the sorting of records and the merging of files.

Modules: Pltmod

Parameters:

Name	Type	Usage
L5	I	Block length vs records for input

N5	I	Number of records per file on files ISCR5 and ISCR6 respectively
N6	I	
NM	I	Number of merges required
MM	I	Maximum number of records that can be held in core
LEN	I	Number of words per record
MAX	I	Total number of records

COMMON /STRNRC/

Usage: This common provides work space for those routines which accumulate or print stresses or strains for an asymmetrically loaded body.

Modules: Pltmod

Parameters:

Name	Type	Usage
TH	R*4	Theta or Z coordinate
R	R*4	Radial coordinate
MU	R*4	Shear modulus
NU	R*4	Poisson's ratio
EPR	R*4	R-strain
EPT	R*4	Theta strain
EPZ	R*4	Z-strain
GAMRZ	R*4	RZ shear
GAMRT	R*4	R-theta shear
GAMZT	R*4	Z-theta shear
H	R*4	Herrman's variational

THERM	R*4	Thermal load

COMMON/TROUBL/

Usage: This common contains an error flag for termination of program.

Modules: Inmod, Driver, Pltmod, Prtmod, Solmod, Stsmod

Parameters:

Name	Type	Usage
IRTRN	I	Error flag
		= 0 NORMAL RETURN
		.GT.0 TERMINAL ERROR
		.LT.0 TRY NEXT CASE

COMMON /WORK/

Usage: This is a work common which also stores material properties and loads for an asymmetric loading element.

Modules: Solmod

Parameters:

Name	Type	Usage
EE	R*8	Young's modulus
PNU	R*8	Poisson's ratio
BFRL	R*8	Body forces
BFTHL	R*8	
BFZL	R*8	
THRL	R*8	Thermal load
P	R*8	Pressure
TAU	R*8	Shear

PR	R*8	Traction loads
PTH	R*8	
PZ	R*8	
RS(3)	R*8	Triangular element coordinates
ZS(3)	R*8	
AR	R*8	Area of a triangular element

ERROR CODES

There are several user completion codes that may be given by the program. These are given here as an aid in debugging.

100	Issued by TDRR. An end of file was encountered on a read.
200	Issued by TDRR. A residual count error was detected on a read.
444	Issued by XFPTRP. An exponent overflow condition was detected.
500	Issued by GTMAIN. Insufficient core available to continue the run.
1000	Issued by TASSIO. A permanent I/O error has occurred on a data set.
1010	Issued by TASSIO. An error has been detected in the TASSCB. This may be caused by a bad file number or by the TASSCB being destroyed.
1020	Issued by TASSIO. Attempted to open a file that was already open.
1030	Issued by TASSIO. Error in a getmain required to open a file.
1040	Issued by TASSIO. An error occurred in the open routine.
1110	Issued by TASSIO. An attempt was made to write N words where N is less than one.
1120	Issued by TASSIO. Attempted to write on a file that was not open.
1200	Issued by TASSIO. An end-of-file was encountered and IRTN was set to -1.
1210	Issued by TASSIO. Attempted to read N words where N was less than zero.
1220	Issued by TASSIO. Attempted to read a file that was not open.

1230 Issued by TASSIO. Wrong length record. The number of words requested did not match the number of words in the record.

1310 Issued by TASSIO. Tried to do a backread on a file that was not open.

1320 Issued by TASSIO. Wrong length record on a backread. The number of words requested did not match the number of words in the record.

3276 Issued by ABORTA. This routine is called at various places in core where an error condition requiring termination with a dump has been detected. The traceback will show where the error occurred.

OVERLAY

Only the philosophy of the overlay structure will be given here. The detail of the structure can be readily seen from the linkedit list or a list of the overlay cards.

We have attempted to set up the overlay in a modular form to match the structure of the program. This way only the modules that are actually used will be loaded.

Within each module we have tried to group commons and routines together so that only the needed routines will be loaded and also so that the chains are as nearly equal in length as possible under the other restrictions.

This has resulted in a highly overlayed program with a fairly flat structure. The solution module is the longest module, and care should be taken that the program is not unduly lengthened by modifications to this module.