AFFDL-TR-70-118

AN AUTOMATED PROCEDURE FOR THE OPTIMIZATION OF PRACTICAL AEROSPACE STRUCTURES

VOLUME II—PROGRAMMER'S MANUAL

WALTER J. DWYER
ROBERT K. EMERTON
PATRICIA L. SABATELLI
GRUMMAN AEROSPACE CORPORATION

TECHNICAL REPORT AFFDL-TR-70-118, VOLUME II

APRIL 1971

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

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.

400-June 1971-CO305-40-71-672

AFFDL-TR-70-118

AN AUTOMATED PROCEDURE FOR THE OPTIMIZATION OF PRACTICAL AEROSPACE STRUCTURES

VOLUME II - PROGRAMMER'S MANUAL

WALTER J. DWYER
ROBERT K. EMERTON
IRVING U. OJALVO

GRUMMAN AEROSPACE CORPORATION

FOREWORD

Volume II of this report was prepared by the Structural Mechanics Section of the Grumman Aerospace Corporation, Bethpage, New York. It provides programmer's information on the structural optimization programs developed under USAF Contract No. F 33615-69-C-1278, which was initiated under Project No. 5710, "Structural Synthesis and Automated Design", Task No. 146705. The effort was administered by the Air Force Flight Dynamics Laboratory, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio. Dr. Vipperla B. Venkayya (FBR) was the project monitor. The theoretical development of the automated structural optimization methods and computer program user's information developed under this contract are presented in Volume I, "Theoretical Development and User's Information".

The project engineer and principal investigator of the present effort was Walter J. Dwyer, Structural Methods Engineer, Structural Mechanics Section. The authors wish to thank Frank Nolan, Nicholas Angrisano, Frank Van Roten and Clayton Wilkie for their help with the general organization and writing of the program.

This report covers work conducted from 15 January 1969 to 15 August 1970 and was submitted to the Air Force in August 1970. The contractors designation for Volume II is ADR 02-01-71.2.

The technical report has been reviewed and is approved.

Francis J. Janik Jr
Chief, Theoretical Mechanics Branch
Air Force Flight Dynamics Laboratory


BLANK PAGE

ABSTRACT

This volume documents the computer programs described in Volume I of this report entitled "An Automated Procedure for the Optimization of Practical Aerospace Structures". Both the main structural optimization program and the shell dynamics program are written in Fortran IV language. This manual contains a description of the overlay structure, data set arrangement, and subroutines of both programs.

TABLE OF CONTENTS

SECTION 1
INTRODUCTION

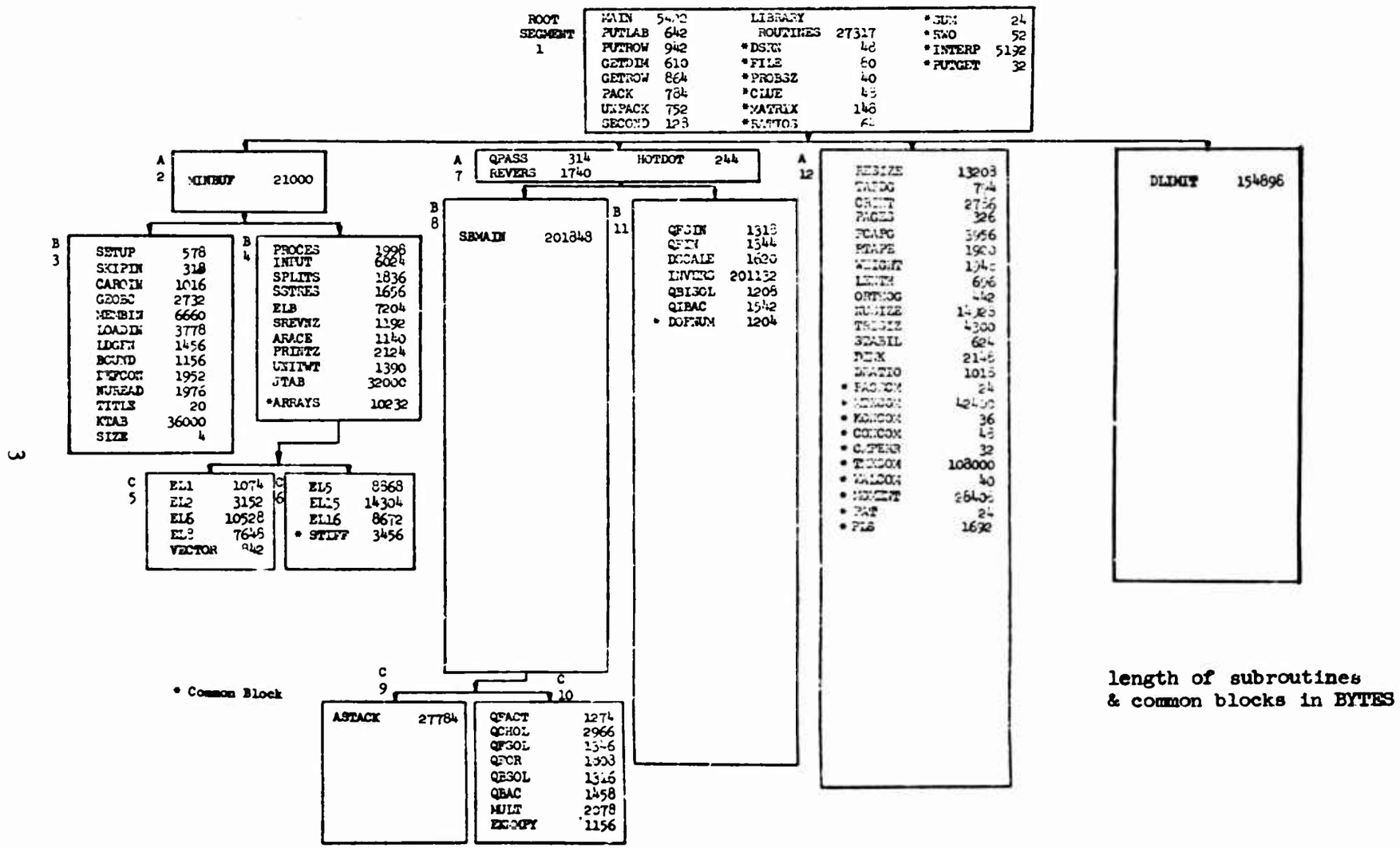
This manual documents programming information for the large scale finite element structural optimization program of Volume I, and the dynamic optimization study program.

The first portion contains a description of the overlay structure and data set allocation for the IBM 360/75 and the IBM 7094 version of the optimization program. Lengths of each common block, subroutine, and overlay link are given. All of the subroutines in the program are then described. The second portion contains flow charts and a description of the dynamic optimization study program.

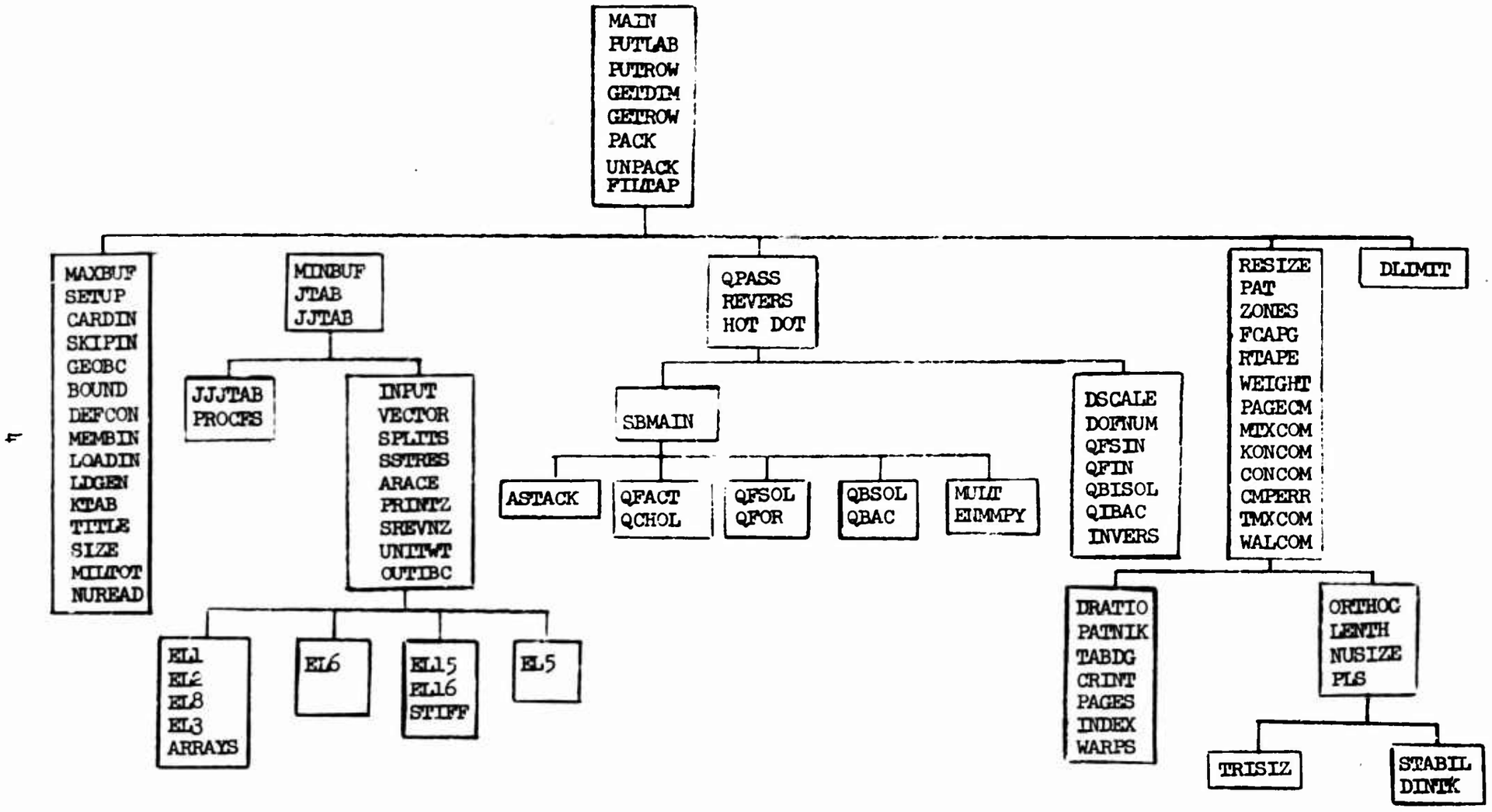
SECTION 2

OVERLAY CHARTS

The following chart is for the overlay structure of the large structural optimization program as used on an IBM 360/75 with a core partition of 380,000 bytes. The chart is approximately to scale and the lengths of the subroutines and common blocks are given in bytes. The second overlay chart is for the IBM 7094 version of the program.



Overlay Chart for the IBM 360/75 Version of the Optimization Program



Overlay Chart for the IBM 7094 Version of the Optimization Program

SECTION 3

DATA SET ALLOCATION

The following chart shows the allocation of data sets on an IBM 360/75 to hold all the blocks of data generated by the large structural optimization program. A maximum of four files per unit are used. A total of twenty one data blocks are created. The second chart shows an alternate data set allocation using only seven data sets.

UNIT #	UNIT NAME	FILES			
		1	2	3	4
1	JDSRN	NTAPC GEOMETRY	NTAPBC B.C.	NTAPL LOADS	NTAPM MEMBERS
2	ISCR	SCRATCH			
3	JSCR	SCRATCH			
4	MDSRN	NTAPK STRUCTURAL STIFFNESS MATRIX			
8	NDSRN	NTAPST STRESSES	NTAPM MEMBERS		
9	LDSRN	NTAPDC DEFLECTION CONSTRAINTS	NTAPES ELEM. STIFF	NTAPS CORNERFORCES	
10	IDSRN	NTAPLT L^{-1} LOWER TRIANGLE INVERSE	NTAPD DEFLECTION		
11	KDSRN	NTAPS L^{-1} LOWER TRIANGLE INVERSE	NTAPD DEFLECTION		
12	KSCR	NTAPS CORNERFORCES	SCRATCH		
13	LSCR	NTAPT NTABD,NTABR,MTABR	DEFLECTION DEFFERENCES		
15	NSCR	SCRATCH			

Data Set Allocation for the 360/75 Version of the Optimization Program

UNIT #	UNIT NAME	FILES			
		1	2	3	4
1	JDSRN	NTAPC GEOMETRY	NTAPBC B.C.	NTAPL LOADS	NTAPM MEMBERS
2	ISCR	SCRATCH			
3	JSCR	SCRATCH			
4	MDSRN	NTAPK STRUCTURAL STIFFNESS MATRIX NTAPDD DEFLECTION DIFFERENCES	SCRATCH		
8	NDSRN	NTAPST STRESSES	NTAPM MEMBERS		
9	LDSRN	NTAPDC DEFLECTION CONSTRAINTS	NTAPES ELEM. STIFFNESSES	NTAPS CORNERFORCES	SCRATCH
10	IDSRN	NTAPLT DECOMPOSED LOWER TRIANGLE	NTAPD DEFLECTIONS	NTAPIS INVERSE STIFFNESS K^{-1}	

Data Set Allocation for the IBM 7094 Version of the Optimization Program

SECTION 4

DESCRIPTION OF SUBROUTINES

Each of the subroutines in the structural optimization program has been reviewed and, where necessary, comment statements have been added to the listing. As additional help in understanding the various subroutines, a short description of each one is presented here. Each description contains an outline of the algorithm, an explanation of the subroutine input and output, error messages that may arise, the subroutines called, the variables in the subroutine argument list, and a list of important variables. The subroutines are arranged in the order in which they appear in the program listing.

Program MAIN

a. Algorithm

This program controls the optimization of the structure by calling various subroutines in the proper sequence as follows:

1. call SETUP - this subprogram assigns tape and disk units to hold the various data blocks which will be generated.
2. call CARDIN - this subprogram controls the reading and checking of the input information.
3. call PROCES - the degree of freedom numbers are assigned to the node points.
4. call INPUT - generate element stiffness and stress matricies.
5. call SBMAIN - solve for deflections and corner forces.
6. call RESIZE - using the corner forces compute the stresses and resize the structure based on the stress and size constraints.
7. call DRATIO - if there are deflection constraints ,get the largest deflection constraint ratio.
8. Repeat steps 5 thru 7 until weight increases or the maximum number of cycles has been performed;then either print the final stresses or enter deflection constraint mode with the lightest design yet obtained.
9. If program is entering or is in deflection constraint mode ,call DSCALE which will find the violated degrees of freedom and write them on a scratch tape.
10. call INVERS - compute as much of the inverse of the stiffness as is needed to compute the gradient.
11. call DLIMIT - resize the members to satisfy the deflection constraints.
12. call SBMAIN, call RESIZE, call DRATIO - analyze structure and compute stress and deflection constraint ratios
13. Perform steps 9 thru 12 until weight starts to increase or until the maximum number of cycles is reached. At that point print out stresses of the optimized structure.

b. Input/Output

This program reads in the initial structure and loads,and prints out the optimum structure.

c. Error Messages

None

d. Subroutines Required

All

e. Argument List

None

f. Important Variables

DRATO - old deflection constraint ratio
DRATN - new deflection constraint ratio
WRATO - old stress constraint ratio
WRATN - new stress constraint ratio
SUMO - old total structural weight
SUMN - new structural weight
SFO - old scale factor
SFN - new scale factor
SLFAL - deflection constraint relaxation factor

Subroutine SBMAIN

a. Algorithm

This subroutine controls the stacking of the element stiffness matrix, the solution of equations routines to obtain the deflections, and the generation of the corner forces.

b. Input/Output

The subroutine analyzes the current structure and places the corner forces and deflections on tape.

c. Error Messages

None

d. Subroutines Required

ASTACK, QFACT, QFSOL, REVERS, QBSOL., ENMMPY, MULT

e. Argument List

None

f. Important Variables

KORE - size of work area for stacking and solving equations

Subroutine SETUP

a. Algorithm

This subroutine was written to collect together all the tape and disk unit assignments.

b. Input/Output

There is no input to this program. The output is unit numbers that will store the different sets of data needed in the program.

c. Error Messages

None

d. Subroutines Required

None

e. Argument List

None

f. Important Variables

None

Subroutine SKIPIN

a. Algorithm

This subroutine reads cards of data in the input stream until it finds the blank card at the end of the block.

b. Input/Output

none

c. Error Messages

none

d. Subroutines Required

none

e. Argument List

none

f. Important Variables

none

Subroutine CARDIN

a. Algorithm

This subroutine reads the label card at the beginning of each block of input data and,based on the parameter in the label statement, transfers control to the correct subroutine to read the data in that group. Also,the instability table data is read in directly by this subroutine. If, at the end of the input data there are no errors, control is passed back to the main program. Otherwise execution is terminated.

b. Input/Output

The label card at the beginning of each type of data is read. Instability tables are read in.

c. Error Messages

ERROR IN READING IN STABILITY TABLES.
NUMBER OF LOAD CONDITIONS NOT SPECIFIED.
DUE TO INPUT ERRORS, EXECUTION WILL NOT CONTINUE

d. Subroutines Required

SKIPIN, GEOBC, MATRAL, MEMBIN, LOADIN, BOUND, DEFCON, NUREAD.

e. Argument List

None

f. Important Variables

LDCOND - number of loading conditions
ITYPED - parameter in the label statement indicating what type of data is about to be read.
IGO - error switch,0 = no input errors

Subroutine GEOBC

a. Algorithm

This subroutine reads geometry and boundary conditions. If ITYPED equals 1, geometry and boundary conditions are read from the same card. If ITYPED equals 2, only geometry is read. If ITYPED equals 3, only boundary conditions are read. The x, y, znd z coordinates are stored in arrays X. Y, and Z in the order in which they are read in. If no errors are present in the input data the corresponding joint number is stored in array JTNØ and the geometry data is written as a matrix on tape NTAPC. If there are no errors present, degrees of freedom numbers are assigned to the allowable displacements at the nodes. A matrix containing the degree of freedom numbers for each node is written on tape NTAPBC.

b. Input/Output

Geometry and/or boundary conditions are read in. A tape containing the geometry matrix and a tape containing the boundary condition matrix are created.

c. Error Messages

TWO JOINTS HAVE THE SAME NODE NUMBER
JOINT NUMBER IS TOO LARGE, MAXIMUM IS ____
MAXIMUM NUMBER OF JOINTS EXCEEDED
BOUNDARY CONDITION FOR NODE ____, COMPONENT ____, IS NOT VALID

d. Subroutines Required

PUTLAB, PUTROW

e. Argument List

ITYPED

f. Important Variables

ITYPED - described above

MAXJNO - maximum joint number allowed

KORDER - one dimensional array containing node numbers

KOR - running count of the number of joints used

JTNO - joint number array

X - x - coordinate of joint

Y - y - coordinate of joint

Z - z - coordinate of joint

ICHKSW - error check switch, 0 if no errors

BUF - buffer for assembling row of geometry or B.C. matrix

IBUF - " " " " " " " " "

K11(J) - number of degrees of freedom of joint J

K11SUM(J) - degree of freedom number of first degree of freedom at node J.

NTAPC - tape number of new tape with geometry matrix

NTAPBC - tape number of new tape with boundary condition matrix

Subroutine MEMBIN

a. Algorithm

This subroutine reads the material property table, if present, and updates the stored material property table. It also reads the member cards and forms a member pseudo matrix on tape, one row for each member, one humdred locations per member. The member data cards are arranged so that input data is minimized. Standard values for many of the member parameters are computed from the material property tables and automatically entered into the proper "bin" in the row of the member matrix. These values may be overridden by applying alternate values on additional member cards. More "bins" have been allocated for member data than is currently being used, so the program may be modified in the future with a minimum of programming changes. A sum of the number of forces in the stress matrix is created.

b. Input/Output

The input for this program is the material property table and the member data cards. The output for this subprogram is the member pseudo matrix residing on tape number NTAPM.

c. Error Messages

THE ABOVE MATERIAL UPDATE CARD HAS AN INVALID QUANTITY
MEMBER CARD HAS INVALID CLASSIFICATION
INVALID MEMBER TYPE
MEMBER NO. ____ HAS THE FOLLOWING JOINT REPEATED
BUFF (____) CONTAINS A ZERO VALUE FOR MEMBER NUMBER _____
MEMBER NO. _____ HAS AN INVALID COMBINATION OF ELASTIC PROPERTIES
MEMBER _____ HAS ELASTIC PROPERTIES MISSING

d. Subroutines Required

CARDIN, PUTLAB, PUTROW

e. Argument List

None

f. Important Variables

IBUFF - buffer area for assembling member pseudo matrix

BUFF - " " " " " " "

MILSUM - total number of corner forces in the structure and length of stress matrix that will be constructed later.

ICHECK - error check clue

Subroutine LOADIN

a. Algorithm

The subroutine reads the load cards, one at a time, and checks to see if the load matrix is in row sort. If the loads are correct, they are written on tape as a pseudo matrix. The first column of each row is the node number, the second column the component of the load, and the third thru last columns contain the loads for each loading condition. Also, the sum of the forces and moments about the origin in each load condition is printed out to aid in checking the input data. If no errors are present, subroutine LDGEN is called to generate the real load matrix using the pseudo load matrix.

b. Input/Output

Input to this routine is the input load cards. Output is a load pseudo matrix and a summary of the applied loads.

c. Error Messages

NODE NUMBER IS INVALID
COMPONENT IS INVALID
LOAD MATRIX IS NOT IN ROW SORT
LOAD CONDITION IS INVALID

d. Subroutines Required

PUTLAB, PUTROW, LDGEN

e. Argument List

LDCOND - number of loading conditions

f. Important Variables

LNODE - node number of present load
LCOMP - component of present load
ELMT - value of present load

SUMMX - total X moment about origin

SUMMY - " Y " " "

SUMMZ - " Z " " "

SUMF - sum of applied forces in three directions

BUF - buffer area for assembling load information to be put on tape

Subroutine LDGEN

a. Algorithm

This subroutine reads the tape containing the pseudo load matrix created in LOADIN. Each row is checked against the boundary condition matrix IBC, and against the list of nodes contained in KORDER. If the load is being applied to a strainable node the load is placed in the proper row of the load matrix. The row is put on tape when a new load is read that does not belong in the present row. At this point, blank rows are put on the tape until the matrix has been indexed down to the point where the new load belongs.

b. Input/Output

The input is the pseudo load matrix on tape number LOADIN. The output is the real load matrix residing on tape MAT1

c. Error Messages

LOAD PLACED ON NODE _____ COMPONENT _____ WHICH IS PRESCRIBED
LOAD SPECIFIED FOR NODE _____ WHICH IS NOT INCLUDED IN B.C.
LOAD PLACED ON NODE _____ COMPONENT _____ WHICH IS NON STRAINABLE
ERROR *** NODE _____ COMPONENT _____ *** LOAD MATRIX IS NOT IN ROW SORT.

d. Subroutines Required

PUTLAB, PUTROW, GETDIM, GETROW

e. Argument List

LOADIN - tape number for input pseudo load matrix
MAT1 - tape number for real load matrix.

f. Important Variables

LN - loaded node number
LC - loaded component number
KORDER - array containing nodes in the order in which they were input
IBC - boundary condition array

Subroutine PROCES

a. Algorithm

This subroutine reads the geometry and boundary condition matrices and checks them for compatibility. The boundary condition matrix is read into core and a running total on the number of degrees of freedom at each node is kept. After the boundary condition matrix is read, the geometry matrix is read. Both are then printed out together so that they may be examined for errors.

b. Input/Output

The geometry and boundary condition matrices are read in and both are printed out for checking.

c. Error Messages

GEOMETRY HAS ____ NODES WHILE B.C. HAS ____
ERROR *** GEOMETRY CONTAINS NODE NUMBER ____ NOT FOUND IN B.C.
I/O ERROR DURING PRE-PROCESSING *** MATRIX GENERATION SUPPRESSED

d. Subroutines Required

GETROW, GETDIM

e. Argument List

None

f. Important Variables

K11 - number of free degrees of freedom for node

Subroutine INPUT

a. Algorithm

This subroutine writes a heading for the member data and calls SPLITZ to initialize the stress matrix. It then reads a row of the member pseudo matrix from the tape and prints out the geometric and physical properties of that member. The subroutine then calls the appropriate finite element subroutine to calculate the individual element stiffness and stress matrices. The above operations are repeated for each member, in turn, until the entire pseudo matrix is read.

b. Input/Output

The member pseudo matrix is supplied and a tape containing the element stiffness matrices and stacking indices used to form the total stiffness matrix is produced.

c. Error Messages

MEMBER NO. ____ HAS INVALID TYPE ____
MEMBER ____ CONNECTS NODE ____ WHICH IS UNDEFINED
INCORRECT MATRIX SUPPLIED FOR MEMBERS -
MATRIX NAME IS _____.

d. Subroutines Required

SPLITZ, PUTLAB, GETDIM, GETROW

e. Argument List

None

f. Important Variables

NTAPST - tape containing stress matrix for the total structure
NTAPK - tape containing stiffness matrix for the total structure

Subroutine SPLITS

a. Algorithm

SPLITS computes the stacking indices for the element stiffness matrices by comparing the nodes of the element to the degree of freedom numbers of the nodes. All the zeroes are compressed out of the element matrices by calling ARACE. Next, the band width of the total stiffness matrix is computed by finding the degree of freedom number of the component of the element stiffness matrix that is furthest from the diagonal of the total stiffness matrix. The weight of the element for unit thickness is calculated by calling UNITWT. The member number, number of nodes, row and column indices into the total stiffness matrix, weight for unit thickness, and member type are written on tape NTAPES. SSTRES is then called and the program returns to the element subroutine.

b. Input/Output

The subroutine uses as input the stiffness matrix and stress matrix from the element subroutine and produces the stiffness matrix for unit thickness and the stacking indices on tape NTAPES.

c. Error Messages

None

d. Subroutines Required

ARACE, UNITWT, PRINT2, SSTRES

e. Argument List

STIFF - element stiffness matrix
STRESS- element stress matrix
LIN - number of nodes for the element
MIL - lines of stress output for the member
MOD - maximum degrees of freedom for each node - 3 for membrane elements, 6 for bending elements

JI - joints of the element

MEMNO - member number

f. Important Variables

IRI, ICI - size of non condensed stiffness matrix

IBANDW - bandwidth of structure's stiffness matrix

NTAPE - tape containing information described above

Subroutine SSTRES

a. Algorithm

The element stress (force) matrices for unit design parameters are "stacked" into a stress (force) matrix for the entire structure. The rows of each element matrix are entered sequentially into the total array and the columns are entered according to degree of freedom numbers associated with the element. As a result, the total matrix, which is stored on tape, will multiply the displacement matrix to yield corner forces (moments) for unit values of the design parameters. The total array is generated and written in blocks if not enough core storage is available.

b. Input/Output

Input to this routine consists of element stress (force) matrices that have had boundary conditions applied to them in SPLITS. The output is the total matrix for the idealized structure for unit design parameters, written on tape MATRIX.

c. Error Messages

None

d. Subroutines Required

PUTROW

e. Argument List

SMALL - element stress matrix
MID - maximum number of rows of any element's stress matrix
NODES - number of nodes of the element
MIL - number of rows of the element's stress matrix
MCOL - number of columns of the element's stress matrix
NSTART - index showing where the first column of the element matrix is placed in the total stress matrix

NGO - number of free degrees of freedom for all nodes of the element

MATRIX - tape containing "stacked" matrix

KLU - index showing where the first row of the element matrix is placed in the total array.

f. Important Variables

WK - one dimensional array containing elements of the structure's stress matrix

MSTART, NKEY - indices used in sorting and placing of the element matrix into the total array.

Subroutine ASTACK

a. Algorithm

This subroutine reads the element stiffness matricies and associated stacking indicies from tape INFILE. These element stiffness matricies are then multiplied by the element thickness obtained from the member pseudo matrix on NTAPM. The element area is also entered into the array AREA so if the stiffness matrix can not be stacked in one block, it will not be necessary to read the member pseudo matrix a second time. The stiffness matrix is formed in blocks in the array WKAREA. The amount that can be stacked in any one block depends on the band width of the stiffness matrix. All of the matricies are read into core, one at a time, for the stacking of each block. The portion of the element stiffness matricies that does not belong in the block of the stiffness matrix being stacked is discarded. There is no limit on the size of the stiffness matrix that may be handled by this subroutine as long as at least one row of the matrix will fit in the work area at a time.

b. Input/Output

The input to this subroutine is the member matrix and the element stiffness matricies. The output is the total stiffness matrix.

c. Error Messages

None

d. Subroutines Required

PUTROW, FILTAP, GETDIM, GETROW

e. Argument List

INFILE - tape containing element stiffness
MATRIX - tape to contain global stiffness matrix for the sturcture
WKAREA - array used to stack stiffness matrix
IWKPR - same as WKAREA but in fixed point
WORK - same as WKAREA but in single precision
ISIZE - size of WKAREA

Subroutine EL1

a. Algorithm

This subroutine generates the bar element stress and stiffness matrices in global coordinates, for unit cross sectional area.

b. Input/Output

The subroutine uses the coordinates of the node points and the elastic properties passed thru common as input. The output is the element stiffness and stress matrices in global coordinates without application of boundary conditions.

c. Error Messages

*** MEMBER NUMBER ____ HAS ZERO LENGTH ****

d. Subroutines Required

SPLITS

e. Argument List

None

f. Important Variables

AKG - global stiffness matrix
STG - global stress matrix
COORD - coordinates of the element node points
AL12 - length of the element

Subroutine EL2

a. Algorithm

This subroutine generates the beam element stiffness and stress matrices in global coordinates. An entry point, EL11, is present to handle a beam with one hinged end.

b. Input/Output

Same as in EL1.

c. Error Messages

****MEMBER NUMBER ___ HAS ___ EQUAL TO ZERO ****

d. Subroutines Required

SPLITS

e. Argument List

None

f. Important Variables

AKG, STG, COORD, ALI2 - same as in EL1

CCT - transformation matrix from local to global coordinates.

BETA - angle between the beam's local x-y plane and its principal axis.

XIYY - moment of inertia about local y axis for unit cross sectional area and zero BETA angle.

XIZZ - moment of inertia about local z axis for unit cross sectional area and zero BETA angle.

XPOL - polar moment of inertia for unit cross-sectional area and zero BETA angle.

Subroutine EL3

a. Algorithm

Entry points EL4 and EL4A in this routine handle the isotropic and anisotropic triangular membrane elements. Global stiffness and stress matrices for a unit thickness are generated. Entry points TRI and TRIA generate local stiffness matrices for isotropic and anisotropic triangles that are passed back to quadrilateral element routines.

b. Input/Output

Same as in EL1.

c. Error Messages

Same as in EL2.

d. Subroutines Required

SPLITS

e. Argument List

None

f. Important Variables

AKG, STG, COORD	- same as in EL1
A11, A22, A33, ...	- stress-strain coefficients for plane stress element
AX2, AL23, AL13	- lengths of triangle's sides
CT	- transformation matrix from local to global coordinates.
BETA	- angle between local x axis and axis defining directional nature of elastic properties.

Subroutine EL5

a. Algorithm

Calling of the subroutine itself will result in generation of global stiffness and stress matrices for a planar isotropic quadrilateral membrane element. Entry point EL5A handles the anisotropic case.

b. Input/Output

Same as in EL1

c. Error Messages

Same as in EL2

d. Subroutines Required

SPLITS

e. Argument List

None

f. Important Variables

AKG, STG, COORD, BETA, CT, - same as in EL3
A11, A22, A33, . . . - same as in EL3
XCG, YCG - coordinates of elements centroid, located for purpose of defining four triangles.

Subroutine EL6

a. Algorithm

This routine generates element stiffness and stress matrices for the warped shear panel.

b. Input/Output

Same as in EL1.

c. Error Messages

*** MEMBER NUMBER ___ HAS ___ EQUAL TO ZERO ***

*** MATRIX ___ OF MEMBER ___ IS SINGULAR ***

d. Subroutines Required

VECTOR, SREVN2, SPLITS

e. Argument List

None

f. Important Variables

AKG, STG, COORD - same as in EL1.

G - shear modulus of elasticity.

AA, BB - - vectors connecting opposite corners of warped quad element- used to determine location of reference plane

ALPHA - flexibility coefficient determined from equilibrium considerations for the shear panel (Ref. 30)

Subroutine EL8

a. Algorithm

Element matrices for the warped quadrilateral membrane element are generated. Entry point EL8A handles the anisotropic element.

b. Input/Output

Same as in EL1.

c. Error Messages

*** THE VARIABLE GAM IN EL8 HAS THE FOLLOWING INCORRECT VALUE ____ ***
*** MEMBER NUMBER ____ HAS ____ EQUAL TO ZERO ***
*** MATRIX ____ OF MEMBER ____ IS SINGULAR ***

d. Subroutines Required

VECTOR, SREVN2, SPLITS, TRI (E.P.), TRIA (E.P.)

e. Argument List

None

f. Important Variables

AKG, STG, COORD, BETA - same as in EL3
CC - coordinate transformation from global to reference plane coordinates.
A,B - vectors connecting quad's opposite corners - used to determine location of reference plane.

Subroutine SREVN2

a. Algorithm

Within some of the finite element subroutines, the need arises to obtain the inverse of a matrix of small order, say 7 x 7. This routine obtains the inverse through the Gauss-Jordan elimination scheme with partial pivoting. It returns the inverse in the position of the original matrix.

b. Input/Output

Input consists of a small order matrix while the output consists of its inverse.

c. Error Messages

None

d. Subroutines Required

None

e. Argument List

A - elements of matrix to be inverted
M - order of matrix A
LOC - location of largest element in absolute value within each column of A
MID - dimension or maximum value of M
NIX - error indicator; if, upon return, NIX is not equal to zero, a column (or row) of A is all zeroes.

f. Important Variables

Same as argument list.

Subroutine VECTOR

a. Algorithm

This subroutine is capable of performing many operations with vectors. Upon setting proper clues in the calling statement, it will compute the dot product or cross product of two vectors, compute the distance between two points, or normalize a given vector.

b. Input/Output

The input consists of two vectors, A and B, upon which various operations are performed. Output is a resultant vector, C.

c. Error Messages

None

d. Subroutines Required

None

e. Argument List

A,B - input vectors
C - resultant vector
KLU - index determining operation, i.e. dot product, cross product, normalization, etc.

f. Important Variables

Same as argument list.

Subroutine ARACE

a. Algorithm

This subroutine eliminates rows and/or columns of the element stiffness or stress matrix A, according to the boundary conditions associated with the element.

b. Input/Output

This subroutine receives the element stiffness and stress matricies thru common and returns thru common the condensed matricies.

c. Error Messages

None

d. Subroutines Required

None

e. Argument List

A - matrix to be reduced
IROWA - number of rows of A
ICOLA - number of columns of A
JI - array containing the ordering of the element nodes as they appear in the input geometry
LIN - number of joints in the element
MOD - three or six - depending on how many degrees of freedom the element has at each node
ISWITCH - 1 - erase columns
2 - erase rows
3 - erase rows and columns
JR - number of rows in array containing A
JC - number of columns in array containing A

f. Important Variables

See argument list

Subroutine QFACT

a. Algorithm

This subroutine reads the positive definite symmetric stiffness matrix and sets up indicies for the subroutine QCHOL to get the lower triangle of an $L L^T$ decomposition one row at a time.

b. Input/Output

This subroutine receives the names of the stiffness and lower triangular matricies and uses them together with there sizes to control QCHOL.

c. Error Messages

**** ERROR ____ ****

**** ERROR - DIMENSIONS READ FROM LABEL OF TOTAL STIFFNESS MATRIX INDICATE THAT THE MATRIX IS NOT SQUARE ****

d. Subroutines Required

GETDIM, PUTLAB, QCHOL

e. Argument List

MA - total stiffness matrix tape name

ML - tape name for inverse of lower triangle

M1 - scratch tape

M2 - scratch tape

f. Important Variables

KORE - size of array used for working storage

M - size of stiffness matrix

Subroutine QCHOL

a. Algorithm

This subroutine performs the factorization of the stiffness matrix under the control of QFACT. The factorization is done in blocks. If the end of the matrix fits in core the clue KEY is set to 0.

b. Input/Output

The subroutine receives the stiffness matrix in blocks from the calling routine QFACT, and returns the lower triangle factorization on the tape ML.

c. Error Messages

None

d. Subroutines Required

UNPACK, GETROW

e. Argument List

A - work area
X - vector to hold one row of stiffness matrix
L(K) - number of elements from the first zero element in a row, to the diagonal
M - size of stiffness matrix
NU - row number of first row being processed on this pass.
KORE - size of work area
ML - tape to hold lower triangle
MI - tape with the input matrix
MO - scratch tape
KEY - clue, last time thru KEY = 0
KEE - clue for delivering output
NIX - error return
WORST - cancellation factor
NAME - name of stiffness matrix

f. Important Variables

See argument list

Subroutine QPASS

a. Algorithm

QPASS dummy reads over the required number of rows of the lower triangle decomposition of the stiffness matrix to aid in zoning for the forward solution.

b. Input/Output

This subroutine uses the starting point NU for the zoning of the forward solution, and a scratch array T into which a row at a time of the lower triangle, residing on ML, is to be dummy read.

c. Error Messages

None

d. Subroutines Required

None

e. Argument List

NU - starting point in zoning forward solution
ML - tape containing lower triangle
T - scratch array

f. Important Variables

See argument list

Subroutine QFSOL

a. Algorithm

QFSOL sets up and manages the zoning for the forward solution of $L^T X = L^{-1} y = Z$ using L, and y, getting Z.

b. Input/Output

This subroutine uses the lower triangle decomposition of the stiffness matrix, residing on ML, and the applied loads on MY, to produce the Z matrix on MZ.

c. Error Messages

**** ERROR ____ ****

d. Subroutines Required

GETDIM, PUTLAB, QFOR, QPASS

e. Argument List

ML - tape with lower triangle

MY - tape with applied loads

MZ - tape to receive the Z matrix

M1 - scratch tape

M2 - scratch tape

f. Important Variables

See argument list

Subroutine QFOR

a. Algorithm

QFOR actually computes the forward solution of $L^{T}X = L^{-1}y = Z$. It is called by QFSOL every time that the available core must be zoned.

b. Input/Output

This subroutine uses the lower triangle decomposition of the total stiffness matrix, residing on ML, and the applied loads on MY, to produce the intermediate Z matrix on MZ.

c. Error Messages

None

d. Subroutines Required

GETROW, UNPACK, PUTROW

e. Argument List

T - scratch arrays
Z - scratch arrays
L - scratch arrays
M - number of rows in lower triangle
N - number of load cases
NU - zoning clue
MIDZ - number of free degrees of freedom
ML - tape containing lower triangle
MZ - tape to contain intermediate Z matrix
MI - scratch tapes
MO - scratch tapes

f. Important Variables

See argument list

Subroutine QBSOL

a. Algorithm

QBSOL solves the equation $L^T X = Z$ to obtain X which is the matrix of nodal deflections. In general QBSOL only manages the solution; the actual computations are done in QBAC. If JDEFL is non zero, the deflections are printed out.

b. Input/Output

This subroutine uses the Z matrix on MZ and the lower triangle matrix on MB to produce the deflections on MX.

c. Error Messages

**** ERROR ____ ****

d. Subroutines Required

GETDIM, PUTLAB, QBAC

e. Argument List

MB - tape containing lower triangle
MZ - tape containing Z matrix
MX - tape to contain deflection matrix
M1 - scratch tape
M2 scratch tape

f. Important Variables

See argument list

Subroutine QBAC

a. Algorithm

QBAC, when called by QBSOL, actually computes the backward solution of the equation $L^T X = Z$, thus obtaining the matrix of nodal deflections X.

b. Input/Output

This subroutine uses the intermediate Z matrix in reverse order, residing on MZ, and the lower triangular matrix in reverse order on MB, to produce the deflections on MX, also in reverse order.

c. Error Messages

None

d. Subroutines Required

GETROW, UNPACK, PUTROW

e. Argument List

T - scratch arrays
B - scratch arrays
X - scratch arrays
L - scratch arrays
M - number of rows in lower triangle
N - number of load cases
MU - zoning clues
KORE - amount of available storage
MB - tape containing lower triangle in reverse order
MX - tape to create nodal deflections in reverse order
MI - scratch tapes
MO - scratch tapes

f. Important Variables

See argument list

Subroutine REVERS

a. Algorithm

This subroutine is used to reverse the order of a matrix on a tape when the matrix will not fit in core. The subroutine reads the matrix into the array BUFFER. When the array is full, the subroutine keeps reading the matrix in, putting the new rows into BUFFER on top of the old rows until all the matrix is read in. At this time the array will contain the end of the matrix which is then written out in reverse order onto tape MAT2. The first matrix is then again read in until the last row read onto MAT2 is reached Then the portion of the matrix in BUFFER is read out backwards onto MAT2. This continues until all the matrix on MAT1 has been reversed and put on MAT2.

b. Input/Output

The subroutine uses a matrix on tape MAT1 as input, and puts out the same matrix in reverse order on MAT2.

c. Error Messages

None

d. Subroutines Required

GETDIM, PUTLAB, GETROW, PUTROW

e. Argument List

MAT1 - tape containing matrix to be reversed

MAT2 - tape containing reversed matrix

f. Important Variables

See argument list

Subroutine QFSIN

a. Algorithm

QFSIN sets up and manages the zoning for the forward solution of $L^{T}X=L^{-1}I=Z$. Using the lower triangle, L, and an identity matrix I, the intermediate Z matrix is computed.

b. Input/Output

This subroutine uses the lower triangle decomposition of the stiffness matrix, residing on ML, to produce the Z matrix on MZ.

c. Error Messages

**** ERROR ____ ****

d. Subroutines Required

GETD M, PUTLAB, QFIN, QPASS

e. Argument List

ML - tape containing lower triangle

MZ - tape to receive Z matrix

M1 - scratch tape

M2 - scratch tape

f. Important Variables

See argument list

Subroutine QFIN

a. Algorithm

QFIN actually computes the forward solution of $L^T X = L^{-1} I = Z$. It is called by QFSIN every time that the available core must be zoned.

b. Input/Output

This subroutine uses the lower triangular decomposition of the total stiffness matrix, residing on ML to produce the intermediate Z matrix on MZ.

c. Error Messages

None

d. Subroutines Required

GETROW UNPACK, PUTROW

e. Argument List

T - scratch arrays
Y - scratch arrays
Z - scratch arrays
M - number of rows in lower triangle
NU - zoning clues
L - zoning clues
NL - tape containing lower triangle
MZ - tape to contain the Z matrix
MI - scratch tape
MO - scratch tape

f. Important Variables

See argument list

Subroutine MULT

a. Algorithm

This subroutine multiplies together two matrices , neither of which fit in core storage, to form a new matrix C = [A][B]. One row of A is read into core at a time, and as much of the B matrix as will fit is read in. The subroutine becomes rather slow if all of the B matrix will not fit. The subroutine MULT only controls the tape manipulation and storage. The actual multiplication is carried out in the called subroutine MMPY.

b. Input/Output

The subroutine reads in the A and B matrices and outputs the answer martix C.

c. Error Messages

**** DIMENSION ERROR ___ ****

d. Subroutines Required

GETROW, MMPY, GETDIM, PUTROW

e. Argument List

MAT1 - A matrix
MAT2 - B matrix
MATANS - C matrix, product of A and B
MTEMP1 - scratch tape
MTEMP2 - scratch tape
MATNAM - name of answer matrix C
IPRINT - 1 for intermediate output, zero for no intermediate output

f. Important Variables

See argument list

Subroutine ENMMPY

a. Algorithm

This subroutine multiplies together two matrices stored in core to produce the partial product of two larger matrices that do not fit in core. The multiplication is carried out in packed form. That is, strings of zeros are not multiplied explicitly. The storage for this subroutine is controlled by MULT.

b. Input/Output

This subroutine receives two matricies from the calling routine MULT and returns with the product matrix.

c. Error Messages

None

d. Subroutines Required

None

e. Argument List

BUFFER - array containing row of matrix
IBUFF - same as BUFFER but interpreted as fixed point
DBUFF - double precision answer matrix.

f. Important Variables

See argument list

Subroutine PUTLAB

a. Algorithm

PUTLAB is a subroutine which will put a matrix label on a programmer chosen data set.

b. Input/Output

The subroutine receives from the calling routine the matrix name, and size, and the unit number. If NAMLST is set to 1, the program will print out this same information.

c. Error Messages

None

d. Subroutines Required

FILTAP

e. Argument List

NTAPE - unit which will hold the matrix
MATNAM - matrix name
JROWS - number of rows of the matrix
JCOLS - number of columns of the matrix

f. Important Variables

See argument list

Subroutine PUTROW

a. Algorithm

This routine will put out a row of a matrix on the unit specified and in the format designated by IPACK. The row consists of ICOUNT consecutive elements.

b. Input/Output

The subroutine receives a row of a matrix from the calling routine and writes it on the specified unit in either packed or unpacked form.

c. Error Messages

None

d. Subroutines Required

PACK, UNPACK

e. Argument List

NTAPE - unit which will hold the row of the matrix
IPACK - if set to 1, put out packed row
BUFFER - array holding row to be written on tape
ICOUNT - number of elements to be written; if 0 or -1, write end of file

f. Important Variables

See argument list

Subroutine GETDIM

a. Algorithm

This subroutine will get the label of a matrix from the specified data set and if NAMLST is set to 1, print out the label information.

b. Input/Output

Using the information supplied by the calling routine this subroutine will position the tape and read the matrix label from the specified data set.

c. Error Messages

None

d. Subroutines Required

FILTAP

e. Argument List

MATRX - tape unit to be read for label
MATNAM - matrix name
KROW - rows in the matrix
KCOL - columns in the matrix

f. Important Variables

See argument list

Subroutine GETROW

a. Algorithm

This subroutine will obtain the next row of the matrix on the specified unit and store the row in a BUFFER.

b. Input/Output

This program reads a row of the matrix from the specified unit for use by the calling routine.

c. Error Messages

None

d. Subroutines Required

PACK, UNPACK

e. Argument List

NTAPE - tape unit number from which the row of the matrix is to be read
IPACK - 0, return row in packed form
-1, return row in unpacked form
BUFFER - array to contain a row of the matrix
ICOUNT - number of words in the row

f. Important Variables

See argument list

Subroutine PACK

a. Algorithm

This subroutine is used to pack rows of a matrix so they may be written on a data set in an efficient manner. This is done by representing strings of zeroes by a single fixed point negative integer where the value of the integer represents the number of zeroes in the string. Non-zero numbers are preceeded by a fixed point number indicating the number of non-zero numbers that follow. A single zero in a string is represented explicitly. For example given the following row of a matrix:

0.,0.,1.,2.,0.,5.,7.,0.,0.,0.,0.,0.,0.,

it would be packed to become

-2,5,1.,2.,0.,5.,7.,-6

b. Input/Output

The subroutine receives a row of a matrix and returns to the calling routine a packed row.

c. Error Messages

None

d. Subroutines Required

None

e. Argument List

IFIRST - row to be packed

ICOUNT - number of elements in the row to be packed. In the example this would be 13.

IOUTPT - packed row returned to calling routine

IOUTCT - number of elements in the packed row returned to calling routine. In the example this would be 8.

IWORD - value of first word in packed row; in the example this would be -2

f. Important Variables

See argument list

Subroutine UNPACK

a. Algorithm

This subroutine unpacks rows of matrices that have been packed by the PACK subroutine.

b. Input/Output

Using a packed row of a matrix, the subroutine generates a row of a matrix with explicit zeroes.

c. Error Messages

None

d. Subroutines Required

None

e. Argument List

IFIRST - row of the matrix to be unpacked
ICOUNT - length of the packed row
IOUTPT - unpacked row of the matrix returned to the calling routine
IOUTLT - length of unpacked row
NLZERS - number of leading zeroes

f. Important Variables

See argument list

Subroutine PRINT2

a. Algorithm

PRINT2 will print out all element stiffness and stress arrays when the IDEBUG clue is set to 1.

b. Input/Output

This subroutine prints out the element stiffness or stress matrix CCC in wallpaper output format.

c. Error Messages

None

d. Subroutines Required

None

e. Argument List

CCC - matrix to be printed out
ANAME - matrix name (AKG or STG)
BNAME - matrix name (AKG or STG)
INAME - member number
II - row dimension of actual matrix to be outputted
JJ - column dimension of actual matrix to be outputted
IWALL - number of columns of matrix to be printed across page
ISWTCH - clue to indicate wallpaper printout
JR - row dimension of CCC
JC - column dimension of CCC

f. Important Variables

See argument list

Subroutine BOUND

a. Algorithm

This subroutine processes the condensed boundary conditions and forms the matrix IBC. This matrix is then passed to the subroutine GEOBC where the boundary conditions are matched against the geometry.

b. Input/Output

This subroutine reads the condensed boundary conditions and passes the boundary condition array to the subroutine GEOBC.

c. Error Messages

*** ERROR MESSAGE *** NUMBER OF JOINTS SPECIFIED (____) IS TOO LARGE

*** ERROR MESSAGE *** JOINT NUMBER ____ IS GREATER THAN SIZE OF MATRIX

d. Subroutines Required

GEOBC

e. Argument List

None

f. Important Variables

IBC - boundary condition array

Subroutine DEFCON

a. Algorithm

This subroutine reads the deflection constraints and forms a deflection constraint matrix on tape NTAPDC. The constraints are checked to see if they are admissable and, finally, the constraints are printed out with an appropriate heading.

b. Input/Output

The data block containing the deflection constraints is read. After reading and checking, they are printed.

c. Error Messages

*** ERROR **** DEFLECTION CONSTRAINT PLACED ON A FIXED DEGREE OF FREEDOM - NODE ____ COMPONENT ___

d. Subroutines Required

PUTLAB

e. Argument List

None

f. Important Variables

NTAPDC - tape to contain deflection constraints
LDOF - degree of freedom number

Subroutine DSCALE

a. Algorithm

This subroutine scales the deflections by multiplying the deflections of the feasible design by the current value of the constraint-relaxation factor and then checks the constraints against the allowable deflections. If the scaled deflections exceed the allowable deflection, the difference is printed out and written, together with the degree of freedom and load case information, on tape NTAPDD.

b. Input/Output

The subroutine matches the deflections on NTAPD to the deflection constraints on NTAPDC and writes the violated constraint differences on NTAPDD.

c. Error Messages

None

d. Subroutines Required

PUTLAB, GETDIM, GETROW

e. Argument List

None

f. Important Variables

LC - load case
NVDOF - number of violated degrees of freedom
DIFF - difference between allowable and actual deflection

Subroutine RESIZE

a. Algorithm

This is the main subroutine for resizing,using the nodal stress method. The subroutine starts by reading the array NTABD and NTABR from tape or creating them if the subroutine is being executed for the first time. These arrays are used for determining how the corner forces from the corner force tape should be summed to create the nodal forces. In the small core version of the program (7094) this information is generated and then stored in the member pseudomatrix in order to be able to equivalence the NTABD array with the TFORCE array. After the creation of these tables, the member tape is rewound and the first row of the member pseudomatrix is read. Using the information in the member data to control the reading of the corner force tape, the corner forces are entered into the array TFORCE thru the subroutine FCAPG and the shear flows are written on a scratch tape. There is a separate section of code for each of the element types in the structure. This subroutine also controls the computation and printing of the shear flows and beam moments.

b. Input/Output

This subroutine uses the member pseudomatrix, the nodal geometry and the corner force matrix to generate the nodal forces for use in NUSIZE.

c. Error Messages

ERROR AT MEMBER DATA LABEL
ERROR AT MEMBER FORCE LABEL
ERROR READING MEMBER DATA MATRIX
COMAP. INTERFACE ERROR RETURN
ERROR AT COORDINATE DATA LABEL
ERROR READING COORDINATE MATRIX

d. Subroutines Required

PUTLAB, GETDIM, GETROW, FILTAP, FCAPG, PUTROW, RTAPE, LENTH, SFTAPE, NUSIZE
TABDG, PAGES, CRINT

e. Argument List

None

f. Important Variables

MEM - member number
NTYP - member type
NI thru NL - nodes of the element
BAREA - area of bars or beams
THICK - thickness of planar elements
E - Young's modulus
NTABD - array containing the topology table
XMEM 81 - tensile allowable stress
XMEM 82 - compressive allowable stress
KLUT - clue to control searches for cap forces, shear flows, bending moments, and warp loads.
NJTS - number of joints in the structure

Subroutine TABDG

a. Algorithm

This subroutine generates the force direction table for cap forces or warp loads. For each node in the structure, the nodes connected to it are placed in the array NTABD. Up to twenty nodes may be connected to each node.

b. Input/Output

All input and output is done thru common.

c. Error Messages

*** IN SEARCH CYCLE (KLUT) = ____ FOR MEMBER NO. ____ 20 COLUMNS EXCEEDED IN TABLE FOR DIRECTION ____ TO ____

d. Subroutines Required

None

e. Argumert List

None

f. Important Variables

NNOD - node we are at

NDIR - node we are going to

Subroutine CRINT

a. Algorithm

This subroutine prints out the cap and warp loads and the shear flows at the end of the optimization procedure.

b. Input/Output

Depending on the input value of KLUT, the subroutine prints the proper headings and values for the cap forces and shear flows.

c. Error Messages

None

d. Subroutines Required

None

e. Argument List

None

f. Important Variables

KLUT - less than 5, print out cap forces
equal to 6, print out shear flows
equal to 7, print out warp loads

Subroutine PAGES

a. Algorithm

This subroutine places a date and page number at the top of a page during the printing out of the cap forces and shear flows.

b. Input/Output

See Algorithm

c. Error Messages

None

d. Subroutines Required

None

e. Argument List

None

f. Important Variables

None

Subroutine FCAPG

a. Algorithm

This subroutine sums cap and warp loads and controls the resizing of the structure by zones. The subroutine receives the cap force and, from the NTABD and NTABR arrays, computes the row number in TFORCE into which the force should be summed. Because the cap force from the corner force matrix is for unit thickness, it is first multiplied by the element thickness. The row number into TFORCE is checked to see if that protion of TFORCE is in core. If it is not in core, the cap force is held in a local array and a previous zone of the structure is resized, using scratch member and shear flow tapes containing the members and shear flows that go with the portion of the structure in the zone being resized. Then the rows of TFORCE in core are moved up in the array and the previous zone of TFORCE is discarded. The summary of cap forces then proceed.

b. Input/Output

The subroutine receives the corner force row and places it in TFORCE. Also the scratch member matrix QM, and shear flow matrix QS, is formed.

c. Error Messages

None

d. Subroutines Required

PUTLAB, CRINT, NUSIZE

e. Argument List

None

f. Important Variables

KR - row number in TFORCE
QM - scratch member matrix tape number
QS - scratch shear flow tape number
TF - array containing geometric data associated with TFORCE
NLC - number of load conditions

Subroutine RTAPE

a. Algorithm

RTAPE reads rows of the corner force matrix. If the row being read represents a shear flow, it is written on tape QS. The number of rows read for each call to RTAPE is determined by the input value of NREAD.

b. Input/Output

The subroutine uses the input value of NREAD to determine how many rows of the corner force matrix should be read. The last row read is passed back to the calling subroutine

c. Error Messages

*** END OF FORCE MATRIX REACHED WHEN SEARCH CYCLE (KLUT) = ____ FOR MEMBER ____ TYPE ____.

COMAP INT. ERROR RETURN FROM SUB. RTAPE

d. Subroutines Required

GETROW, PUTROW

e. Argument List

None

f. Important Variables

NREAD - number of rows of corner force matrix to be read.

QS - scratch shear flow matrix

Subroutine WEIGHT

a. Algorithm

This subroutine computes the weight of the individual members in the structure.

b. Input/Output

The node points and type of element is read in and the weight of the element for the old thickness and for the new thickness is returned to the calling routine.

c. Error Messages

None

d. Subroutine Required

None

e. Argument List

None

f. Important Variables

WNEW - weight of element for new thickness
WOLD - weight of element for old thickness
DEN - density of element material

Subroutine LENGTH

a. Algorithm

This subroutine computes the perpendicular length from the line NI- NJ to the node NK by forming the cross product of the vectors (NJ - NI) and (NJ - NK) and dividing by the length of (NJ - NI).

b. Input/Output

The subroutine receives the three node points as input and returns the perpindicular length to the calling routine.

c. Error Messages

None

d. Subroutines Required

None

e. Argument List

NI, NJ - nodes on the line from which the perpendicular distance is to be found

NK - the node off the line - the perdendicular distance is from this node to the line

H - perpendicular distance

COORD - coordinate array

f. Important Variables

See argument list

Subroutine ORTHOG

a. Algorithm

This subroutine converts non-orthogonal stresses to orthogonal stresses.

b. Input/Output

The subroutine receives the non-orthogonal stresses and the angle between the direct stresses and returns the orthogonal stresses.

c. Error Messages

None

d. Subroutines Required

None

e. Argument List

XN1 - stress in swept direction
XN2 - stress in direction at angle to XN1
XN3 - non-orthogonal shear stress
BETA - angle between XN1 and XN2

f. Important Variables

See argument List

Subroutine NUSIZE

a. Algorithm

This subroutine computes the new sizes of the elements by using the nodal forces in TFORCE, the shear flows on the scratch shear flow tape, the allowable yield stresses,and the instability tables. The elements are read from the member pseudo matrix one row at a time. The program then checks the member type and transfers to the proper section of code to resize that type of element. At each node, using the nodal forces in TFORCE and the shear flow from the scratch shear flow tape, the orthogonal stresses at the corner are computed. These stresses and the allowable stresses are then used in the arithmetic assignment statement ERATIO to compute the stress ratio at the corner. The stress in the corner in the local element X direction is then checked against the instability table and the most critical is chosen for resizing the corner. All the corners/ends of the element are then averaged to obtain a new thickness. The ratio of old thickness to new thickness is use to define the stress ratio scale factor. The element stresses are printed out the last time thru the subroutine.

b. Input/Output

The subroutine creates a new member tape with the resized element thicknesses in place of the original thicknesses.

c. Error Messages

None

d. Subroutines Required

GETDIM, STABIL, WEIGHT, GETROW

e. Argument List

None

f. Important Variables

SUMOLD - running sum of the old structural weight

SUMNEW - running sum of the new structural weight

KSTAB - stability table number

DISTIJ - distance from I to J

POVERA - stress in a bar

GXY(M) - shear stress for load case M

H(M) - new thickness for a bending element

TF(I,1) - stiffness at a node in direction of force I (E times A)

TF(I,2) - sum of the lengths perpendicular to the force direction I of all the direct stress carrying elements contributing forces in direction I

TF(I,3) - ExA of all the bars and beams along force direction I

Subroutine NUREAD

a. Algorithm

This subroutine reads a given set of tables into a singly dimensional array of common storage. These tables are used for the calculation of allowable stress as ε function of applied stress and/or shear in subroutine DINTK.

b. Input/Output

This subroutine reads in the instability tables and forms common block INTERP.

c. Error Messages

None

d. Subroutines Required

None

e. Argument List

NUMTB - 1 for first call to NUREAD
K for replacing table K
MANDAN - 0 for initial read, 1 for table replacement
NG - error return, 0 if no errors
L1(K) - maximum number of X_i's in table K
L2(K) - maximum number of Y_i's in table K
NUMPTS - number of table entries preceeding table K

f. Important Variables

See argument list

Subroutine STABIL

a. Algorithm

This subroutine calls the proper instability table for the member being resized and returns the allowable stress to the calling routine.

b. Input/Output

The subroutine receives the table number and stress from the calling routine and returns the allowable stress.

c. Error Messages

***ERROR IN INTERPOLATION ROUTINE - NG = ____

d. Subroutines Required

DINTK

e. Argument List

NUMTBL - table number
ARG1 - abscissa of instability table
ARG2 - shear flow in two dimensional tables
FCT - value of the allowable stress

f. Important Variables

See argument list

Subroutine DINTK

a. Algorithm

This subroutine performs table look ups and linear interpolations for functions of one and two variables.

b. Input/Output

The values of the independant variables and the table number are supplied by the calling routine. The allowable stress is returned.

c. Error Messages

None

d. Subroutines Required

None

e. Argument List

L1 - number of x_i's in table
L2 - number of y_i's in table
NUMPTS - number of table entries preceding table K
KODE - dummy array
N1H1B4 - dummy array
N2H1B4 - dummy array
ARG1 - x
ARG2 - y
NUMTBL - number of first table of the tabular system to which x and y are assigned
L3 - number of functions for which values are desired, usually 1
FCT - the interpolated value returned to the calling routine
NG - error return, should be 0, set to 3 if the function was off the table

f. Important Variables

See argument list

Subroutine UNITWT

a. Algorithm

This subroutine computes the weight of the individual member of the structure, for unit value of the design parameter. The value is put on the tape along with the element stiffness matrix, and is used in DLIMIT. The code in this subroutine is similar to that in subroutine WEIGHT.

b. Input/Output

The node points and type of element is read in and the weight of the element for unit thickness is returned to the calling routine.

c. Error Messages

None

d. Subroutines Required

None

e. Argument List

None

f. Important Variables

None

Subroutine DINTK

a. Algorithm

This subroutine performs table look ups and linear interpolations for functions of one and two variables.

b. Input/Output

The values of the independant variables and the table number are supplied by the calling routine. The allowable stress is returned.

c. Error Messages

None

d. Subroutines Required

None

e. Argument List

L1 - number of x_1's in table
L2 - number of y_1's in table
NUMPTS - number of table entries preceding table K
KODE - dummy array
N1H1B4 - dummy array
N2H1B4 - dummy array
ARG1 - x
ARG2 - y
NUMTBL - number of first table of the tabular system to which x and y are assigned
L3 - number of functions for which values are desired, usually 1
FCT - the interpolated value returned to the calling routine
NG - error return, should be 0, set to 3 if the function was off the table

f. Important Variables

See argument list

Subroutine UNITWT

a. Algorithm

This subroutine computes the weight of the individual member of the structure, for unit value of the design parameter. The value is put on the tape along with the element stiffness matrix, and is used in DLIMIT. The code in this subroutine is similar to that in subroutine WEIGHT.

b. Input/Output

The node points and type of element is read in and the weight of the element for unit thickness is returned to the calling routine.

c. Error Messages

None

d. Subroutines Required

None

e. Argument List

None

f. Important Variables

None

c. Error Messages

None

d. Subroutines Required

GETROW, PUTROW, GETDIM, PUTDIM

e. Argument List

None

f. Important Variables

NSTART - starting location for the stacking of the element stiffness matricies into the global stiffness matrix

NGO - size of the blocks of the element stiffness matrices that are to be stacked into the global stiffness matrix starting at NSTANT

D - nodal deflections

SFO - old scale factor needed to move design to the boundary of the feasible space

SCFAC - constraint relaxation factor

XKINV - element stiffness matrix for unit thickness

DELMAX - average area change for each member

Subroutine DRATIO

a. Algorithm

This subroutine reads the deflections from tape NTAPD and the deflection constraints from tape NTAPDC. For those deflections that are constrained, the ratio of actual deflection to allowable deflection is formed. The largest ratio is saved, and becomes the deflection constraint ratio, for use in determining the scale factor needed to reach a feasible design.

b. Input/Output

The deflection and deflection constraints are read in and the deflection constraint ratio is returned to the calling program.

c. Error Messages

None

d. Subroutines Required

GETDIM, GETROW

e. Argument List

None

f. Important Variables

DRATN - new deflection constraint ratio
CMA - allowable maximum deflection
CM1A - allowable minimum deflection

Subroutine EL15

a. Algorithm

This routine generates stiffness and stress matrices in global coordinates for the triangular plate bending element. Entry point PLTRI generates the local stiffness which is passed back to the quad routine (EL16).

b. Input/Output

Same as in EL1

c. Error Message

*** MEMBER NUMBER ____ HAS ____ EQUAL TO ZERO ***

d. Subroutines Required

PRINT2, SPLITS

e. Argument List

None

f. Important Variables

AKG, STG, COORD, BETA, CT - same as in EL3

AX2, AL23, AL13, A11 A22, A33 ... - same as in EL3

A1, A2, A3, B1, B2, B3 - projections of triangle's sides onto its local x and y axes - used in conjunction with the area coordinate formulation

T - Matrix relating nodal displacements to curvatures within a subelement.

Subroutine EL16

a. Algorithm

This routine generates element matrices for the quadrilateral bending element. It calls PLTRI four times to assemble four triangles into a quad with interior degrees of freedom included. These are subsequently condensed out of the local stiffness matrix before transformation to global coordinates.

b. Input/Output

Same as in EL1

c. Error Messages

Same as in EL15.

d. Subroutines Required

PLTRI (E.P.), SREVN2, PRINT2, SPLITS

e. Argument List

None

f. Important Variables

AKG, STG, COORD, BETA, CT - same as in EL3
XCG, YCG - local coordinates of element's centroid.

Subroutine DLIMIT

a. Algorithm

DLIMIT performs the calculations necessary to form the new member matrix in the deflection constraint mode. The first half of the subroutine forms $\partial\delta_j/\partial A_i = [K]^{-1}k_i\delta$ in two steps. First, up to 1500 rows of the deflection matrix are read into core in the array called D. Next, the element stiffness matricies are read in one at a time along with the stacking indicies and unit weight for each member. The deflection matrix is then pre-multiplied by the element stiffness matrix in packed form. If at any time, the deflections that are needed to perform this multiplication are not in core, an additional 750 rows are read in and the first half of those currently in core are discarded. As the product $k_i\delta$ is formed for each member it is written out on to a scratch data set. After the last product has been formed, the first violated degree of freedom is read from a scratch data set; and the corresponding row of the inverse of the stiffness matrix from a second data set. This row is now scaled and post multiplied by the result of $k_i\delta$, for all the members in the structure. The result is called DERV(NM,M). NM is the member number and M is the load condition. This quantity is really $\partial\delta/\partial A_i$. It is next converted to $\partial\delta/\partial W$ by dividing by the weight for unit thickness. The changes in thickness for the various members are then calculated and the sum of the changes in each member for the various load conditions, with violated degrees of freedom are added together This sum is then divided by the number of load conditions with violated degrees of freedom; resulting in the average change in each member. This average change is then added to the original area to produce a new member thickness.

b. Input/Output

This subroutine uses as input the member matrix on unit NTAPM, the deflection matrix on unit NTAPD, the pertinent rows of the stiffness matrix inverse on unit NTAPIS, the deflection differences on unit MTADDD, and the element stiffness.

c. Error Messages

None

d. Subroutine Required

GETROW, PUTROW, GETDIM, PUTDIM

e. Argument List

None

f. Important Variables

NSTART - starting location for the stacking of the element stiffness matrices into the global stiffness matrix

NGO - size of the blocks of the element stiffness matrices that are to be stacked into the global stiffness matrix starting at NSTART

D - nodal deflections

SFO - old scale factor needed to move design to the boundary of the feasible space

SCFAC - constraint relaxation factor

XKINV - element stiffness matrix for unit thickness

DELMAX - average area change for each member

Subroutine INVERS

a. Algorithm

INVERS calls all the routines which are involved in the computation of the inverse of the total stiffness matrix. It takes advantage of the fact that the lower triangle decomposition of the stiffness matrix is saved from the previous stress analysis.

b. Input/Output

This subroutine uses the lower triangle to create the rows of the inverse up to and including, the lower degree of freedom that has constraints placed on it. It stores only those rows that correspond to constrained degrees of freedom.

c. Error Messages

None

d. Subroutines Required

SECOND, QFSIN, REVERS, QBISOL

e. Argument List

None

f. Important Variables

A - scratch array used by all solution routines.

NTAPLT - tape containing lower triangle

NTAPIS - tape containing needed rows of the inverse

Subroutine QBISOL

a. Algorithm

QBISOL solves the equation $L^T X = Z$ to obtain X which is the inverse of the total stiffness matrix. In general, QBISOL only manages the zoning of the solution; the actual computations are done in QBAC.

b. Input/Output

This subroutine uses the Z matrix on MZ and the lower triangular matrix on MB, both in reverse order, and produces the required rows of the inverse in reverse order on MX.

c. Error Messages

**** ERROR ___ ****

d. Subroutines Required

GETDIM, PUTLAB, QIBAC

e. Argument List

MB - tape containing lower triangle

MX - tape containing Z matrix

MX - tape to contain needed rows of inverse

M1 - scratch tapes

M2 - scratch tapes

f. Important Variables

See argument list

Subroutine QIBAC

a. Algorithm

QIBAC is called by QBISOL and performs the same function in obtaining the inverse as QBAC does in calculating the nodal deflections. The one exception lies in the fact that only those rows of the inverse, up to and including the lowest degree of freedom that has constraints placed on it, are calculated and only the rows corresponding to constrained degrees of freedom are saved in reverse order on MX.

b. Input/Output

This subroutine uses the intermediate Z matrix on MZ and the lower triangular decomposition of the total stiffness matrix on MB, both in reverse order, to produce the required rows of the inverse in reverse order.

c. Error Messages

None

d. Subroutines Required

GETROW, UNPACK, HOTDOT, PUTROW

e. Argument List

T - scratch arrays
B - scratch arrays
X - scratch arrays
L - scratch arrays
M - number of rows in lower triangle
N - number of columns in Z matrix (M = N)
MU - zoning clue
KORE - work area
MB - tape to contain needed rows of inverse in reverse order
MI - scratch tape
MØ - scratch tapes

e. Important Variables

See argument list

Section 5

Programmer's Information for Dynamic Optimization Program

Introduction

This program, for minimum weight design of dynamically loaded shells, contains four main subprograms:

SABRE - finds mass and stiffness matrices
DRAST - finds displacements for dynamic solution
MAXST - finds maximum stress for each element
ØPTMIZ- calculates new thicknesses

This programmer's section explains the revised program. Descriptions of the calling sequence, tape usage, array usage, program counters and clue definitions, as well as flow charts of selected routines, are presented. References from the SABØR 3A-DRASTIC programs should be used concurrently with this section since some routines have undergone only limited modification from their original form.

Program Variables and Corresponding Engineering Symbols

RØ	-	ρ	material mass density
R1	-	r_1	radius at station one
R2	-	r_2	radius at station two
Z1	-	Z_1	height at station one
Z2	-	Z_2	height at station two
ALY	-	s	slant height
E1	-	E	Young's modulus
GNU1	-	ν	Poisson's ratio
T0	-	t_o	initial time of integration
T1	-	t_f	final time of integration
DT	-	Δt	integration time step
ES1	-	ϵ_{S1}	meridional strain at station one
ES2	-	ϵ_{S2}	meridional strain at station two
ET1	-	$\epsilon_{\theta 1}$	circumferential strain at station one
ET2	-	$\epsilon_{\theta 2}$	circumferential strain at station two

EST1	-	$\epsilon_{S\theta 1}$	shear strain at station one
EST2	-	$\epsilon_{S\theta 2}$	shear strain at station two
CS1	-	χ_{S1}	meridional rotation at station one
CS2	-	χ_{S2}	meridional rotation at station two
CT1	-	$\chi_{\theta 1}$	circumferential rotation at station one
CT2	-	$\chi_{\theta 2}$	circumferential rotation at station two
CST1	-	$\chi_{S\theta 1}$	twist rotation at station one
CST2	-	$\chi_{S\theta 2}$	twist rotation at station two

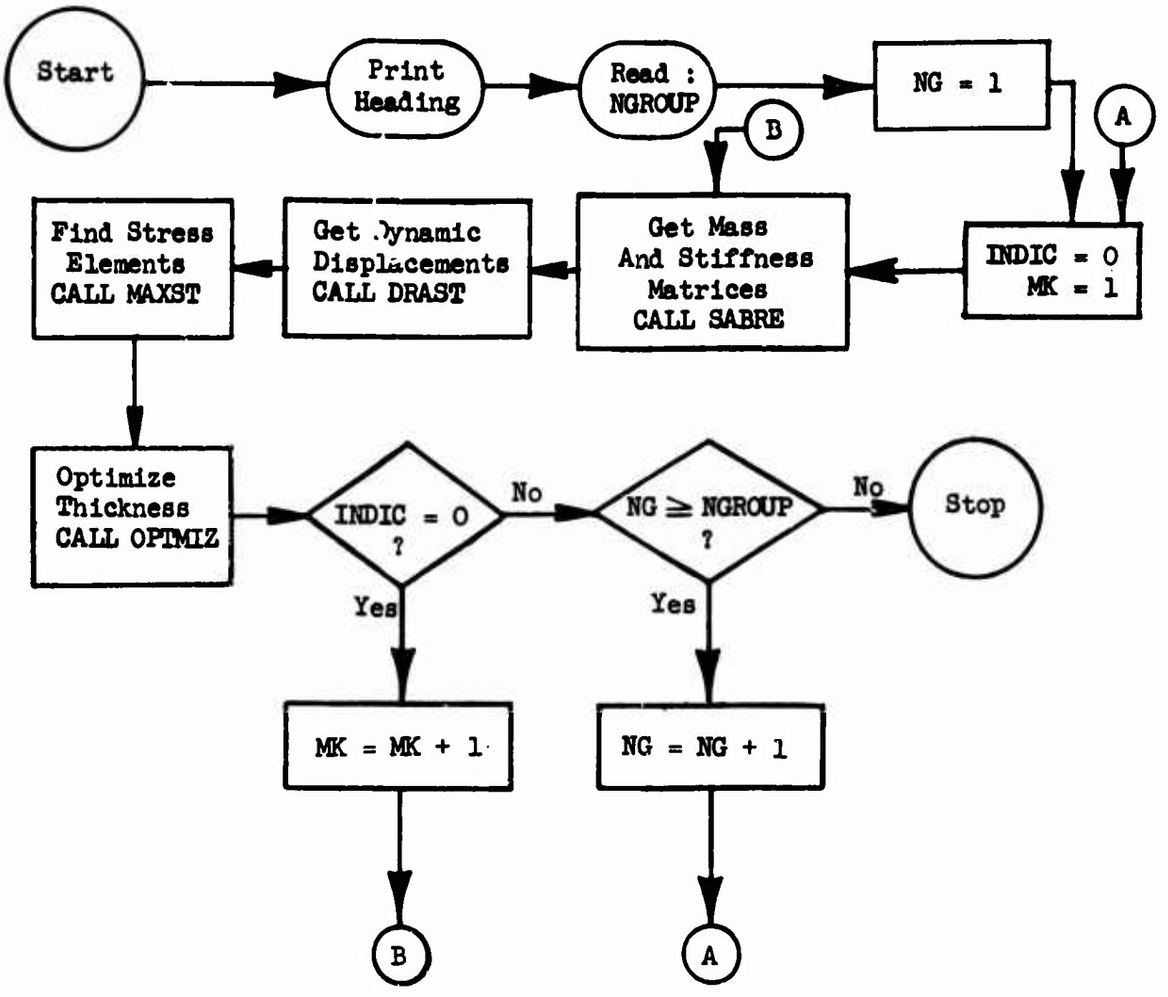
Calling Sequence and Flow Charts

The following array indicates the program's subroutine calling sequence.

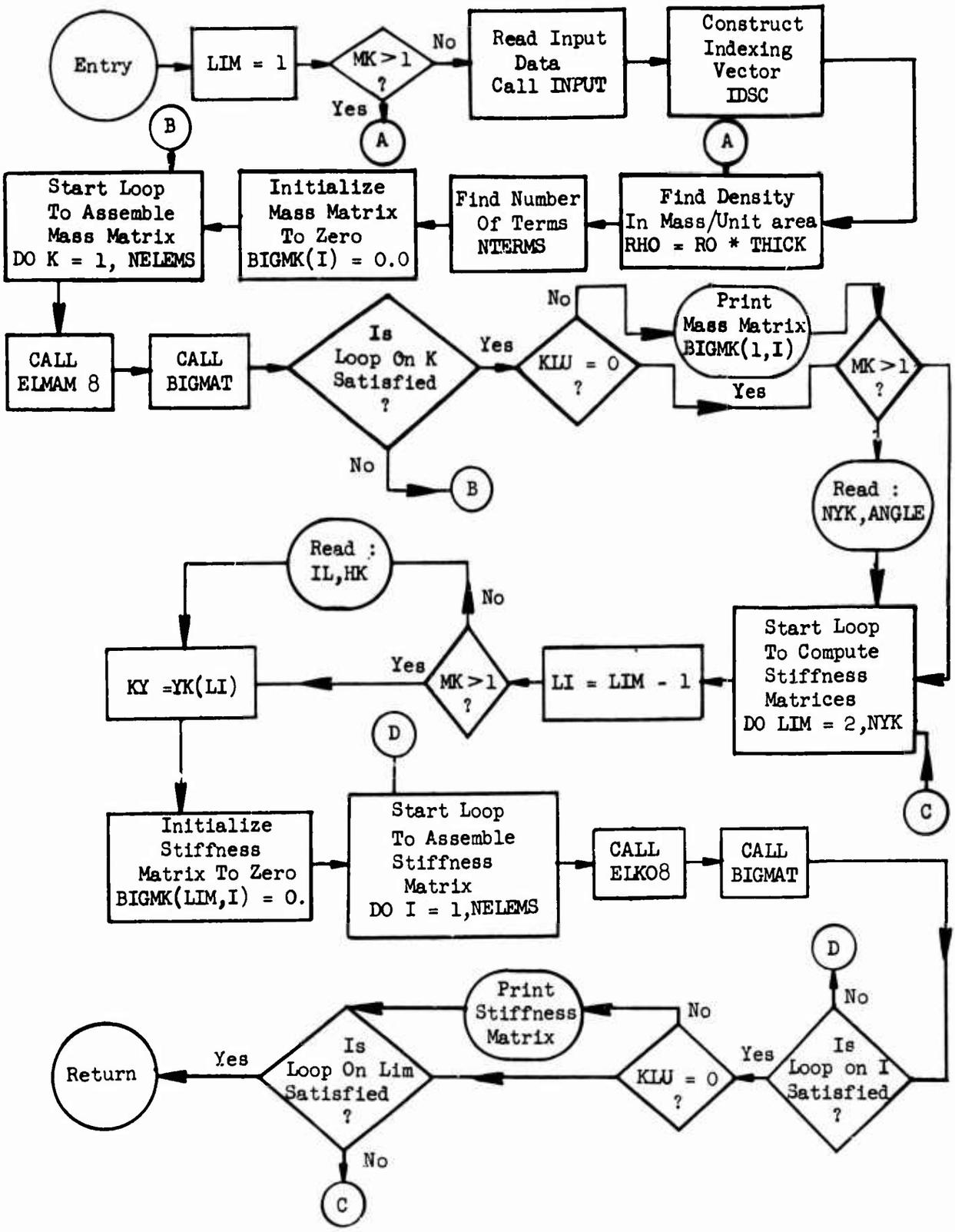
```
MAIN
     SABRE
          INPUT
          ELMAM8
                MMPRT3
          BIGMAT
          ELKO8
                ELEMTA
                    FUNCTI
                MATL12
                MMPLT3
                MMPRT3
          BIGMAT
     DRAST
        INHARM
        START
                BØUND
                FACTØR
        BETAIN
                FØRCE
                SØLTRS
                STRESS
                PRØDUC
        PUNCHY
     MAXST
        SIGMA
     ØPTMIZ
```

Details on the flow of operations and the transfers of control between the main program and the four subprograms are contained in the following flow charts.

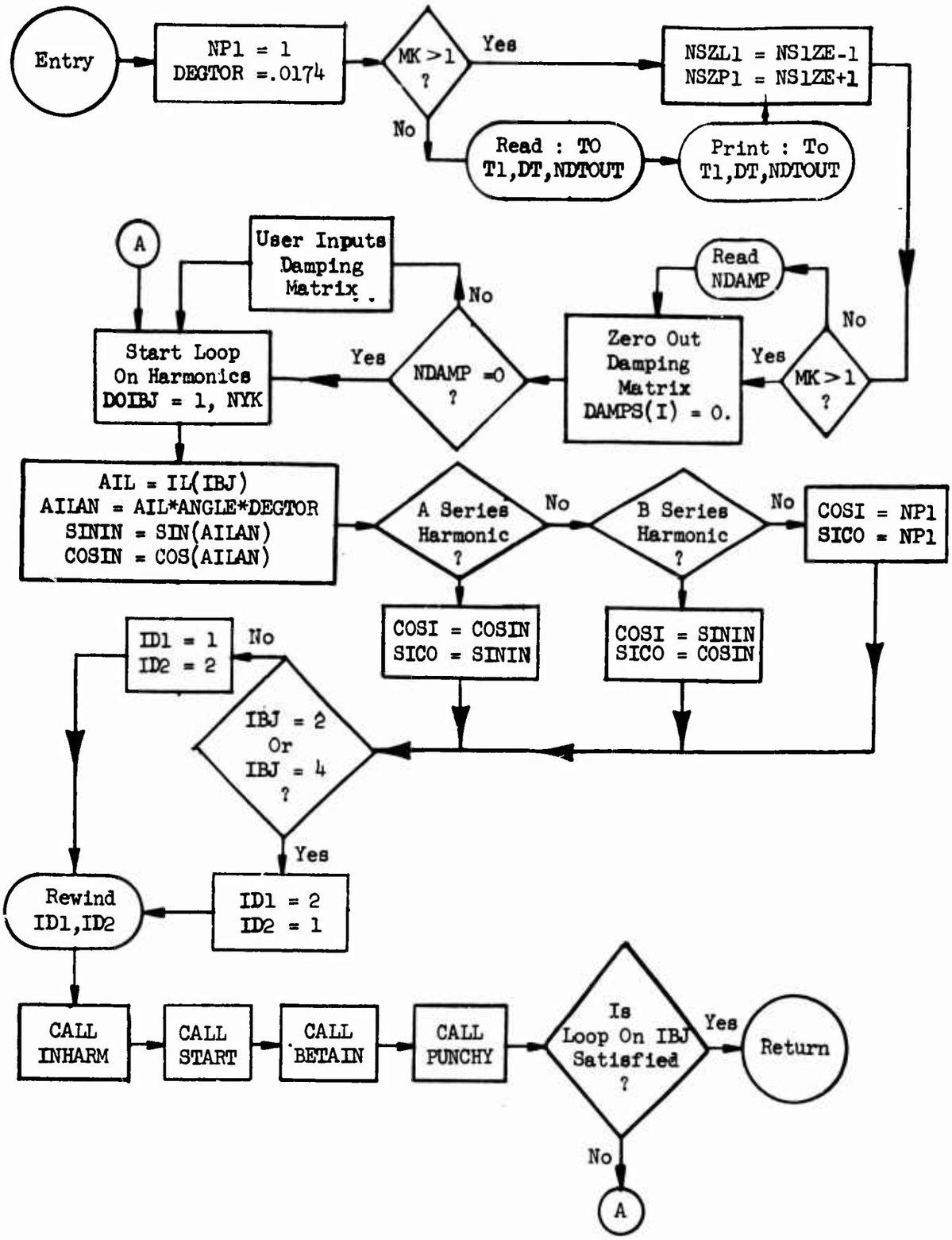
SUBROUTINE MAIN



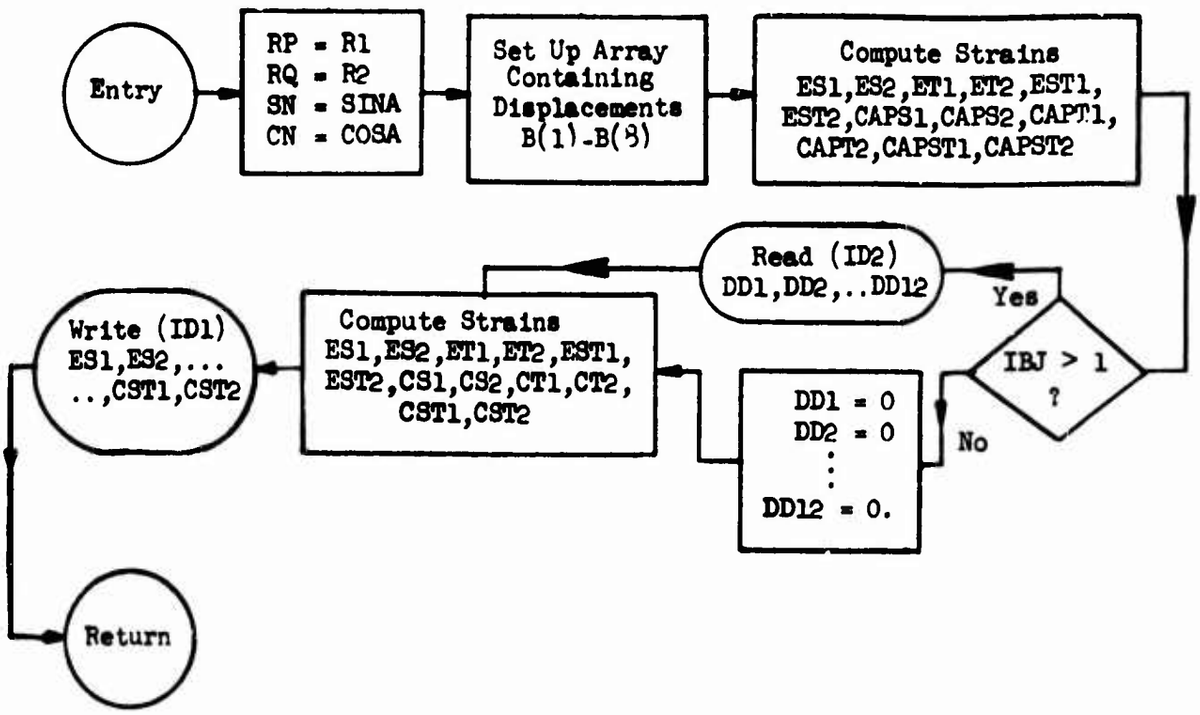
SUBROUTINE SABRE



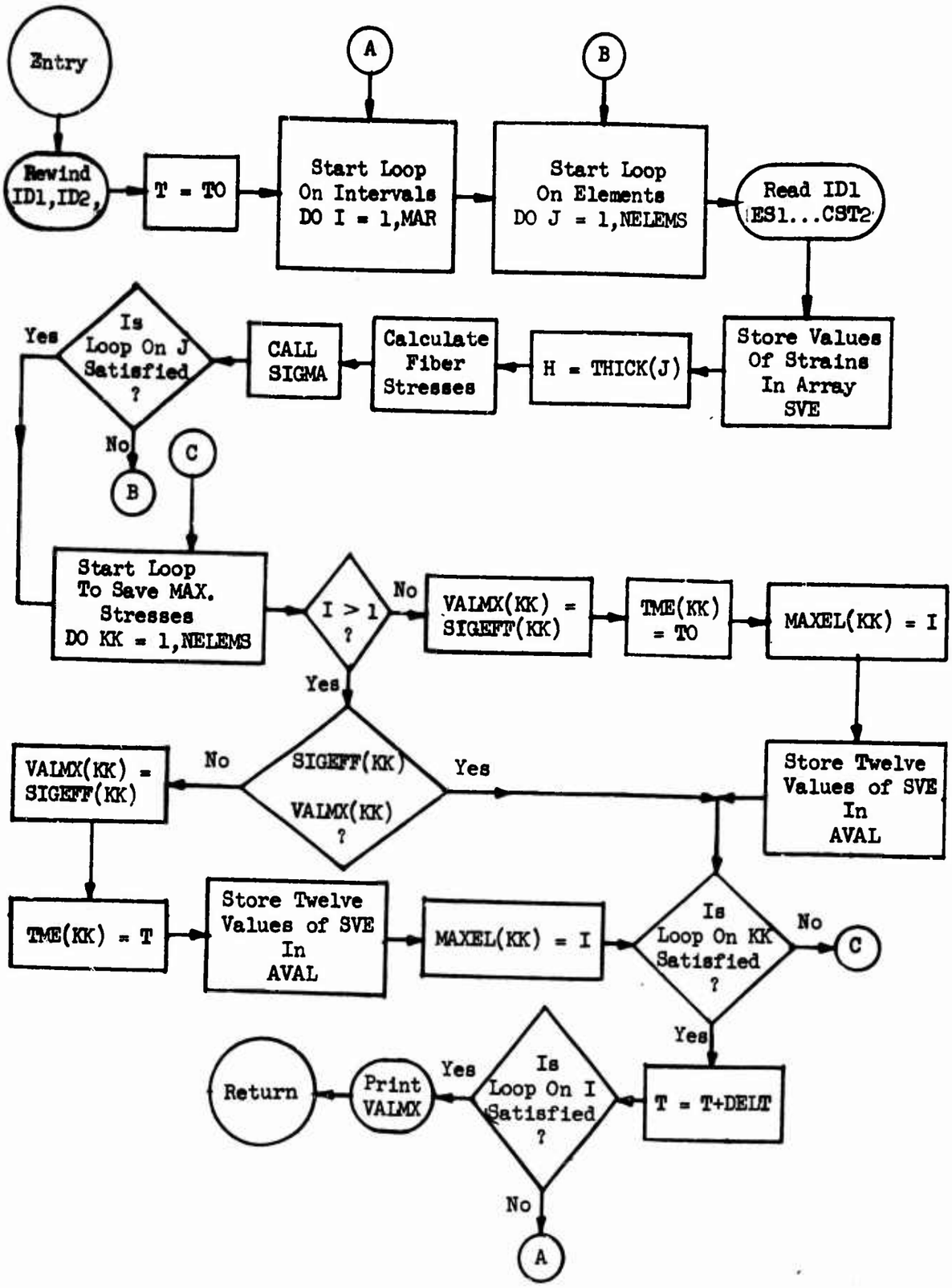
SUBROUTINE DRAST



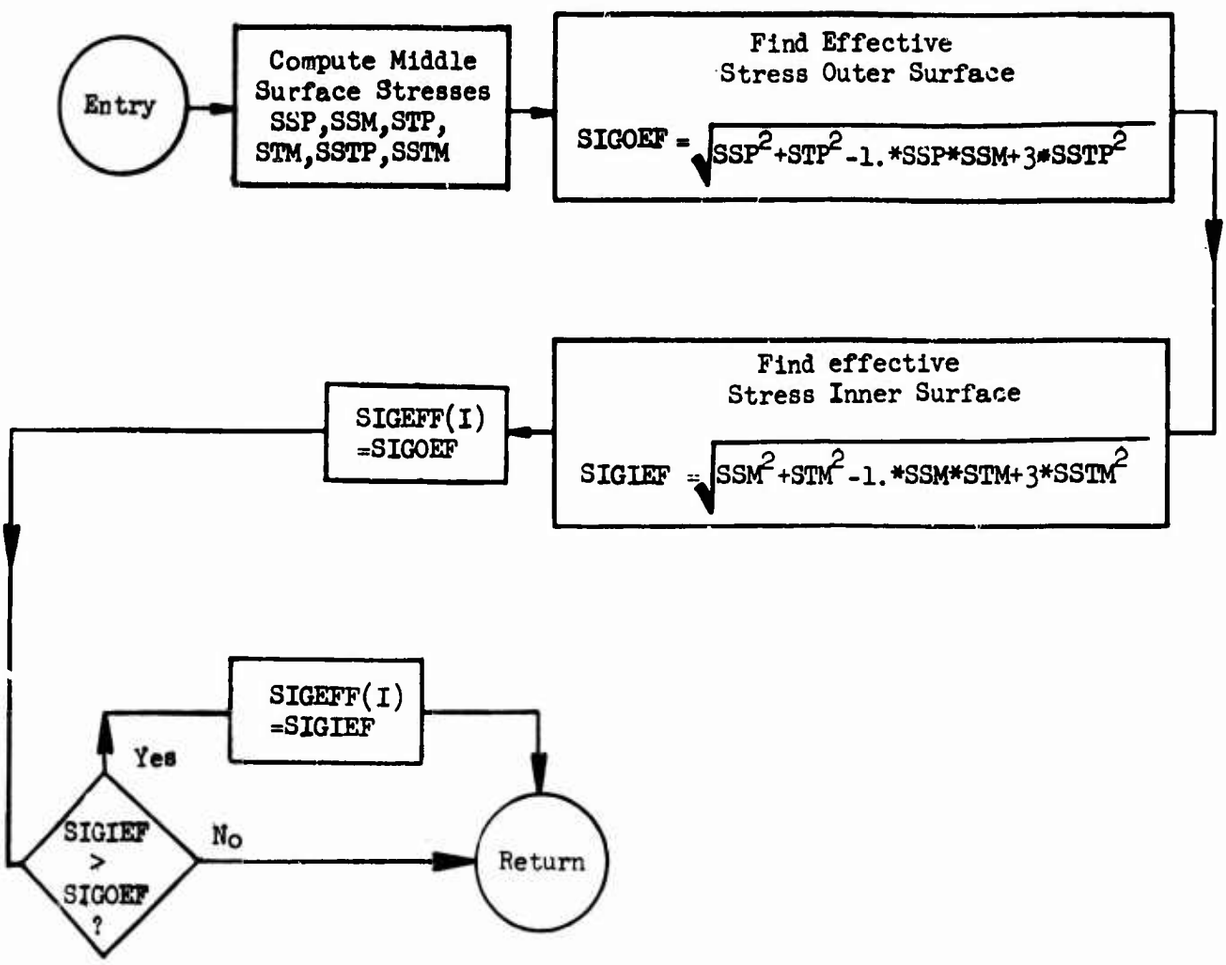
SUBROUTINE STRESS



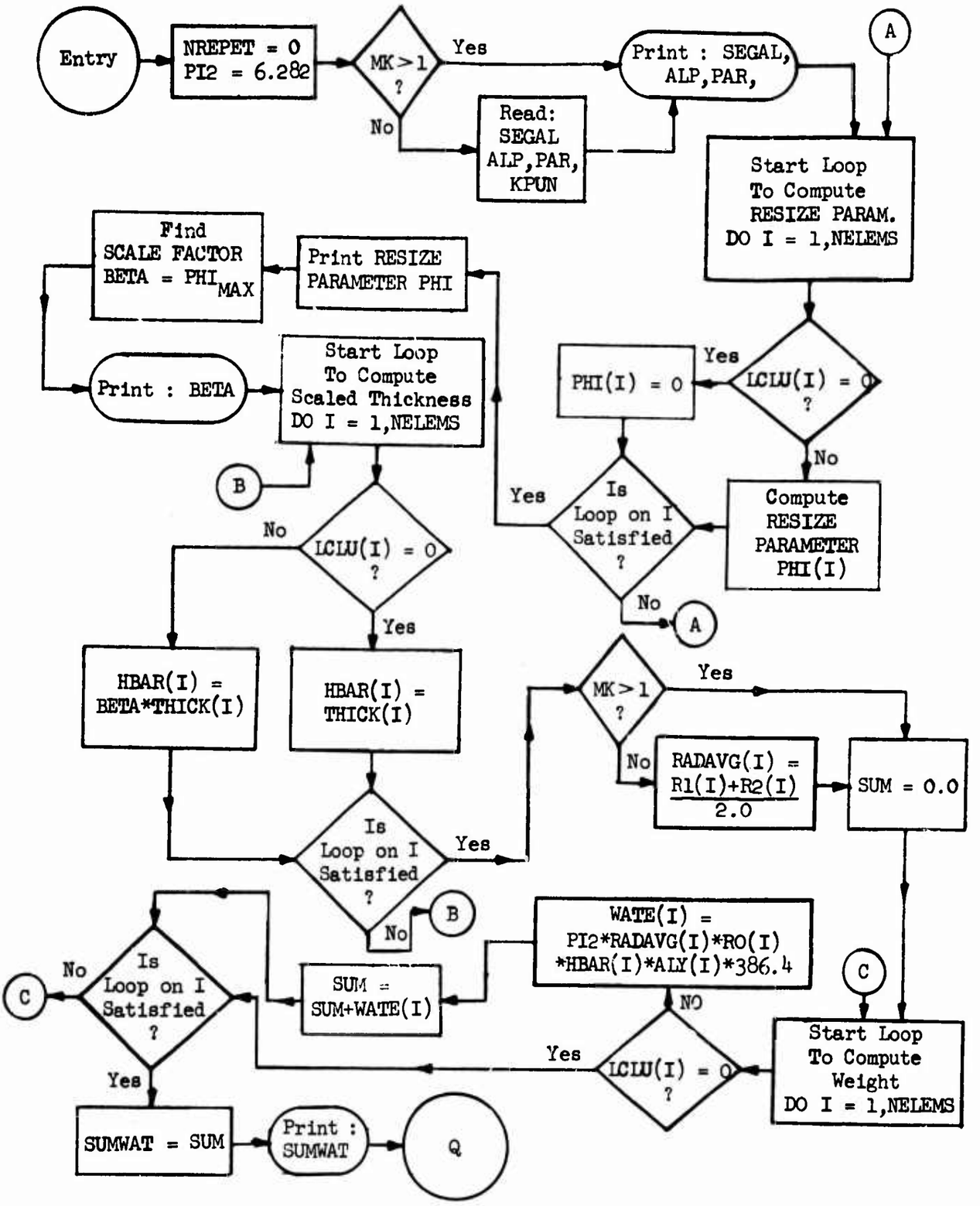
SUBROUTINE MAXST



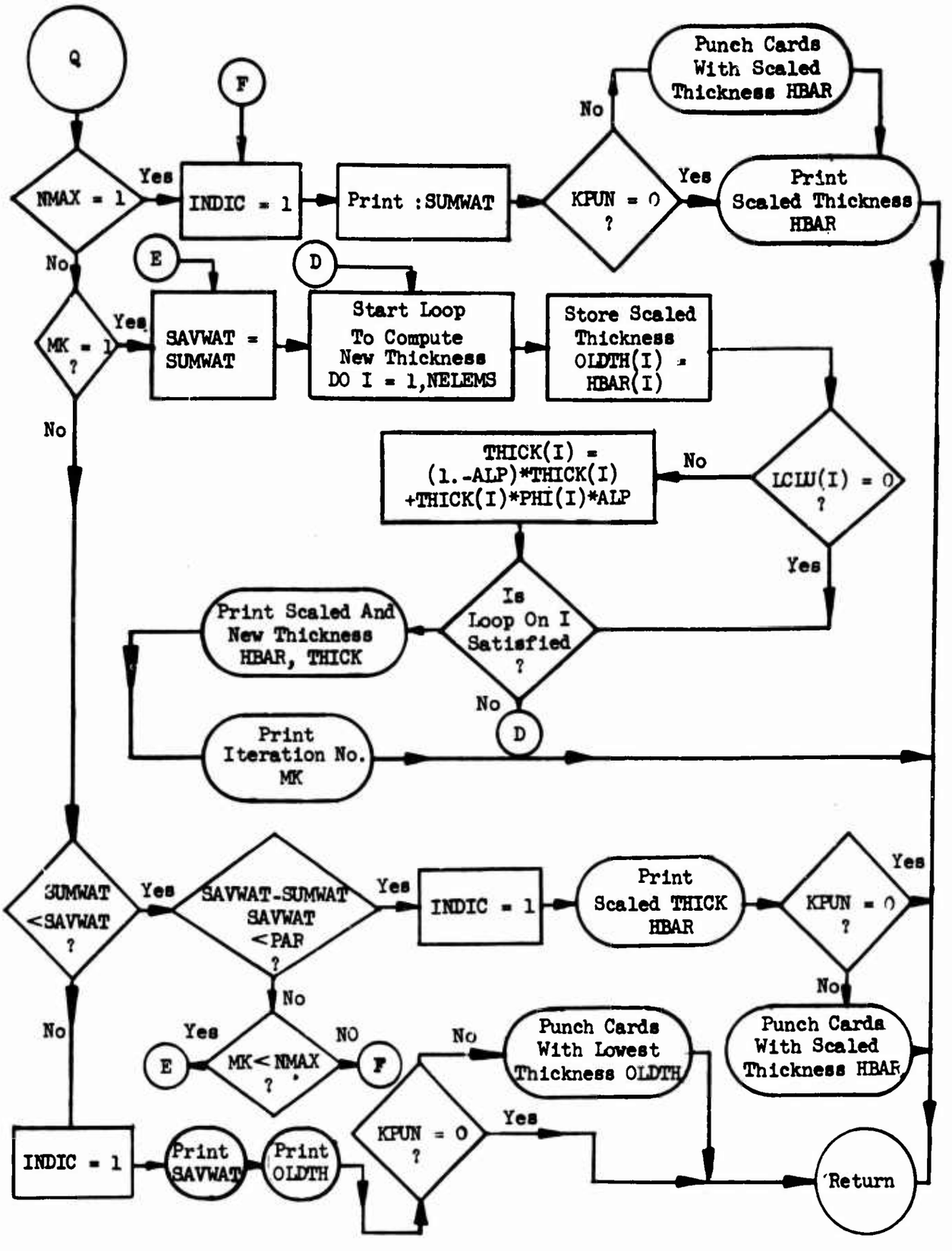
SUBROUTINE SIGMA



SUBROUTINE ØPTMIZ



SUBROUTINE ØPTMIZ (cont.)



Tape Usage

Two scratch tapes are used in this program, ID1 and ID2, with value 1 or 2. One is used to store the element strains calculated for each required time step for the first harmonic. When strains for the next harmonic are computed, the tape is read back, a summation over the harmonics is performed in core, and the results are stored on the second tape. This "flip-flop" procedure continues until the sum of all harmonic strain components for each element, for all specified time steps, are stored on tape. This tape manipulation takes place in subroutine STRESS. The final tape, tape 1 or tape 2 depending on the number of harmonics, is rewound. In MAXST, the strains are read from the tape, fiber stresses are computed from these, and SIGMA is called to find the effective stress for each element, for all time steps.

Array Usage

Certain arrays are of particular significance in this program. Double arrays use their first subscript as the factor determining usage, while the second subscript denotes the dimension for that array. Therefore, the following list will denote a double array by its first subscript in defining its usage.

BIGMK (7, 2610)

- BIGMK (1) - contains mass matrix
- BIGMK (2),..., BIGMK (5) - contains stiffness matrices for up to four harmonics
- BIGMK (6) - work area for integration
- BIGMK (7) - contains damping matrix
  Unless user inputs damping matrix, BIGMK (7) contains zeros.

X(3, 404)

X(1) - contains displacements from integration

X(2) - contains velocities from integration

X(3) - contains accelerations from integration

SIGEFF (100) - holds values of effective stress for each element for any given time step

VALMX (100) - stores maximum value of effective stress for each element for all time steps

AVAL (100, 12) - stores strains associated with the maximum effective stresses for all time steps for all elements (corresponds to values of stress in VALMX). The first subscript is the element number; the second subscript is the dimension holding the twelve strains, ES1, ES2,...,CST2.

Program Counter and Clue Definitions

NG - The number of the current problem. When NG > NGRØUP, program is finished.

MK - The iteration number. Case is complete when it converges or when MK=NMAX.

INDIC - If INDIC = 0, iterations for case continue. If INDIC = 1, case has been completed, program continues with next case, if any.

ID1, ID2 - Tape numbers, 1 or 2, for scratch data sets. For further information see "Tape Usage."

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)	2a. REPORT SECURITY CLASSIFICATION
Air Force Flight Dynamics Laboratory (FBR) Wright Patterson Air Force Base, Ohio 45433	Unclassified
	2b. GROUP

3. REPORT TITLE

AN AUTOMATED PROCEDURE FOR THE OPTIMIZATION OF PRACTICAL AEROSPACE STRUCTURES
VOLUME II PROGRAMMER'S MANUAL

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

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

Walter J. Dwyer
Robert K. Emerton
Patricia L. Sabatelli

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
Feb. 1971	96	None

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
F 33615-69-C-1278	AFFDL-TR-70-118 Volume II
b. PROJECT NO.	
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d.	

10. DISTRIBUTION STATEMENT

Distribution of the document is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
(Volume II)	

13. ABSTRACT

This report documents the computer programs described in Volume I of this report entitled "An Automated Procedure for the Optimization of Practical Aerospace Structures". Both the main structural optimization program and the shell dynamics program are written in Fortran IV language. This manual contains a description of the overlay structure, data set arrangement, and subroutines of both programs. (U)

DD FORM 1 NOV 65 1473

UNCLASSIFIED

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Structural Optimization Finite Element Method Automated Design						