NAVAL SHIP RESEARCH AND DEVELOPMENT CENTER
Bethesda, Maryland 20034

DEFINIT—
A NEW ELEMENT DEFINITION CAPABILITY FOR NASTRAN: USER'S MANUAL

Michael E. Golden
and
Myles M. Hurwitz

COMPUTATION AND MATHEMATICS DEPARTMENT
RESEARCH AND DEVELOPMENT REPORT

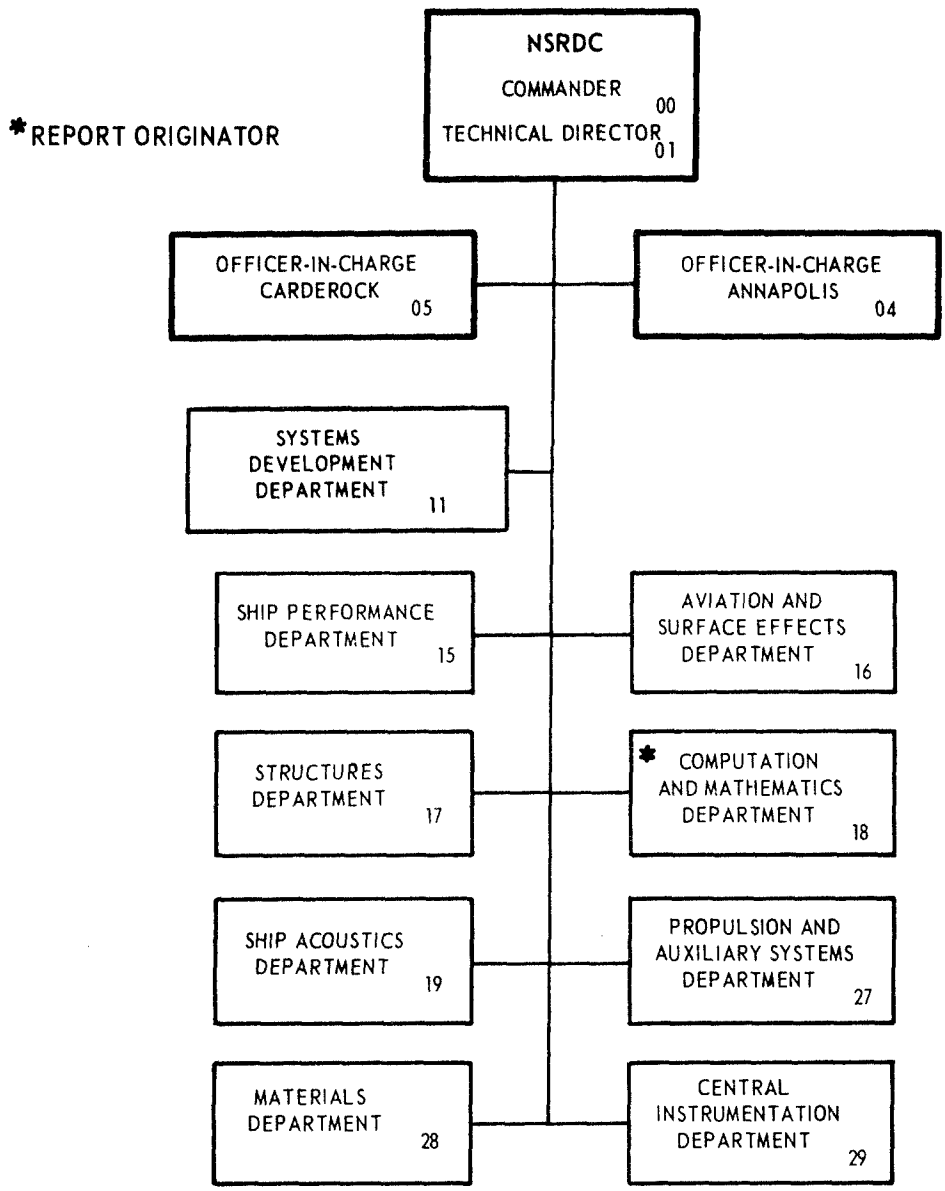
DEC 1973 Report 4250

DEFINIT—A NEW ELEMENT DEFINITION CAPABILITY FOR NASTRAN: USER'S MANUAL

The Naval Ship Research and Development Center is a U. S. Navy center for laboratory effort directed at achieving improved sea and air vehicles. It was formed in March 1967 by merging the David Taylor Model Basin at Carderock, Maryland with the Marine Engineering Laboratory at Annapolis, Maryland.

Naval Ship Research and Development Center
Bethesda, Md. 20034

MAJOR NSRDC ORGANIZATIONAL COMPONENTS



NDW-NSRDC 3960/44 (REV. 8/71)
GPO 917-872

DEPARTMENT OF THE NAVY

NAVAL SHIP RESEARCH AND DEVELOPMENT CENTER

Bethesda, Maryland 20034

DEFINIT—
A NEW ELEMENT DEFINITION CAPABILITY FOR NASTRAN: USER'S MANUAL

Michael E. Golden

and

Myles M. Hurwitz

DEC 1973 REPORT 4250

TABLE OF CONTENTS

TABLE OF CONTENTS—Continued

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

To relieve the user of NASTRAN—the National Aeronautics and Space Administration's general purpose, finite element, structural analysis computer program—from the necessity of becoming involved with internal aspects of NASTRAN when he adds a new element, a new element definition capability has been developed. This capability takes the form of a preprocessor which will generate, according to user specifications, the FORTRAN routines and tables required by NASTRAN for a new element.

This manual contains details and instructions on the use of the preprocessor, and provides numerous examples.

ADMINISTRATIVE INFORMATION

The new element definition capability for NASTRAN was developed under the In-House Independent Research Program, Task Area ZR1040201 and informally reported in a Computation and Mathematics Department technical note. The work of preparing this report was carried out under Work Unit 1-1844-009.

I. INTRODUCTION

NEED FOR THE PREPROCESSOR

NASTRAN—the National Aeronautics and Space Administration's general purpose, finite element, structural analysis computer program[1,2,3]—is widely used in the structural engineering field for a variety of calculations including those for static analysis, piece-wise linear analysis for nonlinear material properties, natural frequencies, transient analysis, and frequency and random response. The dynamic calculations may be carried out either directly or by the modal method.

The system design of NASTRAN is excellent. Occasionally, however, a user may wish to include some structural elements of his own in the program. This he may do if he will first update the appropriate NASTRAN tables, write the additional subroutines which will be needed for the extra elements, and incorporate these updates and additions into NASTRAN. Since such procedures take time, and require a great deal of familiarity with various internal aspects of NASTRAN—the variables, tables, routines, structure, restrictions, and even the philosophy of the program itself—a new element definition capability has been developed to perform a major portion of the above tasks. This new element definition capability, which is described within the pages of this report, is very helpful to the user in defining the elements he wishes to add to NASTRAN.

---

[1] "The NASTRAN Theoretical Manual, (Level 15.0)" Edited by R.H. MacNeal, NASA SP-221(01) (Dec 1972).

[2] "The NASTRAN User's Manual (Level 15.0)" Edited by C.W. McCormick, NASA SP-222(01) (May 1973).

[3] "The NASTRAN Programmer's Manual (Level 15.0)" NASA SP-223(01) (Sep 1973).

This element definition capability takes the form of a preprocessor. The user supplies it with symbolic instructions submitted as "packets" of information, and the preprocessor then generates all the FORTRAN language tables and routines required for the element to be added. The functions performed by the preprocessor and the packets of information it uses are explained in detail later in the report.

The preprocessor was planned for use with the Level 12.0 version of NASTRAN, and so is subject to all the restrictions imposed by that version. Two changes were made to Version 12.0 to adapt it for use with the preprocessor:

(1) Twelve dummy elements were set up. For each of the twelve, table entries were initialized, skeleton subroutines were set up (to be eventually replaced with the FORTRAN subroutines generated for the new element), and appropriate linkage for the subroutines was established;

(2) Additional coding was incorporated within the NASTRAN routines as appropriate to enable NASTRAN to retrieve the user-defined table data from the preprocessor-generated BLOCK DATA subprograms.

After the preprocessor had been written, based on the Level 12.0 version of NASTRAN, another version of NASTRAN was developed—NASTRAN Version 15.1. This version 15.1, which has dummy elements already built in, may be used instead of Version 12.0; in fact, in many cases its use may be preferable, since it provides a considerable savings in computer time over that used by Version 12.0

The separate steps required to insert the routines generated by this element definition capability are outlined in a later section of this report.

The preprocessor and the updated Version 12.0 NASTRAN executable file required for use with the preprocessor are available for CDC 6000-series computers. The preprocessor is written in the SNOBOL computer language, Version 3.

NASTRAN MODULES AFFECTED BY ADDED ELEMENTS

A little background on the NASTRAN functional modules affected by the addition of a new structural element may be helpful to the reader. The NASTRAN functional modules affected by the addition of a new module are:

- Input File Processor (IFP)
- Executive Control Section Analysis (XCSA)
- Geometry Processor (GP1, GP2, GP3)
- Table Assembler (TA1)
- Plot Set Definition Processor (PLTSET)
- Structural Matrix Assembler - Phase 1 (SMA1)
- Structural Matrix Assembler - Phase 2 (SMA2)
- Static Solution Generator - Phase 1 (SSG1)
- Stress Data Recovery - Phase 2 (SDR2)
- Output File Processor (OFP)
- Differential Stiffness Matrix Generator - Phase 1 (DSMG1)

The following paragraphs describe the relationships between the new structural elements and the various modules:

Input File Processor (IFP)

The NASTRAN Input File Processor determines whether the form of each input bulk data card is correct. It checks the name of the bulk data card and each data field for appropriate type, i.e., integer, real, and so on. The IFP module contains seven tables from which information is obtained. Most of this information has already been supplied for the dummy elements available. The remaining information must be supplied to the tables by the preprocessor.

The user provides this table information to the preprocessor. For each new element he must design a connection card, and perhaps a property card.

Executive Control Section Analysis (XCSA)

One of the functions of the XCSA module is to transmit to the remainder of the program the restart tables associated with each rigid format. One table and one subroutine must be updated for each rigid format. However, this information has already been supplied for the dummy elements which have been made available. Therefore, no further updates to XCSA by the preprocessor user are necessary.

Geometry Processor (GP1, GP2, GP3) and Table Assembler (TA1)

The Geometry Processor modules produce tables which contain information concerning grid points, element connection, loads, and coordinate system transformations. The Table Assembler module combines much of the information produced by the Geometry Processor into other tables for use later in the program.

In order to produce these tables, modules GP1, GP2, GP3 and TA1 depend, in part, on another table. Most of the information in this table has already been supplied for the dummy elements. The remainder of the information is obtained from the preprocessor which, in turn, finds the data in the Preliminary Data Packet.

Plot Set Definition Processor (PLTSET)

The PLTSET module prepares information so that structural plots may be made. Again, most of the information has been given for the dummy elements, and the remainder is obtained through the preprocessor via the Preliminary Data Packet. (This means that the new element may be plotted.)

Structural Matrix Assembler - Phase 1 (SMA1)

The SMA1 module calculates the stiffness matrix for each element in the problem and inserts it into the structure stiffness matrix. It is in this module that the stiffness matrix routine for the new element must be placed. The linkage between the new routine and the remainder of the program has already been established. The preprocessor will generate the new routine from the information provided in the Stiffness Matrix Packet. Later, we will describe how this or any new routine may be inserted into NASTRAN.

Structural Matrix Assembler - Phase 2 (SMA2)

The SMA2 module does for the mass matrix and viscous damping matrix what SMA1 does for the stiffness matrix. The preprocessor generates the new mass matrix according to the Mass Matrix

Packet. The viscous damping matrix routine is generated according to the Viscous Damping Matrix Packet.

Static Solution Generator - Phase 1 (SSG1)

The SSG1 module calculates the load vector for a status problem. If the new element has applied thermal loads due to temperatures at the grid points, the method for calculating the thermal loads for the element must be given in the Thermal Loading Vector Packet from which the preprocessor will generate the appropriate routine. Again, the linkage for the new routine has already been set up. Only the new routine and its insertion into NASTRAN are required.

Stress Data Recovery - Phase 2 (SDR2)

For all elements the SDR2 module calculates final stresses and forces. These calculations are performed in two stages. The first stage consists of forming the element stress matrix and passing it, along with other element properties, to the second stage. The second stage computes the final stresses and forces. Therefore, two new routines are required for each new element, one for each stage. The preprocessor will generate these two routines from the information given in the Stress Matrix and Stress and Force Calculation Packets. Once again, the linkage for the new routines has been established.

Output File Processor (OFP)

The OFP module prints element stresses, forces, and displacements. This module is almost totally table-dependent, and if stresses and/or forces are to be output for a new element, the correct tables must be updated. The preprocessor will update the necessary tables with information supplied by the Output Packet.

Differential Stiffness Matrix Generator - Phase 1 (DSMG1)

The DSMG1 module does for the differential stiffness matrix what the SMA1 module does for the elastic stiffness matrix. The preprocessor generates the new differential stiffness matrix according to the Differential Stiffness Matrix Packet.

MATRIX-PARTITION PHILOSOPHY

Since matrix partitions are basic to the matrix calculations made by NASTRAN and therefore the preprocessor, the reader will need to understand the NASTRAN philosophy which underlies the preprocessor design. Note the distinction between the term "word" and the term "variable" as used in the following discussion: a variable is considered to be an array of words, each word being one element of the array. An array may be either a matrix, a vector, or a scalar.

In general, a grid point in NASTRAN is considered to have six degrees of freedom. Therefore, if an element has N grid points, the element stiffness matrix is of the order 6NX6N. In most programs, this 6NX6N matrix will be computed for an element and then inserted into the structure stiffness matrix. However, this procedure is not used in NASTRAN. Instead, during any one entry into the stiffness matrix routine for an element, each row of 6X6 partitions, is calculated, one partition at a time, and inserted into the structure stiffness matrix. For example, consider the 6NX6N matrix to be composed of N^2 6X6 matrix partitions. Each partition KIJ has associated with it two grid points I and J (I and J will be identical on the diagonal). If an element has three grid points, its stiffness matrix will be partitioned as follows:

	1	2	3
1	K11	K12	K13
2	K21	K22	K23
3	K31	K32	K33

Therefore, in our example, the element subroutine will be entered three times, once for each row of partitions. This method of insertion is used for stiffness, mass, differential stiffness, and viscous damping matrices.

During one entry into the thermal load vector routine for an element, the load corresponding to each grid point is calculated and inserted individually into the overall load vector. Therefore, only one entry into the thermal load vector routine, for a particular element, is necessary.

This method of matrix and vector insertions made it desirable to design the preprocessor to accept input which describes both matrix and vector partitions, although a full 6NX6N matrix may be specified by the user. Since we are attempting to make the preprocessor as general as possible within the limitations of NASTRAN, i.e., to allow a wide variety of elements to be specified, several combinations of options have been provided for specifying matrices or matrix partitions. The options may be divided into two groups.

The first group of options concerns the dimensioning of the matrix partitions to be specified. The choices are 1X1, 3X3, 6X6, 6NX6N. Although the usual matrix partition size is 6X6, a 3X3 matrix partition will suffice if the degrees-of-freedom of the element are 1, 2, and/or 3 (i.e., only translations); or 4, 5, and/or 6 (i.e., only rotations). In such cases, the generated routine will insert the 3X3 partition into the particular 6X6 partition required by NASTRAN for insertion into the structure stiffness matrix. An example is the triangular membrane element now in NASTRAN. If an element is so structured that an individual matrix partition cannot be described, then the full 6NX6N matrix may be specified instead.

In this case, the full 6NX6N matrix is calculated, after which the appropriate 6X6 partitions are extracted as needed. However, this method is more costly than the one describing each partition, since the full matrix must be calculated N times, once for each entry into the element routine. The 1X1 partition allows the user to incorporate scalar elements with NASTRAN.

The second group of options concerns the way in which partitions are defined. There are three possibilities. One describes each of the N^2 partitions separately. Since the full 6NX6N matrix is symmetric, only $N(N+1)/2$ partitions need be specified, i.e., the symmetric and diagonal partitions; the preprocessor will generate the remaining partitions for the user. A second way describes all N^2 partitions with one equation in which variables that end with the letters I or J take on various values. For example, in this second method, the variable CI might represent the variables C1, C2, C3,..., CN, where N is the number of grid points. Thus, all the partitions could be described with just one equation. A third way to define the partitions specifies an equation which represents the full 6NX6N matrix and enables partitions to be extracted as described previously.

DUMMY ELEMENTS

To facilitate the insertion of new elements into NASTRAN by means of the new element definition capability, NASTRAN (Version 12.0) has been modified to include twelve dummy elements. The general release Version 12.0 of NASTRAN contains 38 elements, with a reasonable limit on the total number of elements (a limit partially imposed by NASTRAN) of 50. The new elements are therefore named ELEM39, ELEM40,..., ELEM50. These elements need not be used in any particular order.

NASTRAN was further modified by inserting within the NASTRAN tables the names of the dummy elements, some internal codes, and some default values, most of which may be overridden by the user. The new element subroutines are already named so that they may be linked with the remainder of NASTRAN by supplying the appropriate FORTRAN CALL statements. For ELEM39, the corresponding subroutine names (which follow present NASTRAN naming conventions) are as follows:

Stiffness Matrix	KLEM39
Mass Matrix	MLEM39
Viscous Damping Matrix	VLEM39
Thermal Loading Matrix	M39
Stress Matrix	SEL391
Stress and Force Calculation	SEL392
Differential Stiffness Matrix	DLEM39

The names of the routines for the other dummy elements follow a similar pattern.

II. INPUT TO THE PREPROCESSOR

USE OF DATA PACKETS

The input to the preprocessor describes methods to NASTRAN for performing calculations (a stiffness matrix, a mass matrix, and so on), so that the analyst, using his element, may take advantage of the existing NASTRAN analyses and system design. Although the preprocessor relieves the analyst of the chore of learning NASTRAN's requirements and idiosyncrasies related to writing element subroutines and updating tables, the analyst retains responsibility for specifying his requirements to the preprocessor. To simplify this task, which itself could be an arduous one, depending upon the complexity of the element involved, the input has been categorized into different "packets" of information, all but two made up of subpackets. There are ten packets at the present time:

- Preliminary Data Packet
- Global Variable Packet
- Stiffness Matrix Packet
- Mass Matrix Packet
- Viscous Damping Matrix Packet
- Thermal Loading Vector Packet
- Stress Matrix Packet
- Stress and Force Calculation Packet
- Output Packet
- Differential Stiffness Matrix Packet

One of these—the Preliminary Data Packet—must be submitted each time a run is to be made. This packet contains such information as the element name (chosen from a list of dummy names provided), the number of grid points, the degrees of freedom that the element is allowed, etc. All other packets are optional and may be supplied at the user's discretion. If the optional Output Packet is supplied, tables related to the NASTRAN output format of user stress and force calculations will be updated. Each of the others, with the exception of the Global Variable Packet, supplies information in symbolic form

that is necessary for the generation of a particular FORTRAN subroutine. Thus, for example, the Stiffness Matrix Packet would supply symbolic definitions of variables which the preprocessor would need to use in interpreting the information and generating a FORTRAN subroutine which would be used later in calculating element stiffness matrices.

The Global Variable Packet is provided for the user in keeping with NASTRAN's policy of recalculating rather than saving and retrieving information that is needed in more than one situation. The Global Variable Packet makes it possible for the user to specify his multipacket variables the one time instead of each time they are used.

Detailed descriptions of the ten packets are provided in Section III. The subpackets which compose the packets are described in another segment of this section.

CARD FORMAT

All input to the preprocessor is in free-field format. Data fields within a data card are separated by commas, with two successive commas indicating that a default value for the field is to be taken. A dollar sign punched in any column of the card indicates that a continuation data card follows. If there is no $ punch on the card, it is either the last card or the only card in the sequence.

VARIABLE NAMES

The variable names that may be used in the preprocessor expressions are not completely arbitrary, although very nearly so. There are two categories of so-called "illegal" variable names: Those that may never be used (Table 2), and those that may be used under certain limitations (Tables 1, 3, 4). In the first category are the names already being used as SNOBOL variables in the preprocessor. If the analyst were to use these names, the preprocessor could become confused and the

results would be unpredictable. In the second category are those reserved names which have already been defined by NASTRAN COMMON statements or by the preprocessor and may be used only according to these preassigned definitions. Table 3 lists the COMMON names, the variables in each COMMON, and the preprocessor definition of each variable. Table 4 lists the variables made available to the user by the preprocessor. Individual packet descriptions specify which COMMON names are used in a particular subroutine. Tables 5 and 6 provide a more detailed description of portions of Tables 3 and 4 respectively, while Table 7 combines the Tables 1 through 6 into a comprehensive alphabetical listing for the user's convenience. These tables follow:

TABLE 1 - FORTRAN LIBRARY FUNCTIONS
RECOGNIZED BY THE PREPROCESSOR

Function Name	Function Name
DSIN	SIN
DCOS	COS
DSQRT	SQRT
DEXP	EXP
DLOG	ALOG
DATN	ATAN
	TAN
DABS	ABS

TABLE 2 - SNOBOL VARIABLE NAMES DENIED THE USER

Note: These variable names may neither be referenced nor redefined

A	CKFLAG	DTEST
ABORT	COL	DUMDAT
ALPH	COMMA	DUMVAR
ANGLE	COMMO	D11
ARB	COMM1	D12
ATEST	COMM2	D2
A1	COMM3	D21
A3	COMM4	D22
A4	COMMON	D3
B	COMPLEX	D4
BAL	CONTINUE	ELMDEF
BDATA	CPVAR	EL1
BDO	DATA	EL2
BEG	DEFEND	EL3
BLANK	DEFER	EN
BLANK15	DEFFLAG	ENDG
BLANK5	DELFLAG	EQFLAG
BLANK20	DELIMITER1	EQI
BLANK6	DELIMITER2	EQJ
BLANK8	DIM	EQK
BLANK9	DIMSAVD	EQT
BLANK72	DISFLAG	EQUA
B1	DISFLAG1	ERMS
CARCTRL	DISFLAG2	ERRCNT
CARD	DISPAT	F
CARDPRINT	DMP	FAIL
CC	DO	FENCE
CHARACTERS	DOLLAR	FILE
CHECKER	DS	FINFLAG

TABLE 2 - SNOBOL VARIABLE NAMES DENIED THE USER—Continued

FORCE	GMV3	IS1
FORMAT	GMV4	IS2
FORT	GMV5	I1
FORTFLAG	GNCHECK	I2
FORTFUNCT	GNCHECK1	I3
FORSAVE	GOTO	I4
FPASS	GVGLAG	I5
FUNCTIONS	GV1	J
FUNCT11	GV2	JE
FUNCT12	HA	JPASS
FUNCT21	HB	JPOINTER
FUNCT22	HIER	JPROGFLAG
FUNCT31	HOWDEF	KFLAG
FUNCT32	I	KI
FXV1	ID	LENG
FXV2	IDATA	LENGER
FXV3	IDD	LENGER1
FXV4	IDSAV	LIMIT
FXV5	IE	LINER
FXV6	IF2	LINE1
FXV10	IND	LINE21
F1	INDC	LINE22
F2	INDSAV	LINE3
F3	INFLAG	LINE4
F4	INPUT	LIST
GENCHECK	INSCHK	LIST1
GENSAVE	INST	LIST2
GETCARD	INTEGER	LL
GMPA	INTVAR	LOC
GMV1	IOFP1	LOC1
GMV2	IPARAM	LP

TABLE 2 - SNOBOL VARIABLE NAMES DENIED THE USER—Continued

LR	NUMBER	PAT
MAGPH	NUMBERS	PATOU1
MAINVAR	NUMBER5	⋮
MATCH	NUMBER05	PATOU11
MATCH1	NV	PATA
MATCH2	N1	⋮
MATCH3	N2	PATZ
MATEQ	N3	PATF1
MAX1	N4	⋮
MAX2	N5	PATF6
MAX3	N6	PAT1
MAX4	N7	PAT2
MAX5	N8	⋮
MEQP	OPER	PAT100
MESS	OPTION	PCC
N	OPTION1	PERIOD
NC	OPT1	PHCOM
NEWFUN	OPT2	PHCOM1
NG	OUTFLG	PHSTR
NMN	OUTFLG8	PIECE
NMS	OUTFLG9	PIECES
NMU	⋮	POINTER
NN	OUTFL24	POLSAV
NOCARDS	OUTPUT	PRE
NOGONE	PACKED	PREI
NOPIECES	PACKET	PRFL
NOREAD	PACKID	PROGFLAG
NORES	PACKNO	PUNCH
NOVAR	PACKPAT	P1
NULL	PARTITION	P2
NUMB	PASS	P3

TABLE 2 - SNOBOL VARIABLE NAMES DENIED THE USER—Concluded

QCOM	S	TRB
QD	SAVE	TRB1
QD1	SAVED	TRB2
QDIM	SAVTAB	TRPAT
QELNAMQ	SAVVAR	TR1
QEQ	SLASH	TR2
QEQU	SOR1	UL
QOP	SOR2	UNARY
QQEQ	STARS	UR
QTIME	STNO	V
QTIM1	STR	VAR
QTIM2	STRESS	VAREQ
QUE	STRING	VARFLAG
QX	SUCCEED	VARGENR
QXYZ	SV1	VARN
Q1	SV2	VARNAM
Q2	S1	VARTYP
Q3	S2	V2
⋮	⋮	V4
REAL	S7	V5
REM	TA	V6
REPL	TABLE	WORDS
RESDEF	TDIM1	WRITFLAG
RESDIM	TDIM2	
RESEQ	TEMPOR	
RESVAR	TESTPAT	
RIMAG	THRU	
ROW	THRUU	
RSV1	TIMER	
RSV2	TRANFLAG1	
RUNNO	TRANFLAG2	

TABLE 3 - NASTRAN-DEFINED VARIABLE NAMES

Note 1: These variables are available to the user as already defined by NASTRAN, but may not be redefined by the user.
Note 2: The asterisk indicates the variable names most commonly used.

Common Name	Variable	Definition
MATIN	MATID	Material ID
	INFLAG	Input variable as described on Page 66.
	ELTEMP	Element temperature.
	STRESS	Used in piecewise linear analysis.
	SINTH	SIN of the anisotropic material angle.
	COSTH	COS of the anisotropic material angle.
MATOUT	Depends on INFLAG value	See Table 5
SMA1ET	ECPT	Reference variable for real numbers.
	NECPT	Reference variable for integer numbers.
	NGRID	A dimensional variable such that NGRID(I) is the Ith grid point of the element.
	MATID1	Material ID
	ID1,ID2,...,IDN	Coordinate system number for the 1st, 2nd,...,Nth grid points of the element.
	*X1,X2,...,XN	X-coordinates of the 1st, 2nd,...,Nth grid points of the element in basic coordinates.
	*Y1,Y2,...,YN	Y-coordinates of the 1st, 2nd,...,Nth grid points of the element in basic coordinates.

TABLE 3 - NASTRAN-DEFINED VARIABLE NAMES—Continued

Common Name	Variable	Definition
	*Z1,Z2,...,ZN	Z-coordinates of the 1st, 2nd,...,Nth grid points of the element in basic coordinates.
	DUMV	Internal dummy variables.
	All variable names given on the Preliminary-Data Packet connection and property cards.	All user-defined connection card and property variables as given in the Preliminary Data Packet.
SMA1IO	DUM1	Internal dummy variable.
	DUM2	Internal dummy variable.
	DUM3	Internal dummy variable.
	IFKGG	Internal file number.
	IF4GG	Internal file number
SMA1CL	IOPT4	Internal variable.
	K4GGSW	Internal variable.
	NPVT	Internal variable.
SMA1DP	I	Dummy variable.
	J	Dummy variable.
	IS	Dummy variable.
	IP	Dummy variable.
	I1,I2	Dummy variable.
	QI, I=1,...,9	Dummy variables.
	*PI	π to 9 significant digits
SMA2ET	Same as SMA1ET	
SMA2IO	IFMGG	Internal file number.
	IGMGG	Internal variable.
	IFBGG	Internal file number.
SMA2CL	BGGIND	Internal variable.
SMA2DP	Same as SMA1DP	

TABLE 3 - NASTRAN-DEFINED VARIABLE NAMES—Concluded

Common Name	Variable	Definition
EDTSP	Same as SMA1ET	
TRIMEX	Same as SMA1ET	
SDR2X5	Same as SMA1ET	
	PH1OUT	Internal variable.
	FORVEC	Internal variable.
SDR2X6	Same as SMA1DP	
SDR2XX	Z	Internal variable
SDR2X4	DUM	Dummy variable.
SDR2X4	IVEC	Internal variable.
	IVECN	Internal variable.
	*TEMP	Element temperature (average of the grid point temperatures for the element).
	DEFORM	Element deformation (not used with preprocessor).
SDR2X7	PH1OUT	Internal variable.
	FORVEC	Internal variable.
SDR2X8	Same as SMA1DP	
DS1ADP	Same as SMA1DP	
DS1AAA	NPVT	Internal variable.
DS1ET	Same as SMA1ET	

TABLE 4 - PREPROCESSOR-DEFINED VARIABLE NAMES

Note: These variables are available to the user as already defined by the Preprocessor; they may not be redefined.

Variable Name	Definition	Packets in which Available
PI	3. 14159265	All
T1 T2 : TN	Transformation matrices for the first, second,...,Nth grid point for an element from the basic coordinate system to the global coordinate system ID1,ID2,...IDN	All
G	Material properties matrix (Tables 5 and 6)	All
TTI(J)	Temperature at the Jth grid point for an element	Thermal Loading
DISP1 DISP2 : DISPN	Displacement vector for the first, second,...,Nth grid point for an element. The vector is 3X1 if the degrees of greedom are 1, 2, and/or 3; or 4, 5, and/or 6. The vector is otherwise 6X1.	Stress and Force Calculation
XYZ1 XYZ2 : XYZN	Alternate method of referencing X1, Y1, Z1; X2, Y2, X2; ...XN, YN, ZN defined in COMMON block SMA1ET. They are 3X1 vectors predefined by the preprocessor.	All

TABLE 5 - EXPANDED DEFINITION OF COMMON MATOUT (TABLE 3)

Note: For INFLAG values 1 and 4, the Material ID must be supplied on a MAT1 card; for INFLAG values 2 and 3, the material ID may be provided on either a MAT1 or MAT2 card.

Form of MATOUT	Definition
INFLAG=1	
E	Young's modulus.
GG	Shear Modulus.
XNU	Poisson's ratio.
RHO	Density.
ALPHA	Thermal expansion coefficient.
TSUBO	Thermal expansion reference temperature.
GSUBE	Structural element damping coefficient.
SIGTEN	Stress limit for tension.
SIGCOM	Stress limit for compression.
SIGSHE	Stress limit for shear.
INFLAG=2	
G11	The 3X3 symmetric material property matrix.
G12	
G13	
G22	
G23	
G33	
RHO	Density
ALPHA1	Thermal expansion coefficient vector.
ALPHA2	
ALP12	
TSUBO	Thermal expansion reference temperature.
GSUBE	Structural element damping coefficient.
SIGTEN	Stress limit for tension.

TABLE 5 - EXPANDED DEFINITION OF COMMON MATOUT (TABLE 3)—Concluded

Form of MATOUT	Definition
SIGCOM	Stress limit for compression.
SIGSHE	Stress limit for shear.
INFLAG=3	
G11	The 3X3 symmetric material property matrix.
G12	
G13	
G22	
G23	
G33	
RHO	Density.
ALPHA1	Thermal expansion coefficient vector.
ALPHA2	
ALP12	
TSUBO	Thermal expansion reference temperature.
GSUBE	Structural element damping coefficient.
SIGTEN	Stress limit for tension.
SIGCOM	Stress limit for compression.
SIGSHE	Stress limit for shear.
XJ11	The 2X2 transverse shear inverse matrix.
XJ12	
XJ22	
INFLAG=4	
RHO	Density.

TABLE 6 - EXPANDED DEFINITION OF RESERVED VARIABLE G

Note: These variables are available to the user as already defined by NASTRAN. They may not be redefined.

	Definition of G
INFLAG=1	$\frac{E}{1 - (XNU)^2} \begin{bmatrix} 1 & XNU & 0 \\ XNU & 1 & 0 \\ 0 & 0 & GG \end{bmatrix}$
INFLAG=2	$\begin{bmatrix} G11 & G12 & G13 \\ G12 & G22 & G23 \\ G13 & G23 & G33 \end{bmatrix}$
INFLAG=3	Same as for INFLAG=2.
INFLAG=4	G is undefined.

TABLE 7 - COMPREHENSIVE ALPHABETICAL LIST OF VARIABLE NAMES EITHER RESTRICTED OR FORBIDDEN THE USER

Note: The symbol † denotes those variable names which may be referenced (but not redefined) by the user; all others are forbidden the user.

A	B1	DELIMITER1	†DUM3
ABORT	CARCTRL	DELIMITER2	D11
†ABS	CARD	†DER	D12
†ALOG	CARDPRINT	†DEXP	D2
ALPH	CC	DIM	D21
†ALPHA	CHARACTERS	DIMSAVD	D22
†ALPHA1	CHECKER	DISFLAG	D3
†ALPHA2	CKFLAG	DISFLAG1	D4
†ALP12	COL	DISFLAG2	†E
ANGLE	COMMA	DISPAT	†ECPT
ARB	COMMO	†DISP1	ELMDEF
†ATAN	COMM1	†DISP2	†ELTEMP
ATEST	COMM2	⋮	EL1
A1	COMM3	†DLOG	EL2
A3	COMM4	DMP	EL3
A4	COMMON	DO	EN
B	COMPLEX	DOLLAR	ENDG
BAL	CONTINUE	DS	EQFLAG
BDATA	†COS	†DSIN	EQI
BDO	†COSTH	†DSQRT	EQJ
BEG	CPVAR	DS1AAA	EQK
†BGGIND	DATA	DS1ADP	EQT
BLANK	†DABS	DS1ET	EQUA
BLANK15	†DATN	DTEST	ERMS
BLANK20	†DCOS	†DUM	ERRCNT
BLANK5	DEFEND	DUMDAT	†EXP
BLANK6	DEFER	†DUMV	F
BLANK8	DEFFLAG	DUMVAR	FAIL
BLANK9	†DEFORM	†DUM1	FENCE
BLANK72	DELFLAG	†DUM2	FILE

TABLE 7 - COMPREHENSIVE ALPHABETICAL LISTING OF VARIABLE NAMES EITHER RESTRICTED OR FORBIDDEN THE USER—Continued

FINFLAG	GNCHECK	IND	†K4GGSW
FORCE	GNCHECK1	INDC	LENG
FORMAT	GOTO	INDSAV	LENGER
FORT	†GSUBE	†INFLAG	LENGER1
FORTSAVE	GVFLAG	INPUT	LIMIT
†FORVEC	GV1	INSCHK	LINER
FPASS	GV2	INST	LINE1
FUNCTIONS	†G11	†INT	LINE21
FUNCT11	†G12	INTEGER	LINE22
FUNCT12	†G13	INTVAR	LINE3
FUNCT21	†G22	†INV	LINE4
FUNCT22	†G23	IOFL1	LIST
FUNCT31	†G33	†IOPT4	LIST1
FUNCT32	HA	†IP	LIST2
FXV1	HB	IPARM	LL
FXV2	HIER	†IS	LOC
⋮	HOWDEF	IS1	LOC1
FXV6	†I	IS2	LP
FXV10	ID	†IVEC	LR
F1	IDATA	†IVECN	MAGPH
F2	IDD	I1	MAINVAR
F3	IDSAV	I2	MATCH
F4	†ID1	I3	MATCH1
†G	†ID2	I4	MATCH2
GENCHECK	⋮	I5	MATCH3
GENSAVE	IE	†J	MATEQ
GETCARD	†IFBGG	JE	MAX1
†GG	†IFKGG	JPASS	⋮
GMPA	†IFMGG	JPOINTER	MAX5
GMV1	IF2	JPROGFLAG	†MADID
⋮	†IF4GG	KFLAG	†MATID1
GMV5	†IGMGG	K1	MATIN

TABLE 7 - COMPREHENSIVE ALPHABETICAL LISTING OF VARIABLE NAMES EITHER RESTRICTED OR FORBIDDEN THE USER—Continued

MATOUT	OPER	PAT2	QTIM2
MEQP	OPTION	⋮	QUE
MESS	OPTION1	PAT100	QX
N	OPT1	PCC	QXYZ
NC	OPT2	PERIOD	Q1
†NECPT	OUTFLAG	PHCOM	Q2
NEWFUN	OUTFL8	PHCOM1	⋮
NG	OUTFL9	PHSTR	REAL
†NGRID	⋮	†PH1OUT	REM
NMN	OUTFL24	†PI	REPL
NMS	OUTPUT	PIECE	RESDEF
NMU	PACKED	PIECES	RESDIM
NN	PACKET	POINTER	RESEQ
NOCARDS	PACKID	POLSAV	RESVAR
NOGONE	PACKNO	PRE	†RHO
NOPIECES	PACKPAT	PREI	RIMAG
NOREAD	PARTITION	PRFL	ROW
NORES	PASS	PROGFLAG	RSV1
NOVAR	PAT	PUNCH	RSV2
†NPVT	PATOU1	P1	RUNNO
NULL	⋮	P2	S
NUMB	PATOU11	P3	SAVE
NUMBER	PATA	QCOM	SAVED
NUMBERS	PATB	QD	SAVTAB
NUMBER05	⋮	QDIM	SAVVAR
NUMBER5	PATZ	QD1	SDR2XX
NV	PATF1	QELNAMQ	SDR2X4
NV1	⋮	QEQU	SDR2X5
NV2	PATF6	QOP	SDR2X6
N1	PAT1	QQEQ	SDR2X7
⋮		QTIME	SDR2X8
N8		QTIM1	†SIGCOM

TABLE 7 - COMPREHENSIVE ALPHABETICAL LISTING OF VARIABLE NAMES EITHER RESTRICTED OR FORBIDDEN THE USER—Concluded

†SIGSHE	TEMPOR	VARTYP
†SIGTEN	TESTDAT	V2
†SIN	THRU	V4
†SINTH	THRUU	V5
SLASH	TIMER	V6
SM1CL	TRANFLAG1	WORDS
SMA1ET	TRANFLAG2	WRITFLAG
SM1IO	TR	†XJ11
SMA2ET	TRB	†XJ12
SMA2IO	TRB1	†XJ22
SOR1	TRB2	†XNU
SOR2	TRIMEX	†X1
†SQRT	TRPAT	†X2
STNO	TR1	⋮
STR	TR2	†Y1
†STRESS	†TSUBO	†Y2
STRING	†TTI	⋮
SUCCEED	†T1	†Z
SV1	†T2	†Z1
SV2	⋮	†Z2
S1	UL	⋮
S2	UNARY	
⋮	UR	
S7	V	
TA	VAR	
TABLE	VAREQ	
†TAN	VARGENR	
TDIM1	VARFLAG	
TDIM2	VARN	
†TEMP	VARNAM	

SYMBOLIC EQUATION EXPRESSION

Much of the input will consist of symbolic expressions. These expressions must be written in FORTRAN and must adhere to all FORTRAN rules, including those for the naming of real and integer variables, balanced parentheses, hierarchy of arithmetic operations, etc. There is one very important relaxation of the FORTRAN rules. Input to the preprocessor is matrix-oriented, that is, all variables are considered to be matrices. Scalars are 1X1 matrices. Therefore, if A and B are matrices, the preprocessor will (1) recognize the expression A*B as a matrix multiplication, (2) make sure that it is a well-defined matrix multiplication (although a 1X1 matrix may multiply a matrix of any order), and (3) generate the code necessary to perform the multiplication. This preprocessor-defined symbolic definition of matrix operations is an important extension to the available FORTRAN capability.

The functions listed in Table 1 and the four new functions TR, INV, DER, and INT may never be used as variable names. Of these four new symbolic functions, only the following two are operational at the present time:

TR	matrix transpose function
INV	matrix inversion function

The TR function may be used in the preprocessor input to indicate the transpose of a matrix or a matrix expression, but should not be used in such a way that the transpose of an entire right-hand side of an expression is taken, since this would prove extremely wasteful. The following three examples will indicate some uses of the transpose function. Each example should be considered as the right-hand side of an equation. All the variables refer to matrices.

Example 1: TR(A)*B

Example 2: TR(A)

Example 3: TR(C1*TR(E1)*T1)*G(C1*TR(E1)*T1)

All three examples illustrate acceptable uses of the transpose function, although the second example will generate unnecessary coding. The preprocessor will take an expression such as that in the first example and generate a call to a NASTRAN matrix multiply routine, and will at the same time set a flag which will indicate that B is to be multiplied by the transpose of A. However, the actual transpose of A will never be computed; the transpose operation will be performed by referencing the elements of A in a different manner.

The purpose of the INV function is to numerically invert a nonsingular matrix. The preprocessor will generate the coding necessary to check for a singular matrix; however, every use of the INV function will produce this same error processing coding. The user may thus wish to hand-optimize his generated subroutine to remove all but one sequence of this error-checking coding.

Through a special preprocessor variable called DETERM, the user can obtain the determinant of his matrix for subsequent calculations. One possible source of error exists, as follows: Should the user employ the INV function more than once in a particular packet and wish to use several of the determinants, he should save the wanted determinants immediately after their generation. In the generation of the FORTRAN coding produced by interpretation of the INV function the same variable DETERM is used to store the determinant no matter how may times INV is used. Consult Sample Program 2 as to the method of avoiding the error.

Variables forbidden the user and those which may only be used under certain limitations are listed alphabetically in one comprehensive listing in Table 7, which is a compilation of all of the variable names from Tables 1 through 6, and includes the preprocessor-allowed functions, TR, INV, DER, and INT. If an allowed FORTRAN function (Table 1) is used in the definitions of

variables in the Stiffness Matrix Differential or Mass Stiffness Matrix Packets, the functions must be in double-precision form. In all other packets, the functions must be in single-precision form. Tables 1 through 7 may be found on pages 13 through 28.

RESTRICTIONS ON USE OF PACKETS

In adding a new element to NASTRAN, only one packet other than the Preliminary Data Packet is needed—the Stiffness Matrix Packet. If other packets are included, and errors are discovered in these other packets during the preprocessor run (although none are found in the Stiffness Matrix Packet), only the packets with errors will have to be submitted again on a subsequent run. At the user's option, the user's parameter cards, described in subsequent paragraphs, specify (among other items) a preprocessor run number for this dummy element being generated which enables the user to keep track of the number of restarts attempted.

GLOBAL PARAMETER CARDS

These cards provide information which remains pertinent throughout the data deck and guides the preprocessor in the execution of the data. They are, therefore, global parameter cards in the sense that they provide a basic framework for all the user data that follow .

Mandatory Cards

The first set of cards in the user's input data deck is supplied to specify certain program options to the preprocessor. The following four option parameters must be specified by the user either by permitting default values or by providing the appropriate card in the input deck:

- Output parameter
- Run number parameter
- Maximum number of SNOBOL statements to be executed
- Dump parameter

Note that these four mandatory parameters and the other optional parameters described in the subsequent paragraphs may be submitted in any order.

- The Output Parameter

This card specifies how the FORTRAN coding that has been generated is to be printed and/or punched. Punching of the output allows the user to hand-change minor details in the generated coding without rerunning the preprocessor. The format for the card is as follows:

PARAM=I where I is an integer with a value of 1 or 2;

=1 Print, but do not punch the output subroutines and BLOCK DATA subprograms

=2 Print and punch as for a 1 value

A value of 2 should be used only when the user is reasonably certain that the input deck is error-free. The default value is 1.

- The Run-Number Parameter

This card indicates to the preprocessor whether the current run of the program is the initial execution or an update to correct previous errors. This format for the card is as follows:

RUN NUMBER=I, where I is a non-negative integer indicating the run number.

An initial run is indicated by a parameter value of zero. If the data is for an initial run, the preprocessor produces a message stating that fact. An informative message is produced for all nonzero values of the parameter; the parameter itself forms parts of the message. The default value is 0.

- The Maximum Number of SNOBOL Statements to be Executed

This card indicates to the preprocessor how many statements may be executed before a SNOBOL system error occurs. The format for this card is as follows:

LIMIT=I, where I is a positive integer which, if given, should be greater than 400,000, since 400,000 is the default value.

This parameter is used to increase the number of statements allowed when the input data deck is unusually long.

- The Dump Parameter

This card specifies how SNOBOL variables are to be treated upon completion of execution. The format for this card follows:

DUMP=A where A is either the word YES or the word NO.

The default value, NO, indicates that no SNOBOL variables are to have their final values printed when execution ends. The word YES indicates that the SNOBOL variables termination values are to form a part of the output from the preprocessor. This parameter aids in the debugging of data decks when the errors are not readily apparent.

Optional Cards

Six optional parameter cards are available: the PUNCH PRELIMINARY parameter card, the READ PRELIMINARY parameter card, the TIME YES parameter card, the GLOBAL VARIABLES=I card, the PACKET VARIABLES=J card and the DECK SIZE=I card. If the PUNCH PRELIMINARY parameter card is included, the results of the interpretation of the Preliminary Data Packet are punched for submission in all subsequent runs in lieu of the Preliminary Data Packet. Processing of this punched data will marginally decrease total preprocessor execution time.

The READ PRELIMINARY parameter card is used only if the run under consideration is not an initial run, and if the PUNCH PRELIMINARY card was used to punch the interpreted Preliminary Data Packed. Use of the READ PRELIMINARY parameter card indicates that the already-punched Preliminary Data Packet information is to be read rather than the user's original Preliminary Data Packet. Failure to include the READ PRELIMINARY card when the user has substituted the punched Preliminary Data Packet for his original Preliminary Data Packet will result in the preprocessor generating error messages and halting execution during the interpretation of the Preliminary Data Packet.

The user may obtain timing information for the various data packets composing his particular data deck by coding a TIME YES card into his parameter card packet. The maximum number of allowable global and local packet variables may be increased by coding the two cards

```
GLOBAL VARIABLES = I
PACKET VARIABLES = J
```

where I and J are integer values designating the maximum number of global and local packet variables, respectively. The default value in either case is 75.

The maximum number of cards allowable in the user input data deck may be increased by using the card

```
DECK SIZE = I
```

where I is a positive value. The absence of this card limits the user to a maximum of 500 input cards.

All of the cards just described permit free format, in the sense that blanks may be used wherever desired except within the words themselves. Consider the following examples of a parameter card packet

```
PARAMbb = bbl
DUMPbbbbbb = bYES
```

where the small letter b designates a blank character. In this example, the PUNCH PRELIMINARY CARD has been omitted, so the interpreted Preliminary Data Packet will not be punched; the input data deck size is limited to 500 cards; and the maximum number of variables for the global and other packets is 75. Preprocessor output will be printed but not punched, and all SNOBOL variable termination values are printed.

PREPROCESSOR CONTROL CARDS

The various control cards for the preprocessor are indicated here, together with their functions. Some of the cards must be used at least once per input data packet, while others may be used or not as desired.

1. COMMENT X where X is any alphanumeric information which the user may want to code.

This card is useful in the listing of the input deck. It serves to make the data more readable as the user may code comments as he would in a FORTRAN program.

2. BEGIN X where X may be any one of the following character strings (the lower-case letter b representing one or more blanks):

```
GLOBAL
STIFFNESS
VISCOUS b DAMPING
MASS
THERMAL b LOADING
STRESS
STRESS b AND b FORCE
OUTPUT
DIFFERENTIAL b STIFFNESS
```

This card indicates to the preprocessor a new packet in the input data; it also serves as a delimiter between the user packets. If the user has a packet containing an error, the next occurrence

of a BEGIN card for a subsequent data packet would end processing of the incorrect packet and begin processing of the next packet. It must be the first card of each packet with the exception of the Preliminary Data Packet, which is assumed to begin immediately following the parameter cards.

3. END X where X is any desired string of alphanumeric characters. The function of this card is the same as the "COMMENT" control card. It may be used as the last card of a particular packet to indicate the end of a packet. For example, the card "ENDbGLOBALb PACKET" would appear in the listing of the input deck before the next "BEGIN" card to indicate the end of the Global Variable Packet.

4. DEFINITIONS b FINISHED

This card serves as a delimiter between Subpackets B and C within the packets (see page 39); and should be the last card of Subpacket B. It also indicates the end of a FORTRAN insertion subpacket (see page 51).

5. INPUT b FINISHED

This card serves as the delimiter to the entire input deck. It should therefore be the last card of the data deck. It indicates to the preprocessor that all input data has been read and processed.

PACKET SETUP

As already noted, all input information with the exception of the parameter option information is assembled into packets. Each of these packets but two—the Preliminary Data Packet and the Output Packet—is itself composed of three mandatory separate subpackets (Subpacket A, Subpacket B, and Subpacket C) which

are described in the following paragraphs. If the user finds it necessary, he may submit a fourth optional subpacket, the user-supplied FORTRAN insertion subpacket. This subpacket will be defined by the user in Subpackets B and C. The user may supply as many insertion subpackets as he finds necessary.

Subpacket A

Since the form of Subpacket A varies in the different packets, no generalizations can be made other than that it usually consists of only two or three cards. Each Subpacket A will be described in detail in the individual packet descriptions.

Subpacket B

Subpacket B sets the overall specification of all user-defined variables for a particular packet. It consists of a set of cards, each—with the exœption of the last—having the following four fields of information on the data card in the order shown, separated by commas:

Field 1 The variable name

Field 2 The first dimension of the variable (number of rows)

Field 3 The second dimension of the variable (number of columns)

Field 4 The manner of definition for the variable.

The first field, the variable name, has several restrictions. First, the name itself is limited to no more than six characters. Secondly, the name must conform to all FORTRAN specifications for variable names. The variable name may not be one of the preprocessor's reserved words (Table 7). There is no default that the user may assume; the existence of a null field in this location produces a preprocessor error message.

The second and third fields define the dimensions of the variable given in the first field. All user-defined variables therefore assume the form of a two-dimensional matrix. To define a column vector, row, or a scalar variable, a user must define the variable as an 1XN, NX1, or a 1X1 array, respectively. These two fields may be null, the default value being 1. Cards such as

```
W,,,COMM
X,,2,TERM
```

and

define the variable W as a 1X1 scalar, and the variable X with dimensions 1X2. The cards

```
Y,,,EQUA
Z,10,5,TERM
```

define Y as a 1X1 scalar and Z as a 10X5 matrix.

The final field, the manner of definition for the variable, will contain one of the four keywords TERM, EQUA, COMM, or DEFER. TERM indicates that a term-by-term manner of definition will be employed, i.e., each member of the array will be defined separately and each term will be defined with a scalar symbolic equation. In the discussion of the dimensioning of variables Z and X above, each would be defined separately by scalar equations in a subsequent subpacket.

The keyword EQUA is specified when the user wishes to define a given variable by a matrix equation. Here, the user does not define each element of the vector or matrix but defines the entire vector or matrix through a FORTRAN-like matrix equation. Addition, subtraction, multiplication, and exponentiation are permissible, and the variables upon which the operations are performed may be matrices, vectors, scalars, and constants. Other special operators such as the transpose function TR described earlier are also available for use in the equation. A more detailed description of these equations appears in the definition of the next subpacket.

The keyword COMMON declares the associated variable name to be a FORTRAN COMMON variable, the COMMON statement being coded in a user-defined INSERT packet (see page 51). The user therefore does not have to define the variable explicitly in Subpacket C although he may use the variable as he would any other NASTRAN COMMON variable.

The keyword DEFER declares that the user will not define the variable explicitly in Subpacket C as usual but will define it through one or more user-supplied INSERT packets. It is used to define variables which cannot be defined by any other preprocessor method.

The default specification of this fourth field is TERM. The TERM manner of definition is preferred, since the preprocessor will execute far fewer SNOBOL statements to generate the necessary FORTRAN coding than if the EQUA mode of definition were chosen.

The last card of Subpacket B contains the preprocessor control card "DEFINITIONS b FINISHED" as described earlier. It indicates that the subpacket is complete and that no further input data for the subpacket follows. A very simple example of a Subpacket B, which relates to other examples already used, follows:

```
W,,,COMMON
X,,2,TERM
Y,,,EQUA
A,10,5,TERM
DEFINITIONS b FINISHED
```

A more complex example of a Subpacket B is provided in Figure 1. Variables XLV12, XLV13, XX2, YY3, and A are scalars to be defined by the term-by-term method. XX3 is a scalar defined through a matrix equation. Variables V12, V13, XII, XKK1, XKK, and XJJ are 3X1 vectors defined by a term-by-term method; and variables E1, C1, C2, and C3 are 3X2 matrices defined again through the term-by-term method.

```
V12,3,1,TERM
XLV12,,,
V13,3,,TERM
XII,3,,
XKK1,3,1,TERM
XLV13,,,
XKK,3,,TERM
XJJ,3,1,TERM
E1,3,2,
XX2,,,
XX3,,,EQUA
YY3,,,
A,,,
C1,3,2,
C2,3,2,
C3,3,2,TERM
DEFINITIONS FINISHED
```

Figure 1 - Example of Subpacket B

Subpacket B has two restrictions, the first being that each variable name must be a legal one as to FORTRAN conventions (illegal variables are listed in Table 7), and the second pertaining to the order of the variables. The FORTRAN coding will be generated according to the order of the variables as they are given in this subpacket. Therefore, if Variable A is to be defined in terms of Variable B, Variable B must precede Variable A in this subpacket. This requirement is irrespective of whether the variable is to be defined term-by-term or by a matrix equation. Therefore, the correct ordering of the variables in this subpacket is vital to the correct execution of the generated routine.

Subpacket C

This subpacket defines variables set up in Subpacket B. The elements of the variable array may be defined in either of two ways:

(1) term-by-term, or
(2) matrix equation

The data in Subpacket C is given on a series of card sets, one set for each variable to be defined. The appearance of the data sets will vary according to the form of definition used.

Term-by-Term. If the user has specified that the variable is to be defined by the term-by-term method, the card set is as follows. The first card of the set gives the name of the variable to be defined, the name starting in Column 1 for the best optimization of the preprocessor coding. The following set of cards is coded by the user to define all nonzero elements of this array. (All variables are considered to be arrays.) Each card has three fields:

Field 1	First subscript (Default is 1)
Field 2	Second subscript (Default is 1)
Field 3	Definition of the element of the array with these subscripts

The three fields are separated by commas, and the user should left-adjust the fields and compress out all blanks in order to optimize execution. If the element definition extends beyond the first card, the user enters a dollar sign ($) in the Field 3 of each card with the exception of the last. An example of a card set using the sample definition for X from the description of Subpacket B is:

```
X
1,1,A*B+C
,2,X1-X2 $
*X3+Y3
```

where $X_{1,1}$=A*B+C and $X_{1,2}$=X1-X2*X3+Y3

Note the use of the continuation sign and the assumption of the first subscript for $X_{1,2}$. The expressions used in a term-by-term definition may have subscripted variables; indeed, the user is urged to supply both subscripts for all user-defined variables, as the preprocessor will execute faster. The expression is assumed to be a scalar, rather than a matrix expression. The only non-integer subscript allowed is the variable name IP. Its value has meaning only in the Stiffness, Mass, Thermal Loading, and Differential Stiffness Packets. For these packets, IP contains the current NASTRAN pivot point being calculated. For an element with N grid points, the variable employing IP as a subscript must have a dimension of N if the generated subroutines are to execute correctly.

Matrix Equations. If the user wishes to define the variable through a matrix equation, the card set is made up as follows. The first card is punched with the variable name to be defined, starting in Column 1 for fastest execution. The user then codes the matrix expression on the next and succeeding cards, using the same manner of indicating continuation if necessary. The matrix expression may be scalar if the user so desires.

Using the same specifications for Y described in the first sample, Subpacket B, the user might define Y in the following manner:

```
Y
T1*X3+$
X1+Y3
```

Another example of Subpacket C is given in Figure 2, pages 44, 45. This example defines the elements of the variables specified in Subpacket B in Figure 1. Note that there is no delimiter to indicate the end of Subpacket C; the preprocessor will contain to process the input data deck until the next packet is encountered. Every variable used in the definitions of Subpacket C must be defined as to dimension, etc., through the

preceding Subpacket B, unless (1) the variable used is a NASTRAN COMMON variable, and the definition is thus already known to the preprocessor, (2) the variable is global and already defined in the Global Variable Packet, (3) the variable is a preprocessor-defined variable, or (4) the variable is a FORTRAN function.

A special type of matrix equation may be used to generate certain types of variables. A variable may be generated by rows or columns instead of by either a matrix equation or a term-by-term definition. The user declares the variable's definition to be by matrix equation in Subpacket B (keyword EQUA).

```
V12
COMMENT
COMMENT  X1,...,Y1,...,Z1,...  ARE COMMON VARIABLES AVAILABLE FOR USE
COMMENT
1,,X2-X1
2,1,Y2-Y1
3,,Z2-Z1
V13
1,,X3-X1
2,1,Y3-Y1
3,1,Z3-Z1
XLV12
,,DSQRT(V12(1,1)**2+V12(2,1)**2+V12(3,1)**2)
XII
,,V12(1,1)/XLV12(1,1)
2,1,V12(2,1)/XLV12(1,1)
3,1,V12(3,1)/XLV12(1,1)
XKK1
1,1,XII(2,1)*V13(3,1)-XII(3,1)*V13(2,1)
2,1,XII(3,1)*V13(1,1)-XII(1,1)*V13(3,1)
3,1,XII(1,1)*V13(2,1)-XII(2,1)*V13(1,1)
XLV13
1,1,DSQRT(XKK1(1,1)**2+XKK1(3,1)**2)
XKK
1,1,XKK1(1,1)/XLV13(1,1)
2,1,XKK1(2,1)/XLV13(1,1)
3,1,XKK1(3,1)/XLV13(1,1)
XJJ
1,1,XKK(2,1)*XII(3,1)-XII(2,1)*XKK(3,1)
2,1,XKK(3,1)*XII(1,1)-XII(3,1)*Xkk(1,1)
3,1,XKK(1,1)*XII(2,1)-XII(1,1)**XKK(2,1)
E1
1,1,XII(1,1)
2,1,XII(2,1)
3,1,XII(3,1)
1,2,XJJ(1,1)
2,2,XJJ(2,1)
3,2,XJJ(3,1)
XX2
,,XLV12(1,1)
XX3
COMMENT
COMMENT  V13 IS A 3 X 1 VECTOR---TR(V13) IS A 1 X 3 VECTOR
COMMENT  XII IS A 3 X 1 VECTOR---SO XX3 IS A 1 X 1 VECTOR,
COMMENT  IE, A SCALAR
COMMENT
TR(V13)*XII
```

FIGURE 2 - Example of Subpacket C

```
YY3
1,1, XLV13(1,1)
A
1,1,.5*XX2(1,1)*YY3(1,1)
C1
1,1,-1./XX2(1,1)
2,2,C1(3,1)
COMMENT  NOTE THAT ELEMENT (2,2) OF THE ARRAY IS DEFINED IN TERMS
COMMENT  OF ELEMENT (3,1)---THE REVERSE WOULD HAVE CAUSED PROBLEMS
3,1,1./YY3(1,1)*(XX3(1,1)/XX2(1,1)-1.)
3,2,C1(1,1)
C2
1,1,-C1(1,1)
2,2,C2(3,1)
3,1,-XX3(1,1)/(XX2(1,1)*YY3(1,1))
3,2,1./XX2(1,1)
C3
2,2,C3(3,1)
3,1,1./YY3(1,1)
```

FIGURE 2 - Example of Subpacket C - Continued

The first card in Subpacket C for this type of definition will contain the variable name to be defined. On succeeding cards, the user codes the keyword ROW or COLUMN followed by an integer defining the appropriate row or column. After the integer, the user punches a comma followed by the row or column matrix equation definition. This definition will consist of only a variable name, the variable having been defined earlier as a row or column vector. The variable being defined must be defined by either all row definition or all column definitions.

These cards that form the definitions of the variable on Card 1 may be in any order; the user does not have to define all of the rows or all of the columns in either ascending or descending order. The cards are free format. The variable defining each row or column must be a previously-defined packet variable. Full matrix equations are not allowable. An example of this special type of matrix equation follows.

Consider the following example: Let A be the 2X2 matrix to be defined

$$A = \begin{bmatrix} B & D \\ C & E \end{bmatrix} ,$$

where B, C, D, and E are previously defined scalar variables. The user may define A in terms of B, C, D, and E. If we define the two vectors F and G to be

$$F = [B,D] \quad , \quad G = [C,E]$$

then A can be defined in Subpacket C as follows:

```
A
ROW 1, F
ROW 2, G
```

If F and G have definitions

$$F = [B,C] \quad , \quad G = [D,E]$$

we could have defined A as follows:

```
A
COLUMN 1, F
COLUMN 2, G
```

Another special type of matrix equation definition is by partitions. If a variable to be defined is a square matrix, the user may define it by partitions.

On the first data card, the user codes the variable name to be defined. On the second and subsequent cards up to a maximum of four, the user codes one of the following four keywords:

(1) UL (define upper left partition)
(2) UR (define upper right partition)
(3) LL (define lower left partition)
(4) LR (define lower right partition)

After the keyword, a comma appears followed by the partition definition in the form of a previously-defined variable name.

Let A be a four-by-four matrix as follows:

$$A = \left[\begin{array}{cc|cc} a & b & c & d \\ e & f & g & h \\ \hline i & j & k & l \\ m & n & o & p \end{array}\right]$$

If B, C, D, and E can be defined as follows,

$$B = \begin{bmatrix} a & b \\ e & f \end{bmatrix} \quad ; \quad C = \begin{bmatrix} c & d \\ g & h \end{bmatrix} \quad ;$$

$$D = \begin{bmatrix} i & j \\ m & n \end{bmatrix} \quad ; \quad E = \begin{bmatrix} k & l \\ o & p \end{bmatrix}$$

then

using the above example, A could be defined by the set of cards:

A
UL, B
UR, C
LL, D
LR, E

The existence of an upper left partition implies the existence of a lower right partition in the subpacket; an upper right partition being coded means that a lower left partition must exist in the subpacket. If only one of each pair is defined, the missing partition assumes the definition of the supplied partition. For example, to create

$$\left[\begin{array}{cc|cc} a & b & c & d \\ e & f & g & h \\ \hline c & d & a & b \\ g & h & e & f \end{array}\right]$$

with B and C defined as above, the user codes in the appropriate Subpacket C:

A
UL, B
UR, C

The lower left and right partitions will be defined by the preprocessor. The data cards forming the definition are free format and may be in any order.

General Remarks Concerning Subpacket C.

If a variable is to be defined by a term-by-term definition, the value of any element of the variable not given in Subpacket C is defaulted to 0. If a variable is to be defined by a matrix equation, all members of the variable are defaulted to 0 unless the matrix equation definition of the variable appears in Subpacket C.

If the user wishes to define an element of an array in terms of another element of the same array, the following two rules must be observed:

- If the two elements have different column numbers (second subscripts), the element with the larger column number may be defined in terms of the element with the smaller column number, but not vice versa. For example,

  A(2,2) = A(3,1)

  is acceptable, but

  A(3,1) = A(2,2)

  is not.

- If the two elements have the same column number, the element with the larger row number (first subscript) may be defined in terms of the element with the smaller row number, but not vice versa. For example,

  A(5,4) = A(2,4)

  is acceptable, but

  A(2,4) = A(5,4)

  is not.

In other words, the preprocessor takes the user's term-by-term definitions and arranges them so that a column-wise definition appears.

The following restriction in the use of matrix equations must be observed. Stated briefly it is this: No more than nine subexpressions may be coded in any one matrix equation definition. The following example containing ten subexpressions illustrates what happens when this restriction is ignored.

Example:
Let the user-variable A have the following definition:
A = (B+O)*(D+E)*(F+G)*(H+W)*(Z+X)*(O+P)*(Q+R)*(S+T)*(U+V)*(B+C)

where B, C, D, E, F, G, H, O, P, Q, R, S, T, U, V, and W are previously defined variables. Further, assume that these variables are properly dimensioned so that the equation is legal.

Each of the expressions (B+C), (D+C), etc., is called a subexpression. Through the Polish notation convertor built into the preprocessor, the definition of A becomes

```
A = B, O+D, E+F, G+H, W+Z, X+O, P+Q, R+S, T+U, V+B
    C + *********
```

Through the preprocessor logic, all the subexpressions in this particular equation must be resolved before the series of multiplications can be performed. Nine intermediate variables not available to the user—Q1, Q2,..., Q9—are used to store these subexpressions. In the preceding equation, the variables Q1, Q2,..., Q9 are defined as follows:

```
Q1 = B+O                    Q5 = Z+X
Q2 = D+E                    Q6 = O+P
Q3 = F+G                    Q7 = Q+R
Q4 = H+W                    Q8 = S+T
                Q9 = U+V
```

During interpretation, the above equation temporarily becomes

```
Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8, Q9, B, C+*********
```

The resolution of the subexpression B+C transforms the equation to

```
Q1, Q2, Q3, Q4, Q5, Q6, Q7, A8, Q9, Q1*********
```

Now the variable Q1 appears twice in the equation. Furthermore, it has the new definition Q1 = B+C. But the user desired Q1 to be the expression B+O at the beginning of the original equation. His equation has been transformed into

```
A = (B+C)*(D+E)*(F+G)*(H+W)*(Z+X)
A = (A *(O+P)*(Q+R)*(S+T)*(U+V)*(B+C))
```

which was not his intent. Unless the user consciously looks for it, he could easily overlook an error of this type. Nor will FORTRAN compilation of the generated coding detect the error.

To operate within this restriction, the user should break up large equations having more than nine subexpressions into smaller equations. In the above example, resolution could take the following form.

```
A1 = (B+0)*(D+E)*(F+G)*(H+W)*(Z+X)
A  = A1*(0+P)*(Q+R)*(S+T)*(U+V)*(B+C)
```

which would produce proper coding.

FORTRAN USER-SUPPLIED INSERTION SUBPACKET

Some FORTRAN coding situations are not amenable to the techniques described earlier. For example, there is no way to code the FORTRAN IF statements which may be used to test for 0-value denominators or to call a user-supplied subroutine. FORTRAN statements may be incorporated within the generated coding in one of two ways: either directly, through manual insertion by the user; or indirectly, by having the preprocessor code it along with the rest of the routine still remaining to be coded.

Since the order of specification of the variables in Subpacket B of a packet defines the order in which the coding is generated (see page 40), the inclusion in Subpacket B of the string

```
INSERT X
```

where X is a character string unique to this packet, will cause the preprocessor to code the FORTRAN statements which appear in Subpacket C following an INSERT X string in that subpacket. The preprocessor will consider all cards following the INSERT card to be part of the insertion subpacket until a DEFINITIONS FINISHED preprocessor control card is encountered. The

DEFINITIONS FINISHED control card in this circumstance acts as a terminator to the INSERT subpacket. For example, suppose in Subpacket B we have the instructions

```
A, 1, 1, TERM
B, 1, 1, TERM
INSERT Q
C, 1, 1, TERM
INSERT V
```

and in Subpacket C we have

```
INSERT Q
IF(A.EQ.B) GO TO 9105
DEFINITIONS FINISHED
INSERT V
GO TO 9110
9105 C = 0
9110 CONTINUE
DEFINITIONS FINISHED
```

After the definitions of the variables A and B have been generated, the coding will be

```
          IF(A.EQ.B) TO TO 9105
          C = (normal definition for C)
          GO TO 9110
9105      C = 0.
9110      CONTINUE
```

Note the following important points:

1. The preprocessor handles INSERT X as though it were a variable. Therefore, if several different insertions are desired, unique X strings will be required.

2. In the Stress Matrix and the Stress and Force Calculation Packets, variables appear on an options card in subpacket A as well as in Subpacket B. The variables in Subpacket A are always

generated after the secondary variables which appear in Subpacket B. Therefore, if coding is to be inserted after the definition of a variable which appears in Subpacket A for these two packets, then the INSERT X string must appear in Subpacket A immediately following the variable. However, this INSERT X string should not be counted as a variable in the count that must also be given in Subpacket A. For example, in Figure 9, the variable THETA will be computed using the arctangent function. Since THETA is a function of the elements of the SIG array and since we wish to test elements of the SIG array, the INSERT X strings must appear in Subpacket A. But note that the INSERT X strings are not included in the count on the preceding card.

If coding is to be inserted during generation of the secondary variables in Subpacket B, the INSERT X string should appear in the appropriate place in subpacket B.

3. The user-supplied FORTRAN statements will be coded exactly as they are punched. Therefore, all FORTRAN rules, such as starting the coding in column 7, must be observed. The preprocessor will not check for FORTRAN syntax errors.

4. Since the subscripts of an array are normally switched by the preprocessor due to NASTRAN restrictions, it would be wise for the user to set up a dummy variable if he desires to add coding containing a subscripted variable.

5. Care should be exercised by the user in supplying FORTRAN statement numbers. Statements numbers should be greater than or equal to ((2(N+1)+1)100), where N is the number of grid points of the element under consideration. The user should automatically check the generated routine for such obvious errors as duplicate statement numbers, since such errors can easily be directly corrected.

The user may also insert FORTRAN DIMENSION, COMMON, and EQUIVALENCE statements into his generated subroutines. He inserts FORTRAN statements after the preprocessor-generated DIMENSION statements by coding an INSERT DIMENSION X packet with the same restrictions as for the INSERT X packet. Statements may be added after the preprocessor-generated COMMON and EQUIVALENCE statements by coding INSERT COMMON X and INSERT EQUIVALENCE X packets, respectively. See Sample Problem 2 in the Appendix B for an example of this procedure.

III. DATA PACKET DESCRIPTIONS

Descriptions of each of the ten packets follow. Examples of each packet are included, unless the packet is identical to another. Tables of variable names which may only be used as already defined are provided on pages 13-28. Names denied the user are also listed.

PRELIMINARY DATA PACKET

Logical Cards

The information specified in this packet is used to update NASTRAN tables and to create FORTRAN COMMON statements that will be of use to the user in his element subroutines. This packet must contain three logical cards, each of which may consist of several physical cards. The following items of information must be supplied:

Card 1

Field 1	Element name
Field 2	Number of grid points for the element
Field 3	Sequence of digits specifying the degrees of freedom for the element
Field 4	Indicator as to whether the element is a scalar element or a structural element
Field 5	Number of permanent elements already residing in the NASTRAN element library
Field 6	Approach acceptability flag for the element
Field 7	Number of data items for the element listed on the connection card
Field 8	Number of data items for the element listed on the property card
Field 9	Sequence of digits for the logical connection card (Card 2)
Field 10	Sequence of digits for the logical property card (Card 3)

Card 2

The user-defined connection-card variables

Card 3

The user-defined property card variables.

The user enters all of the fields 1 through 10 on the first logical data card in the packet. If the total length of these fields is longer than one physical card (80 columns), the user must continue on a subsequent card(s), entering a dollar sign ($) anywhere in the last field of the card which is to be continued. All of the cards but the last must contain the dollar sign. To avoid overlooking this requirement, the user might do well to form a habit of coding a $ in a particular column of a card—say column 72, for example. Examples of the use of the dollar sign convention are provided in the packet descriptions.

Description of Logical-Card Data Fields

Card 1

Field 1, the element name, must be entered in the form ELEMx, where x signifies any integer less than 51 and greater than the number of permanent elements already residing in NASTRAN. (See the description of Field 5). After NASTRAN has been successfully updated, the NASTRAN bulk data connection card and property card mnemonics will be CELEMx and PELEMx, respectively. The integer quantity x will be a subscripting parameter in the later generation of BLOCK DATA subprograms. The default value of x in ELEMx is (Field 5) + 1.

Field 2 must contain an integer which specifies the number of grid points for the element. There is no default value for this parameter. The largest value allowed is 100.

Field 3 specifies the degrees of freedom that the element may take. This field consists of a string of as many as six integers, which may be any combination of the integers 1, 2, 3, 4, 5, 6. These integers refer to NASTRAN's present degrees-of-freedom. Integer 1, 2 and 3 refer to the x, y, and z translations, respectively; integers 4, 5, and 6 refer to the rotations about the x, y, and z axes, respectively. An integer may not appear more than once in the field. Thus, a value of 123456 indicates that the element has all six degrees of freedom; the string 12 indicates that only translations in the x and y directions are possible. There is no default value for this field.

Field 4 contains the scalar indicator, which is an integer with three permissible values: -1, 0, or 1. The default value is 0, which implies that the element is a structural, rather than scalar, element. A value of 1 indicates that the element is a scalar element with grid point and component code. A value of -1 indicates that the element is a scalar element with scalar points only. At the present time only the structural element may be added via the preprocessor, so the value in this field must be 0.

Field 5 indicates the number of permanent elements already residing in NASTRAN. The default value is 38 (the number of elements in the Level 12.0 general release version). The number of elements in Level 15 is 61. This field prevents the user from generating FORTRAN coding which will then overlay parameter values of existing NASTRAN elements.

Field 6 contains the approach acceptability flag for the new element. This variable must be an integer ranging between the values -2 and +2. The default value of 0 indicates that any approach is allowable for the element. The values -2 and +2 indicate that the force and displacement approaches, respectively, are illegal for the element. The values -1 and +1 indicate that the element is not used by the force or displacement approaches respectively. Since NASTRAN does not contain the Force approach at the present time, the parameter in this field should be 0.

Fields 7-10 refer to the connection card variables (CV1, CV2,...) and the property card variables (PV1, PB2,...) described on page 100. The integer in Field 7 specifies how many user-defined data items are contained on the connection card for the new element. (The variable names themselves are submitted on Card 2 of this packet.) The integer supplied must be greater than 0 even though there is no default value, since the anisotropic material angle must be specified in any case (page 102). The user may use these variables in the definitions of variables in other packets. He supplies the various connection variables values to NASTRAN through the NASTRAN bulk data connection card CELEMx.

Field 8 contains an integer which specifies how many user-defined data items are contained on the property card for the new element. This parameter may have a 0 value, indicating that the element does not make use of a property card—in which case Card 3 would be omitted from the Preliminary Data Packet. If so, the element material property ID must then be specified on the connection card (page 100). The property variables themselves are defined on the NASTRAN bulk data property card PELEMx. The sum of the integers in Fields 7 and 8 must not exceed 97-5N, where N is the number of grid points for the element.

Field 9 must contain a sequence of n integers, with n the number specified in Field 7. Each integer indicates the type of the user-defined variable which is located in the corresponding position on the connection card. Thus the first integer supplies the type of the first variable located on Card 2, the second integer supplies the type of this second integer on Card 2, and so on. Commas must not be used to separate the integers. The type of integers and the variable types they stand for in NASTRAN are as follows:

1	Integer
2	Real
3	BCD
4	Double precision
5	Anything not covered by 1, 2, 3 or 4

At the present time only the integer, real, and BCD representations will be recognized, although future expansion of the preprocessor will enable any of the various variable types listed to be used. At this time the user's sequence will be made up of a combination of 1's, 2's, and 3's. The default value for this field is n number of 2's.

Field 10 is similar to Field 9 except that it concerns the user-defined property card variables listed on Card 3 rather than the connection card variables given on Card 2. The number of integers must be m, with m the number indicated in Field 8. Field 10 must be empty if Field 8 contains a 0 value.

Card 2

Card 2 contains the list of user-defined connection card variables for the new element. The user is presently restricted to using only real, integer, and BCD variables. Variables beginning with the letters I, J, K, L, M, or N will become real variables. The variables must be separated by commas. Continuation onto other cards is allowed if the $ punch is used.

Card 3

If Field 8 has a non-zero value, the user then supplies Card 3 which lists the user-defined property card variables PV1, PV2,... . The format is the same as that for Card 2. The variables used on this card may not be listed on Card 2, and vice versa. If the value in Field 8 is 0, this card must be omitted.

Preliminary-Data-Packet Example and Discussion

In the example of Figure 3, Fields 4, 5, and 6 of Card 1 assume the default values 0, 38, and 0, respectively. The sample element is to have the name ELEM40 with three grid points; translation in the x and y directions is allowable. Fields 7 and 8 indicate one connection card variable and two property card variables. Fields 9 and 10 indicate that these variables are to be real variables.

Card 2 specifies that the user-defined connection card variable is to be named TH. Card 3 specifies that the user-defined property card variables are to be named T and FMU.

```
COMMENT
COMMENT BEGIN PRELIMINARY DATA
COMMENT
ELEM40, 3, 12,,,, 1, 2, 2, 22
COMMENT CONNECTION CARD VARIABLE
TH
COMMENT PROPERTY CARD VARIABLES
T, FMU
END PRELIMINARY DATA PACKET
```

Figure 3 - Example of a Preliminary Data Packet

GLOBAL VARIABLE PACKET

This packet sets up "global" variables—variables which may be used in several different packets without being redefined in each packet. The Global Variable Packet has the following four parts:

Preprocessor control cards
Subpacket A
Subpacket B
Subpacket C

Preprocessor Control Cards

The first card of the packet may only contain the string BEGINGLOBAL. This control card indicates to the preprocessor that processing of the Global Variable Packet is to begin. Blanks may be embedded anywhere on this card, even within the string

itself, as in the example BEGINbGbLbObBbAbL in which the letter b represents a blank. This feature, which applies to all the packets, facilitates the reading of the input data. The preprocessor control card COMMENTx, where x is any phase desired by the user, can appear anywhere within the packet.

Subpacket A

Subpacket A consists of one logical card (which may encompass one or more physical cards) which lists all of the variables that are to be defined as global variables. Subpacket A may consist of a blank card. If any variables are listed, they must be separated by commas. The list may overflow onto successive data cards if the dollar-sign convention is observed. For example, the following notation

```
X12, V13, XII, $
VAR1, VAR2
```

specifies that five variables are to be defined in the Global Variable Packet. Use of the second card is allowable, since a dollar sign has been punched on the first card. There is one restriction to this initial specification of variables, however. The user must list variables in the order in which they are to be used in the definitions. Suppose V13 and VAR1 are to be used for the definition of VAR2. Then V13 and VAR1 must be punched prior to VAR2 on the card. The variable names must conform to all FORTRAN rules for variable name construction.

If Subpacket A contains only a blank card, the order of the variables in Subpacket B (discussed on page 40) is critical, which is true in all of the packets.

All of the variables listed in Subpacket B are considered global variables, whether or not Subpacket A lists any variables. Consequently, if any of the Subpacket B variables are used in the

packets that follow without first being redefined, the preprocessor will use the Global Variable Packet definition. Consider the example of a global variable A which is redefined in the Stress Matrix Packet. Each time the variable A is used in the packets preceding the Stress Matrix Packet—the Stiffness Matrix, the Mass Matrix, the Viscous Damping, and the Thermal Loading Packets—the preprocessor will resort to the Global Variable Packet for the definition of A. Note that once a global variable has been redefined, the new definition remains for any packets that follow.

Subpacket B

In Subpacket B, the user specifies the dimensions and manner of definition for each of the variables to be defined. Variables listed in Subpacket B which did not appear in Subpacket A must also be defined. Thus, if Subpacket A consisted of only a blank card, the user would have to define all the new variables in Subpacket B, using the proper order described earlier.

Subpacket C

In Subpacket C, the user actually defines all the variables symbolically, either term-by-term or with a matrix equation. All Subpacket B variables not defined in Subpacket C assume the default value of 0. If a matrix equation definition is supplied, it overrides the default value. However, a term-by-term definition will override only those particular array elements which the user has specified. An example of the use of the zero default in a term-by-term definition is for the case in which A is to be a 2X2 diagonal matrix. The user need only code A(1,1) and A(2,2) since all of the other elements left undefined will assume the default value of 0.

Global Variable Packet Example and Discussion

A complete example of a Global Variable Packet is provided in Figure 4. In this example the variables V12, V13, XII, XKK1, XKK, and XJJ are all 3X1 vectors to be defined term-by-term. XLV12, XX2, YY3, XLV13, and A are scalars to be defined term-by-term; XX3 is a scalar to be defined through a matrix equation. Finally, E1, C1, C2, and C3 are 3X2 matrices to be defined term-by-term. Note the use of the zero default in Subpacket C for a term-by-term definition to set C1(1,2), C1(2,1), C2(1,2), C2(2,1), C3(1,1), C3(1,2), C3(2,1) and C3(3,2) to zero. All term-by-term definitions are actual FORTRAN expressions involving scalars, while the matrix equation definition for XX3 involves operations upon matrices (a transpose and multiplication of two previously defined matrices). The variables listed in Tables 1 through 6 are available to the user, subject to the restrictions noted earlier. Thus, COMMON variables X1, X2, X3, Y1, Y2, Y3, Z1, Z2, and Z3 are defined in Table 3.

```
BEGIN GLOBAL
COMMENT GLOBAL VARIABLE LIST
V12,XLV12,V13,XII,XKK1,XLV13,XKK,XJJ,E1,XX2,XX3,YY3,A,C1,C2,C3
V12,3,1,TERM
XLV12,,,
V13,3,,TERM
XII,3,,
XKK1,3,1,TERM
XLV13,,,
XKK,3,,TERM
XJJ,3,1,TERM
E1,3,2,
XX2,,,
XX3,,,EQUA
YY3,,,
A,,,
C1,3,2,
C2,3,2,
C3,3,2,TERM
DEFINITIONS FINISHED
V12
COMMENT
COMMENT  X1,...,Y1,...,Z1,...  ARE COMMON VARIABLES AVAILABLE FOR USE
COMMENT
1,,X2-X1
2,1,Y2-Y1
3,,Z2-Z1
V13
1,,X3-X1
2,1,Y3-Y1
3,1,Z3-Z1
XLV12
,,DSQRT(V12(1,1)**2+V12(2,1)**2+V12(3,1)**2)
XII
,,V12(1,1)/XLV12(1,1)
2,1,V12(2,1)/XLV12(1,1)
3,1,V12(3,1)/XLV12(1,1)
XKK1
1,1,XII(2,1)*V13(3,1)-XII(3,1)*V13(2,1)
2,1,XII(3,1)*V13(1,1)-XII(1,1)*V13(3,1)
3,1,XII(1,1)*V13(2,1)-XII(2,1)*V13(1,1)
XLV13
1,1,DSQRT(XKK1(1,1)**2+XKK1(2,1)**2+XKK1(3,1)**2)
XKK
1,1,XKK1(1,1)/XLV13(1,1)
2,1,XKK1(2,1)/XLV13(1,1)
3,1,XKK1(3,1)/XLV13(1,1)
```

Figure 4 - Example of a Global Variable Packet

```
XJJ
1,1,XKK(2,1)*XII(3,1)-XII(2,1)*XKK(3,1)
2,1,XKK(3,1)*XII(1,1)-XII(3,1)*XKK(1,1)
3,1,XKK(1,1)*XII(2,1)-XII(1,1)*XKK(2,1)
E1
1,1,XII(1,1)
2,1,XII(2,1)
3,1,XII(3,1)
1,2,XJJ(1,1)
2,2,XJJ(2,1)
3,2,XJJ(3,1)
XX2
,,XLV12(1,1)
XX3
COMMENT
COMMENT  V13 IS A 3 X 1 VECTOR---TR(V13) IS A 1 X 3 VECTOR
COMMENT  XII IS A 3 X 1 VECTOR---SO XX3 IS A 1 X 1 VECTOR,
COMMENT  IE, A SCALAR
COMMENT
TR(V13)*XII
YY3
1,1, XLV13(1,1)
A
1,1,.5*XX2(1,1)*YY3(1,1)
C1
1,1,-1./XX2(1,1)
2,2,C1(3,1)
COMMENT  NOTE THAT ELEMENT (2,2) OF THE ARRAY IS DEFINED IN TERMS
COMMENT  OF ELEMENT (3,1)---THE REVERSE WOULD HAVE CAUSED PROBLEMS
3,1,1./YY3(1,1)*(XX3(1,1)/XX2(1,1)-1.)
3,2,C1(1,1)
C2
1,1,-C1(1,1)
2,2,C2(3,1)
3,1,-XX3(1,1)/(XX2(1,1)*YY3(1,1))
3,2,1./XX2(1,1)
C3
2,2,C3(3,1)
3,1,1./YY3(1,1)
END GLOBAL PACKET
```

Figure 4 - Example of a Global Variable Packet—Concluded

STIFFNESS MATRIX PACKET

This packet sets up the variables necessary to generate an element stiffness matrix. The user has available to him the variables contained in common blocks MATIN, MATOUT, SMA1ET, SMA1IO, SMA1CL and SMA1DP. The packet has four parts:

Preprocessor control cards
Subpacket A
Subpacket B
Subpacket C

Preprocessor Control Cards

The preprocessor control cards are similar to those in the Global Variable Packet, except that the first card must be the string BEGINSTIFFNESS instead of the string BEGINGLOBAL.

Subpacket A

Subpacket A consists of two cards. The first is the Stiffness Matrix Packet option card which contains two fields:

Field 1 An INFLAG value
Field 2 An indication as to the method used for the stiffness matrix definition

Card 1

The value of INFLAG in Field 1 governs access to certain NASTRAN COMMON variables which can be of help to the user in building his definitions. These variables are related to the material properties for the element. The INFLAG value must be one of the integers 1, 2, 3, or 4. There is no default value. The variables which may be used for a particular value of INFLAG are indicated in Table 5. If the element uses only isotropic materials, a value of 1 should be used. However, if the element may use either isotropic or anisotropic materials, the integer 2 should be specified. Table 6 lists the definitions for the reserved variable G for INFLAG values of 1, 2, 3, and 4.

Field 2 must contain one of the two keywords K or KIJ. The variable K indicates to the preprocessor that all diagonal and symmetric 6X6 partitions of the element stiffness matrix are to be defined separately. For instance, if the number of grid points specified in the Preliminary Data Packet is 2, the user must defined K11, K12, K21, and K22 separately in Subpackets B and C. The only exception is for that case in which a partition is to be identically zero. If the keyword KIJ is used, the user specifies the partitions by defining KIJ, the general (I,J) stiffness matrix partition. The keyword KIJ implies that the user will be defining the stiffness matrix in Subpacket C by means of a matrix equation. The equation used will be different from all other matrix equations in one important respect: All variables ending with either I or J will represent a series of variables. I and J assume different values 1,2,.., N, where N is the number of grid points. To illustrate, assume the following form for the general matrix partition KIJ in Subpacket C:

KIJ = TR(C1*TR(E1)*TI)*G*(CJ*TR(E1)*TJ)

If the value of N were 2, the following four equations would actually be set up:

$$K11 = (C1*E1^{T}*T1)^{T}\ G(C1*E1^{T}*T1)$$
$$K12 = (C1*E1^{T}*T1)^{T}\ G(C2*E1^{T}*T2)$$
$$K21 = (C2*E1^{T}*T2)^{T}\ G(C1*E1^{T}*T1)$$
$$K22 = (C2*E1^{T}*T2)^{T}\ G(C2*E1^{T}*T2)$$

Note that the variables C1, C2, and E1 must have been defined in Subpackets B and C as to dimension, manner of definition, and actual definition. The variables G, T1, and T2 were predefined in the preprocessor for the user's convenience via Tables 1 to 6.

Card 2

The user codes the right-hand side of a stiffness matrix equation (hereafter to be known as a stiffness matrix expression) on the second card of Subpacket A, continuing onto additional cards as needed by using the dollar-sign convention. The following several paragraphs are devoted to a discussion of the stiffness matrix expression, its restrictions, and its implications. Any reference to a 6X6 partition will also apply to an alternate 1X1 or 3X3 partition.

The stiffness matrix expression must contain the variable K or the variable KIJ. KIJ represents a 6X6 matrix partition; K represents a 6NX6N stiffness matrix with N the number of grid points of the element. Whichever one is used, it must be defined in Subpackets B and C. The KIJ or K used within the stiffness matrix expression indicates what is being computed—either a 6X6 partition or a 6NX6N matrix, respectively. The variable used must relate to the value supplied in the second field of the options card as follows: If K is used in the stiffness matrix expression, the user has no choice but to punch K in the options card. If KIJ is chosen for the stiffness matrix expression, the user may choose to punch either K or KIJ in the second field of the options card, depending upon whether he wishes to define the matrix partitions K11, K12,..., individually (in which case he will choose K), or whether he wishes to define the matrix partitions by means of a master equation (in which case he will choose KIJ). In any case, he has a choice as to the way his matrix partitions will be dimensioned—as 1X1's, as 3X3's or as 6X6's, according to the options he supplies.

As previously mentioned, the punching of KIJ on the stiffness matrix expression and in the second field of the options card implies that the matrix partitions K11, K12, etc., will be defined by a master definition of the variable KIJ in which I and J will be integers within the closed interval [1,N]

with N the number of grid points of the element. Therefore, the user must define KIJ as a 1X1, 3X3, or 6X6 matrix defined by a stiffness matrix equation (keyword EQUA) in Subpacket B. In Subpacket C, the user provides the master equation definition of KIJ. All variables ending with I or J stand for a series of variables.

If the user has the option of placing variables in either the stiffness matrix expression or in the definition of variable KIJ in Subpacket C, he should choose to place them in the definition of KIJ. This placement will result in fewer FORTRAN statements in the preprocessor-generated stiffness matrix subroutine. Therefore, the stiffness matrix expression could become just K or KIJ, if the entire definition of the stiffness matrix or the stiffness matrix partitions could be specified in Subpacket C.

Subpacket B and C

After punching the stiffness matrix expression in Subpacket A of his Stiffness Matrix Packet, the user punches Subpackets B and C to completely define all variables used to build the stiffness matrix or its partitions.

Stiffness Matrix Packet Example and Discussion

In the example of Figure 5, all undefined variables are either global or reserved. The global variables were listed in Figure 4. INFLAG has a value of 2. The user has selected KIJ as his option, thus implying that KIJ will be used in the stiffness matrix expression, and that variables ending in I or J will represent N number of variables. The stiffness matrix expression is A*T*KIJ, with KIJ defined as a 3X3 matrix in Subpacket B. As already explained on page 7, each matrix partition will subsequently be inserted into a 6X6 partition. Finally, the

definition of the general (I,J)th stiffness matrix partition KIJ is given in Subpacket C. Since this new element will have three grid points, as indicated in the Preliminary Data Packet example (Figure 3), the definition given will generate nine separate equations—equations in which I and J in the variables CI, CJ, TI, and TJ take on different values 1, 2, 3. In Figure 4, the variable E1, C1, C2, and C3 are defined. Table 4 indicates that the reserved variables T1, T2, and T3 are available for use.

```
BEGIN STIFFNESS
2,KIJ
COMMENT  STIFFNESS MATRIX EXPRESSION FOR THE (I,J)TH PARTITION
A*T*KIJ
COMMENT  BEGIN LISTING OF STIFFNESS MATRIX PACKET VARIABLES
KIJ,3,3,EQUA
DEFINITIONS FINISHED
KIJ
TR(CI*TR(E1)*TI)*G*(CJ*TR(E1)*TJ)
END STIFFNESS PACKET
```

Figure 5 - Example of a Stiffness Matrix Packet

MASS MATRIX PACKET

The format and restrictions for this packet are identical to those set out for the Stiffness Matrix Packet just described, except that the first card will now contain the string BEGINMASS, and the variables MIJ and M will be used instead of the variables KIJ, and K, respectively. The COMMON's used by the routine generated from this packet will be MATIN, MATOUT, SMA2ET, SMA2IO, SMA2CL, and SMA2DP (Table 3). Table 4 lists the other reserved variables available for use with this packet.

Mass Matrix Packet Example and Discussion

In the example of Figure 6, the mass matrix expression is MIJ. This fact, together with the M option supplied on the first card of Subpacket A indicates that the user will supply specifications for M11, M12,..., MNN separately for each non-zero MIJ in Subpackets B and C. In Subpacket B, one scalar variable XMASS is supplied, in addition to the mass matrix partitions. Note that several partitions are unspecified and consequently will be assumed to be identically zero. Subpacket C indicates that the matrix partitions M11, M22, and M33 are diagonal 3X3 matrices with XMASS on the diagonals. The definition of A is supplied from the Global Variable Packet (Figure 4). The COMMON variable RHO's definition is found in Table 5. Finally, FMU and T are user-defined property card variables already made available in the Preliminary Data Packet, Figure 3.

```
BEGIN MASS
COMMENT  SUBPACKET A
2,M
COMMENT MASS MATRIX MATRIX EQUATION
MIJ
COMMENT  SUBPACKET B
COMMENT BEGIN LISTING OF MASS MATRIX PACKET VARIABLES
XMASS,,,
M11,3,3,TERM
M22,3,3,
M33,3,3,TERM
DEFINITIONS FINISHED
COMMENT  SUBPACKET C
COMMENT  PARTITIONS M12,M13,M21,M23,M31,AND M32 ARE IDENTICALLY ZERO
XMASS
1,1,A*(RHO*T+FMU)/3.
M11
1,1,XMASS(1,1)
2,2,XMASS(1,1)
3,3,XMASS(1,1)
M22
1,1,XMASS(1,1)
2,2,XMASS(1,1)
3,3,XMASS(1,1)
M33
1,1,XMASS(1,1)
2,2,XMASS(1,1)
3,3,XMASS(1,1)
END MASS PACKET
```

Figure 6 - Example of a Mass Matrix Packet

VISCOUS DAMPING MATRIX PACKET

This packet is identical to the Mass Matrix Packet as to format and restrictions, except that the first card of the packet will contain the string BEGIN VISCOUS DAMPING (with blanks embedded as needed), and the variables VIJ and V will be used instead of the variables MIJ and M. The COMMON's and the reserved variables available are the same for both packets. Because of the similarity of this packet to others already described, an example has not been included.

THERMAL LOADING VECTOR PACKET

The format for the Thermal Loading Vector Packet is similar but not identical to that for the Stiffness Matrix Packet. The first card will contain the string BEGINbTHERMALbLOADING. The INFLAG variable on the Subpacket A options card operates in the usual way. The second option will be either PPI or PP, with PPI having the same function as KIJ (except that variables ending on J will no longer carry any special significance) and PP will function in the same way as K.

The second card of Subpacket A will contain an expression for the thermal loading vector using either the variable PPI or PP. Just as KIJ must be a 1X1, a 3X3, or a 6X6 matrix, and K must be a 6NX6N matrix in the stiffness matrix expression, so must PPI be a 1X1, a 3X1, or a 6X1 vector. PP must be a 6NX1 vector with N the number of grid points. The thermal expansion coefficient vector will likely be used in the definition of PP to obtain the final thermal loading (as is true in the example in Figure 7 in which ALPHV is the thermal expansion coefficient). Remember that when PPI is present on both cards of Subpacket A, variables ending in J represent only themselves, and not several other variables.

If PPI is used in the thermal loading expression, the user can choose between P or PPI for his options card. However, if PP is used in the expression, PP must be chosen for the options card. Whichever variable is chosen for the thermal loading expression must then be defined in Subpackets B and C.

The user has access to the variable TTI, a vector which contains temperatures at the different grid points of the element. For example, the variable TTI(I) would indicate the temperature at the Ith grid point of the element. Note that in the example of Figure 7, the variable TBAR in Subpacket C is the average of the temperatures at the various grid points minus some reference temperature TSUBO which enters through the named COMMON block MATOUT (Table 5).

The COMMONs used by the routine generated from this packet are MATIN, MATOUT, EDTSP, and TRIMEX (Table 3). Other reserved variables available for use in this packet are those listed in Table 4.

Thermal Loading Packet Example and Discussion

On the first card of Subpacket A in this example of Figure 7, the value of INFLAG is 2. The presence of PPI implies that the general definition of the Ith vector partition will be specified. The thermal loading vector expression is

A*T*PPI*TBAR

which implies the following three equations:

A*T*PP1*TBAR
A*T*PP2*TBAR
A*T*PP3*TBAR

The variable A has already been defined in the Global Variable Packet (Figure 4). T is a property card variable which was listed in the Preliminary Data Packet, Figure 3. PP1, PP2, PP3, and TBAR are to be defined through Subpackets B and C (although according to existing options, only PPI need be specified).

In Subpacket B, ALPHV is defined in terms of the COMMON variables ALPHA1, ALPHA2, and ALP12. (See Table 5.) TBAR's definition is formed from TSUBO and TTI. TSUBO comes from the COMMON variable TSUBO (Table 5). The temperature vector TTI (Table 4) is predefined by the preprocessor. PPI, the general thermal loading vector, has the definition TR(TI)*E1*TR(C1)*G*ALPHV. Since PPI is used on both cards of Subpacket A, it in turn defines the equations

PP1 = TR(T1)*E1*TR(C1)*G*ALPHV
PP2 = TR(T2)*E1*TR(C2)*G*ALPHV
PP3 = TR(T3)*E1*TR(C3)*G*ALPHV

The variables E1, C1, C2, and C3 are defined through the Global Variable Packet (Figure 4). The variables T1, T2, T3 are the preprocessor-defined transformation vectors (Table 4).

```
BEGIN THERMAL LOADING
2,PPI
COMMENT THERMAL LOADING EQUATION
A*T*PPI*TBAR
ALPHV,3,1,TERM
TBAR,,,
PPI,3,,EQUA
DEFINITIONS FINISHED
ALPHV
1,1,ALPHA1
2,1,ALPHA2
3,1,ALP12
TBAR
1,1,(TTI(1)+TTI(2)+TTI(3))/3.-TSUBO
COMMENT  ALPHA1,ALPHA2,ALP12,TSUBO COME FROM COMMON MATOUT
COMMENT  TTI IS THE VECTOR OF GRID POINT TEMPERATURES
PPI
TR(TI)*E1*TR(CI)*G*ALPHV
END THERMAL LOADING
```

Figure 7 - Example of a Thermal Loading Packet

STRESS MATRIX PACKET

This packet generates a FORTRAN subroutine that is used to pass necessary variables to a routine that will compute element stresses and forces. These variables are passed as elements of the NASTRAN array PH1OUT. The preprocessor generates a series of FORTRAN EQUIVALENCE statements to perform this function.

The first card in the data pack must contain the string BEGINSTRESS with blanks wherever convenient.

Subpacket A

Card 1

The next card in the data packet, the first of Subpacket A, will be the options card. The INFLAG option values are the same as for the Stiffness Matrix Packet already described. The second option will contain an integer quantity that specifies the number of words to be loaded into the PH1OUT array, calculated as follows. Suppose that there are a total of nine elements in the variable arrays on the stress matrix card (discussed next). To this number add N+1, in which N represents the number of grid points. The sum will be the value to be supplied for the second option. This number must be less than or equal to 100, or the NASTRAN table will overflow.

Card 2

The second card of Subpacket A is the stress matrix card which must contain a list, in proper order, of all the variables the user wishes to pass to the stress and force calculation routine via the PH1OUT array. Since the variables in COMMON MATOUT are not made available to the Stress and Force Calculation Packet which follows this packet, MATOUT variables desired for use in that later packet must be passed through PH1OUT, and therefore must also be listed on the stress matrix card of this Packet (second card of Subpacket A).

An example of a stress matrix card follows:

TSUBO, SB1, SB2, SB3, ST

where TSUBO, SB1, SB2, SB3, and ST are Stress Matrix Packet generated variables. One restriction applies in general to all stress matrix cards: subscripted variables may not be listed. If the second word of an array SIG is the only one being passed, a suitable variable (SIG2, for example) can be listed on the stress matrix card. SIG2 will then have to be set to SIG(2) in Subpacket C.

Subpackets B and C

Subpackets B and C are similar to those in other packets. The user may employ secondary variables to build these stress matrix card variables. He has access to the global variables and to the COMMON's MATIN, MATOUT, SDR2X5 (Table 3) and to the other reserved variables made available in Table 4.

Special Subpacket A

There are certain circumstances which require that Special Subpacket A be used (Section V discusses these circumstances). This special subpacket consists only of an options card and a card containing the string BLOCK DATA ONLY. None of the other data cards in the Stress Matrix Packet, with the exception of the BEGIN card and the optional END card, need be supplied if the Special Subpacket A is used. If the Stress Matrix Packet with Special Subpacket A is to be used, the example Stress Matrix Packet of Figure 8 will be replaced by the following four-card packet:

```
(1) BEGIN STRESS
(2) 2,35
(3) BLOCK DATA ONLY
(4) END STRESS        (optional)
```

Cards (2) and (3) compose the Special Subpacket A.

Stress Matrix Packet Example and Discussion

In the example of Figure 8, the INFLAG option value is 2. The second option value of 35 was derived as follows: First, remember that the number of grid points has already been established as 3 in Figure 3. The preprocessor sets PH1OUT(2) = NGRID(1), PH1OUT(3) = NGRID(2), and PH1OUT(4) = NGRID(3), where NGRID(I) is the Ith grid point of the element (Table 3). Secondly, add to this value of 3 the sum of the dimensions of TSUBO, SB1, SB2, SB3, and ST (these dimensions being 1X1, 3X3, 3X3, 3X3, and 3X1) for a total of 34. Add one more and the sum is 35. This final addition is due to the fact that the processor places the element ID number in PH1OUT(1).

The stress matrix card indicates that TSUBO, SB1, SB2, SB3, and ST are to be inserted into PH1OUT(5) through PH1OUT(35). Note that the secondary variable ALPHV has not been loaded into PH1OUT. All variables assume their definitions as supplied in Subpackets B and C, as predefined by the preprocessor or as supplied in the Global Variable Packet (for C1, C2, and C3, and for E1 in Figure 4). Scalar definitions for ALPHA1, ALPHA2, and ALPHV are taken from COMMON MATOUT, since the INFLAG value is 2.

After the Stress Matrix Packet has executed, the variables on the stress matrix card are made available for use in the Stress and Force Calculation routine generated. (PH1OUT is passed from COMMON SDR2X5, which is available in the Stress Matrix routine, to COMMON SDR2X7, which is available in the Stress and Force Calculation routine (Table 3)).

```
BEGIN STRESS
2,35
COMMENT  35 WORDS WILL BE PASSED INTO THE STRESS AND FORCE
COMMENT  CALCULATION PACKET--THE 35 WORDS ARE
COMMENT  ELEMENT ID(1),GRID POINTS(3),TSUB0(1),SB1(3 X 3 =9),
COMMENT  SB2(3 X 3 =9),SB3(3 X 3 =9),AND ST(3 X 1 =3)
TSUB0,SB1,SB2,SB3,ST
SB1,3,3,EQUA
SB2,3,3,EQUA
SB3,3,3,EQUA
ALPHV,3,1,TERM
ST,3,1,EQUA
DEFINITIONS FINISHED
SB1
G*C1*TR(E1)*T1
SB2
G*C2*TR(E1)*T2
SB3
G*C3*TR(E1)*T3
ALPHV
1,1,ALPHA1
2,1,ALPHA2
3,1,ALPHA2
ST
-G*ALPHV
END STRESS
```

Figure 8 - Example of a Stress Matrix Packet

STRESS AND FORCE CALCULATION PACKET

This packet generates a FORTRAN subroutine for computing element stresses and forces. The first card will contain the string BEGINSTRESS AND FORCE, with blanks wherever convenient.

Subpacket A

Card 1

The first card in Subpacket A (the next data card) is the options card which contains four fields. All four options will be integer values. The first, the number of stress words to be produced by the subroutine, must be less than or equal to 100. The second, the number of force words to be passed by the subroutine, must be less than or equal to 200. Neither of these

two options has a default value. The third and fourth parameters supplied will serve as pointers. The third points to the first word of the complex stress output string, and the fourth points to the first word of the complex force output string in the NASTRAN variable COMPLX. Both have a 0 default value. At the present time, only the 0 value may be specified for these two variables, which means that the stress and force output for the new elements must be expressed as real single-precision numbers, and not as complex representations. The values for the first two options are derived by adding 1 to the n number of stress words output by the routine or force words output by the routine. The addition of 1 is necessary to allow for the element ID which is passed automatically. If no words are to be produced, a 0 value will be inserted. The stress and force words will be passed to NASTRAN according to user specifications supplied to the Output Packet. The discussion of card 2 which follows will explain what is meant by stress and force words.

The words output from the routine generated by this packet will be printed according to specifications given by the user in the Output Packet. The relationship between the Stress Matrix, Stress and Force Calculation and Output Packets is discussed following the Output Packet section.

Card 2

The second card of Subpacket A lists all user variables (separated by commas) to be passed from the Stress and Force Calculation subroutine. They will be produced in the order in which they are listed on the card, except that the element ID will automatically be inserted before the list of stress words and again before the list of force words. However, the user must not himself punch the element ID on the card. Thus the order of the output will be: Element ID, stress words; Element ID, force words. Any secondary variable used to build these variables need not be listed on this card. No subscripted variables may be included on the list.

The variables on this card are interpreted as stress or force variables according to the first and second options described in the preceding page. If the first option is a non-zero integer N and the second option is a non-zero integer M, then the first N-1 words contained on the second card of Subpacket A are assumed to be stress words and are placed in one NASTRAN array. The last M-1 words are assumed to be force words which are placed in another NASTRAN array. The first word of each of the above NASTRAN arrays is reserved for the element ID. If either option is 0, all the variables on the card will be assumed to be either stress variables or force variables, as indicated by the non-zero option. Of course, if both options are zero, there is no need for this packet at all.

Subpackets B and C

Subpackets B and C are punched as in the other packets to define the variables listed on the second card of Subpacket A. The variables listed on the stress matrix card (second Subpacket-A card) of the Stress Matrix Packet as well as the reserved variables (Table 4) are available to the user. The COMMON's used by the routine generated from this packet are SDR2XX, SDR2X4, and SDR2X8 (Table 3).

Stress and Force Calculation Packet Example and Discussion

In the example of Figure 9, the first option value 8 implies that seven stress words are to be passed. The eighth

```
BEGIN STRESS AND FORCE
8,0,0,0
COMMENT  8 STRESS WORDS WILL BE OUTPUT  ---  THESE ARE
COMMENT  ELEMENT ID (1),SIG(3 X 1 =3),THETA(1),SIGP1(1),
COMMENT  SIGP2(1),AND TAU(1)
SIG,INSERT A,THETA,INSERT B,SIGP1,SIGP2,TAU
SIG,1,3,EQUA
SAV,1,1,DEFER
TAU,,,TERM
SIGP1,,,TERM
SIGP2,,,TERM
THETA,1,1,TERM
DEFINITIONS FINISHED
SIG
COMMENT  HERE DISPJ IS THE 3 X 1 TRANSLATION VECTOR FOR GRID POINT J
COMMENT  TEMP IS ELEMENT TEMPERATURE
SB1*DISP1+SB2*DISP2+SB3*DISP3+ST*(TEMP-TSUB0)
TAU
,,SQRT((SAV(1,1)/2.)**2+                                $
SIG(1,3)**2)
SIGP1
1,1,(SIG(1,1)+SIG(1,2))/2.+TAU(1,1)
SIGP2
1,1,(SIG(1,1)+SIG(1,2))/2.-TAU(1,1)
INSERT A
      SAV(1,1)=SIG(1,1)-SIG(2,1)
      IF(ABS(SAV(1,1)).LT.1.E-15.AND.ABS(2.*SIG(3,1)).LT.1.E-15)GO TO 90
     100
      IF(ABS(SAV(1,1)).LT.1.E-15)GO TO 9100
DEFINITIONS FINISHED
THETA
1,1,ATAN(2.*SIG(1,3)/SAV(1,1))*28.64789
INSERT B
      GO TO 9200
 9000 THETA(1,1)=0.
      GO TO 9200
 9100 THETA(1,1)=45.
 9200 CONTINUE
DEFINITIONS FINISHED
END STRESS AND FORCE
```

Figure 9 - Examples of a Stress and Force Calculation Packet

word (the first to be passed) is the element ID. The 0 value for the second option indicates that no force words are to be passed, and all of the variables listed on the next card are stress variables. There is a 0 value for both the third and fourth options, as is presently required. The next non-COMMENT data card indicates the seven stress words to be output which are SIG(1), SIG(2), SIG(3), THETA, SIGP1, SIGP2, and TAU. These, together with the element ID, account for the eight words indicated. Notice the INSERT variable names after SIG and THETA which indicate that user-supplied FORTRAN coding is to be added after SIG and THETA are generated. These INSERT variable names are not to be included in the count as force or stress variables.

Subpackets B and C in this example are specified in the usual manner. Note that the variables TSUBO, SB1, SB2, SB3, and ST are available for use even though they are not redefined here, since they have been passed from the Stress Matrix Packet example discussed in the previous packet description (Figure 8). Note also the use of the reserved variables DISP1, DISP2, and DISP3 (Table 4), the displacement vectors for the first, second, and third grid points of the element, respectively. Each of the vectors DISP1, DISP2, and DISP3 may have the following interpretation: If the degrees of freedom (supplied in the Preliminary Data Packet) are some combination of 1, 2, and 3, the numbers supplied are three translations. If some combination of 4, 5, and 6, the numbers supplied are three notations. If some combination of both of these sets of integers, the numbers supplied imply all six displacements, three translations plus three rotations. The variable TEMP is a COMMON variable in COMMON SDR2X4 (Table 3). A third point to be noted is that the arguments of the ATAN function are tested using user-supplied, not preprocessor generated coding.

Special Subpacket A

Just as in the Stress Matrix Packet, use of the Special Subpacket A may sometimes be indicated. If so, the example in Figure 9 would become:

(1) BEGIN STRESS AND FORCE
(2) 8, 0, 0, 0
(3) BLOCK DATA ONLY
(4) END STRESS AND FORCE (Optional)

For discussion of the precise circumstances which prompt the use of this special subpacket, see Section V.

OUTPUT PACKET

The Output Packet generates a BLOCK DATA FORTRAN subprogram which produces updates to various NASTRAN tables containing NASTRAN FORTRAN format statements. Its use complements the Stress Matrix Packet and the Stress and Force Calculation Packet in that it contains the headings and formats needed to label and print the results of the stresses and forces calculated by NASTRAN. Therefore, if stresses and/or forces are to be calculated and printed, all three of these packets—the Stress Matrix, the Stress and Force Calculation, and the Output Packet—must be submitted. If stresses and forces are not to be calculated for a new element, these three packets are unnecessary.

The NASTRAN Output File Processor functional module—and therefore the Preprocessor Output Packet—is rather complicated. The logical cards needed by this packet are listed below, with numbers assigned to facilitate discussion of the various cards in the pages that follow.

(1) BEGIN OUTPUT
(2) STRESS or FORCE card
(3) Options card
(4) Card containing six integers, five of which are heading formats

(5) Sequence of integers ending with a 0, all but the last are format pieces

(6) A format number Card (4)

(7) FORTRAN format for a new heading } Repeated for each new heading format

(8) Integer from Card (5) }
(9) New format piece } Repeated for each integer on Card (5) that represents a new format piece

Optional Logical Cards:

(10) STRESS or FORCE card (whichever was not submitted in Card (2))

.
.
.

(17) Repeat of type of information contained in Cards (3) through (9) above if Card (10) is punched

(18) DEFINITIONS FINISHED

Discussion of the Cards

Card 2

The information in Card (2), which follows the BEGIN OUTPUT card, will be either the word STRESS or the word FORCE. This information will indicate which type of specification output is to be printed by NASTRAN; stresses or forces. There is no default value for (2).

Card 3

The options card (3) contains three parameters: the first, may be either the keyword SORT1 or SORT2; the second, the keyword REAL or COMPLEX; the third (to be used only if COMPLEX has been specified as the second parameter), the keyword MAGNITUDE/PHASE or REAL/IMAG. The user indicates use of the defaults by coding two consecutive commas on the card or by inserting a blank card. For example, a card with

SORT1, COMPLEX,

will have a default value for the third parameter of REAL/IMAG. However, since the preprocessor does not allow complex stress or force output at the present time, default values for the second and third parameters on this card should be used.

The difference between SORT1 and SORT2 output is noted here. SORT2 output is available only for Transient-Response and Frequency-Response NASTRAN problems. In the SORT2 mode, the output is arranged so that results are printed according to point ID; the results are printed at every selected time step (TSTEP bulk data card) for a point ID. In the SORT1 mode, the reverse is true; for each time step, the results are printed for every point ID. As an example of SORT1 and SORT2 use, refer to the NASTRAN Programmer's Manual[3], p.4 62-1. The permanent elements already existing in NASTRAN are presently set up for both SORT1 and SORT2 Output; formats exist for printing results from all rigid formats. However, the preprocessor prevents its new elements from containing SORT1 and SORT2 formats simultaneously. This means that if the SORT1 formats are to be introduced into NASTRAN for the new elements via the preprocessor, some change will have to be made before running a Transient-Analysis or a Frequency-Response problem which selects SORT2 formats. Actually, under the present rigid NASTRAN formats, only SORT2 output is available for the Transient-Response problem.

Two relatively simple solutions to this dilemma are possible. One solution is to perform a preprocessor run with one set of formats, e.g., SORT1, and to then submit a subsequent preprocessor run to provide optional SORT2 formats. The data for this second preprocessor run would consist only of the Preliminary Data Packet and the Output Packet, the input being made up of the set of SORT2 formats. If SORT2 formats later prove desirable, the formats could then be linked into NASTRAN.

The second solution is to provide SORT1 formats only. If these formats have been linked into NASTRAN and a Transient Response analysis is desired, the user places the following DMAP alter statements in the NASTRAN Level 12.0 or Level 15.0 Executive Control deck:

```
ALTER 128, 130
CHKPNT OPP1, OQP1, OUPV1, OES1, OEF1$
OFP OPP1, OQP1, OUPV1, OEF1, OES1, //V, N
CARDNO$
ALTER 136
SDR3, OPP1, OQP1, OUPV1, OES1, OEF1, /OPP2,
   OQP2, OUPV2, OES2, OEF2, $
CHKPNT OPP2, OQP2, OUPV2, OES2, OEF2 $
ENDALTER
```

These DMAP changes will cause NASTRAN to print results (stresses and forces) in SORT1 format, but will make the changes required to perform any X-Y plotting.

Card 4

This card will contain six integers, each separated from the other by a comma. The first is a pointer to the NASTRAN FORTRAN array OFP1BD, its value ranging from 1141 to 1400, with a default of 1141. It determines where the five headings will be stored in NASTRAN's storage scheme. There is one restriction to this integer which is noted a little later in the discussion of (5). The next five integers affect NASTRAN print procedures. A value of 0, -1, or an integer in the range 217 and 336 may be used. Five values must be supplied. If fewer than five format headings are to be supplied, the user must fill the remaining integer spaces with -1's. The -1 value is merely a filler value, used to comply with the five-integer requirement and does not affect the NASTRAN output. A 0 value produces a blank line of output during a NASTRAN run. The integers 217 through 336 indicate output-heading FORTRAN format statement

numbers. The order in which the numbers are listed on the data card is the order in which NASTRAN will print the lines defined for those statement numbers. Consider, for example, a card containing the string 1141, 218, 0, -1, 240, 220. The card will be interpreted in the following way. The pointer has a value of 1141. The user will specify in subsequent cards of the Output Packet complete FORTRAN formats for FORMAT statements 218, 240, and 220. NASTRAN will print the contents of FORMAT statement 218, follow with a blank line (as a consequence of the 0), and then print the contents of FORMAT statements 240 and 220, in just that order. The -1 value will not have any affect, and merely fulfills the requirement that five integers be supplied on the card.

Card 5

Card (5) contains a sequence of integers which direct the selection of NASTRAN format pieces. Each "piece" defines a segment of a line of computer output. Table 8 lists the format pieces defined by NASTRAN which are available to the user. These format pieces, when strung together, produce the computer-listing FORTRAN format for printing stresses or forces. The first integer is a four-digit packed number (for example, 0101, which is the same as 101) in which the two right-hand digits control the number of output lines to be produced by the data record (a data record being defined as the force or stress data for one element), and the two left-hand digits control the number of data records contained per line of output. Therefore, at the user's option, NASTRAN will print stresses and forces for more than one element on the same line of NASTRAN output. If the two left-hand digits are both 0's, each line of the NASTRAN output will contain at most one data record. As an example of how this first integer will be interpreted, assume that the integer is either 101 or 1. The data for a single element will then be printed one data record per line. If data for the elements are to be processed two data records per line, the

number would be 201. If one data record were to produce two lines of output, the integer would be 2 or 102. There is no default value for this integer.

The sequence of integers which follows this packed four-digit integer may be continued onto subsequent cards by using the preprocessor's dollar-sign convention. The last integer of the sequence must be a 0, which acts as a terminator. When added to the first integer of (5), the number of coded integers (including the 0 value), must be less than 1401. This sum is the restriction mentioned earlier in the description of Card (4). Also, the number of integers specified must be less than or equal to 45.

The interpretation of these integers is as follows. None but the last integer of the sequence may have a 0 value. All the others must lie within one of the following two closed intervals: [-40, -1] or [1, 100]. NASTRAN's COMMON block OFP5BD contains two arrays, ESINGL and E. If I, the integer, has a value less than 0, NASTRAN will reference ESINGL(-I); a positive I value will cause NASTRAN to reference E(5*I-4) through E(5*I).

Since a format piece for I in the closed interval [-40, -32] is not defined in Table 8, a new format for the entry in ESINGL must be defined for that value of I. Most existing ESINGL formats are either FORTRAN X or H formats. In any case, an ESINGL format piece should contain no more than four characters. I in the interval [29, 39] implies that E(5*I-4), E(5*I-3), E(5*-2), E(5*I-1), and E(5*I) must be defined. These five words can be divided into two groups: the first to be made up of E(5*I-4) and E(5*I-3); and the second to be made up of E(5*I-2), E(5*I-1), and E(5*I). The first group is termed the standard format; the second group is called the alternate format. The standard format comprises a single E or F format piece which NASTRAN uses to print non-zero terms. NASTRAN uses the alternate format if the value to be printed is exactly zero.

If I is the interval [74, 100], then E(5*I-4) and E(5*I-3) may contain the X, I, or A FORTRAN format specifications.

Cards 6 through 9

The user next defines on Cards (6) through (9) all the full formats and format pieces left undefined by the preprocessor. Each of the five integers following the pointer on (4) must have a full FORTRAN format associated with it. The user must supply a format for those integers greater than 217 and less than 336. An integer value of 0 produces a blank line. A value of -1 defaults to no format. Each format supplied must be less than 134 characters, or one line of computer output. The five integers mean that five output lines at most, are available for each new NASTRAN output heading. The output headings for the 12 new preprocessor elements—as many as ten per element format, five for stress and five for forces—equals 120 new formats in all. These 120 new formats may not exceed 2000 characters.

The preprocessor ensures that any of the five integers which is greater than zero will find a format match. It does so in the following manner. First, the user codes a format number on (6). It must match one of the five integers on (4) or an error will appear in the preprocessor output, and all further FORTRAN code generation for the Output Packet will halt. The actual format is punched on (7) and may spill over onto additional cards as needed by using the dollar-sign convention. A left parenthesis must be the first character of the format, and a right parenthesis must be the last. If either parenthesis is missing, the preprocessor generates an error; and subsequent coding stops. This algorithm retains control until all five integers are defined. Each format should produce one line of NASTRAN output.

The user now defines the format pieces indicated by the sequence of integers (5). If the integer I is in one of the closed intervals [-40, -32], [29, 39], or [74, 100], then specification for the format piece is necessary. Otherwise, the format piece will be assumed in accordance with the listing in Table 8.

The I value is punched on (8). This integer must match one of the integers of the sequence of integers on (5). The preprocessor generates an error message if no match occurs. For I in the interval [-40, -32], the user codes on the subsequent card (9) a four-character format. For I in the interval [74, 100], he punches a string of characters, eight at most. The contents of (9) are the format. The last character specified must be a part of the format and not a trailing comma, since NASTRAN automatically places a comma after the eight-character string specified.

If I is the interval [29, 39], the form of the format piece will be slightly different. The user first specifies one E or F FORTRAN format of at most eight characters as the standard format. He then codes a comma and follows it with a character string of up to 12 characters which will be the alternate format. The last character of the string may not be a comma, since NASTRAN automatically inserts a comma after two twelveth character.

The user must code the paired cards (8) and (9) until each previously undefined integer I on (5) has been assigned an associated format piece. Moreover, the order in which the format pieces are defined must correspond to that in which the I values were previously specified on (5).

Cards 10 through 18

If the second card of the Output Data Packet contains the word STRESS, the user may now code the word FORCE on the Card (10) and follow it with a succession of cards such as

those just described for the headings and formats of the stress output. Thus, the Output Packet may consist of two subpackets, each of an identical form: one subpacket, with the word STRESS on the first card, to update NASTRAN formats with respect to the printout of stress values. The other, with the word FORCE on the first card, updates NASTRAN formats with respect to the printout of force values.

Output-Packet Example and Discussion

```
BEGIN OUTPUT
STRESS

COMMENT  WE ARE DEFINING FORMATS FOR SORT1,REAL OUTPUT
COMMENT  NEW HEADING FORMATS ARE 240,220,AND 225
1150,240,0,220,-1,225
COMMENT  THE FOLLOWING ARE THE I VALUES FOR THE FORMAT PIECES
1,74,-4,30,-33,39,-4,30,-33,30,-4,39,-33,39,   $
-33,30,0
COMMENT  WE WILL NOW SPECIFY THE NEW FORMAT HEADINGS
240
(40X,51HE L E M E N T  S T R E S S E S  F O R  E L E M 4 0 )
220
(1X,7HELEMENT)
225
(3X,3HID.,9X,6HSIG(1),10X,6HSIG(2),10X,6HSIG(3),11X,5HTHETA,$
11X,5HSIGP1,11X,5HSIGP2,12X,3HTAU)
COMMENT  WE WILL NOW SPECIFY THE NEW FORMAT PIECES
74
1X,I7
30
1PE11.4,0PF8.1,3X
-33
5X
39
1PE11.4,0PF8.1,3X
DEFINITIONS FINISHED
COMMENT  ACCORDING TO THE STRESS AND FORCE CALCULATION PACKET
COMMENT  8 VALUES WILL BE OUTPUT, THE FIRST BEING THE ELEMNT ID ---
COMMENT  COMBINING THE SPECIFIED FORMAT PIECES,ADDING IN THE APPROPRIATE
COMMENT  COMMAS,AND THE BEGINNING AND ENDING PARENTHESES(WHICH NASTRAN
COMMENT  DOES) WE GET
COMMENT  (1X,I7,5X,1PE11.4,5X,1PE11.4,5X,1PE11.4,5X,1PE11.4,5X,1PE11.4,
COMMENT  5X,1PE11.4,5X,1PE11.4)
END OUTPUT
```

Figure 10 - Example of an Output Packet

Since the word STRESS is coded on the second card of the example packet, the following data is interpreted with respect to stress values. The third data card, the options card, is blank, so SORT1 and REAL are the parameters selected by the preprocessor. The fourth data card indicates that the pointer into the NASTRAN COMMON block OFPIBD is to be 1150; the new format headings 240, 220, and 225 are to be defined. During NASTRAN execution, Format 240 will be printed first, followed by a blank line. The formats 220 and 225 will be printed next. The -1 value is included on the card only to fulfill the five-integer requirement and does not indicate any action to be taken.

The sequence of integers ending with a value of 0 is supplied on (5). Note the use of the dollar-sign convention to indicate continuation onto a subsequent card. Through interpretation of the first integer—the packed value—the preprocessor knows that one line of output is to be produced per data record.

The integer format heading numbers other than 0 or -1 are defined on (6) through (12). Formats 240, 220, and 225 are therefore coded in paired cards. Note that in each case, the first character is a left parenthesis and the last character is a right parenthesis.

All undefined integers in the sequence on (5) are now defined. As Table 8 indicates, the -4 value produces a format piece of 5X. The integers 74, 30, -33, and 39 produce the following four pieces respectively: 1X, I7; 1PE11.4, 0PF8.1, 3X; 5X; and 1PE11.4, 0PF8.1, 3X. Stress output is printed following Format 225 in the following format:

```
1X, I7, 5X, 1PE11.4, 5X, 1PE11.4, 5X, 1PE11.4,
5X, 1PE11.4, 5X, 1PE11.4, 5X, 1PE11.4, 5X, 1PE11.4
```

Therefore, seven element stresses will be printed on each NASTRAN output line. Note that the format pieces are specified in an order that corresponds to that in which the I values were specified on (5). Since there is no card containing the word FORCE supplied in the packet following the format piece definitions, no output subpacket for force values is included. Coding of the packet is ended, and the card containing the string DEFINITIONS FINISHED verifies that fact.

TABLE 8 - NASTRAN FORMAT PIECES AVAILABLE

Value of I	Format	
	Standard	Alternate
-1	/	None
-2	15X	
-3	10X	
-4	5X	
-5	1X	
-6	/10X	
-7	16X	
-8	2H1	
-9	2H2	
-10	2H3	
-11	2H4	
-12	2H5	
-13	7X	
-14	/16X	
-15	/13X	
-16	4X	
-17	/14X	
-18	11X	
-19	/24X	
-20	1HO	
-21	2H/	
-22	2HEN	
-23	2HDA	
-24	2HDB	
-25	/1HO	

TABLE 8 - NASTRAN FORMAT PIECES AVAILABLE—Continued

Value of I	Format	
	Standard	Alternate
-26	23X	None
-27	/26X	
-28	/9X	
-29	/12X	
-30	/1H	
-31	/20X	
1	1PE15.6	OPF6.1, 9X
2	1PE16.6	OPF7.1, 9X
3	1PE17.6	OPF8.1, 9X
4	1PE18.6	OPF9.1, 9X
5	1PE19.6	OPF10.1, 9X
6	1PE20.6	OPF11.1, 9X
7	1PE21.6	OPF12.1, 9X
8	1PE30.6	OPF21.1, 9X
9	1PE26.0	OPF17.1, 9X
10	1PE24.6	OPF15.1, 9X
11	OPF11.4	OPF8.1, 3X
12	OPF14.4	OPF11.1, 3X
13	1PE28.6	OPF19.1, 9X
14	1PE37.6	OPF28.1, 9X
15	1PE22.6	OPF13.1, 9X
16	1PE14.6	OPF5.1, 9X
17	OPF15.4	OPF21.1, 3X
18	OPF9.4	OPF6.1, 3X
19	OPF15.4	OPF12.1, 3X

TABLE 8 - NASTRAN FORMAT PIECES AVAILABLE--Continued

Value of I	Format	
	Standard	Alternate
20	1PE23.6	OPF14.1, 9X
21	1PE35.6	OPF26.1, 9X
22	1PE25.6	OPF16.1, 9X
23	1PE50.6	OPF41.1, 9X
24	OPF46.4	OPF43.1, 3X
25	OPF15.4	OPF12.1, 3X
26	OPF20.4	OPF17.1, 3X
27	OPF16.4	OPF13.1, 3X
28	OPF22.4	OPF19.1, 3X
40	1PE9.1	A1, 8X
41	6X,A1,3X	I7, 3X
42	I15	None
43	I9, 1X	
44	1HO, I8	
45	1X, I13	
46	1X, I8	
47	1HO, I7	
48	6X, I8	
49	1X, I15	
50	1X, I12	
51	I10	
52	I7, 1X	
53	3X, A4	
54	1HO, I13	
55	1X, I20	
56	5X,A1,3X	

TABLE 8 - NASTRAN FORMAT PIECES AVAILABLE—Concluded

Value of I	Format	
	Standard	Alternate
57	1X,I22	None
58	I12	
59	1X, I19	
60	Blank	
61	I8	A4, 4X
62	I9	A4, 5X
63	I11	A4, 7X
64	I20	A4, 16X
65	I19	A4, 15X
66	1X, I23	None
67	I23	
68	I28	
69	/1H, I18	
70	1HO, I15	
71	1HO, I14	
72	OPF22.4	I9, 13X
73	OPF16.4	I5, 11X

Discussion of the Interrelationships Among Stress Matrix, and Force Calculation, and Output Packets

These next remarks apply in general to the examples provided in Figures 8,9, and 10. Figure 10 is the packet that will ultimately print the results of input packets of Figures 8 and 9. They show how the Stress Matrix, the Stress and Force Calculation, and the Output Packets are interrelated.

Eight is the number of stress words to be produced, as indicated on the options card of the Stress and Force Calculation Packet. The sequence of integers on (5) in the Output Packet must specify a format which will correctly present these stress words to the user via NASTRAN output. For example, in Figure 9 a total of eight stress words are indicated to be produced, the first being the element ID and the other seven being the stress values. As we have already noted, these values will be printed according to the format specified in (5) as follows:

1X, I8, 5X, 1PE11.4, 5X, 1PE11.4, 5X, 1PE11.4, 5X,
1PE11.4, 5X, 1PE11.4, 5X, 1PE11.4, 5X, 1PE11.4

We can see the element ID will be printed in the I7 format, and that the other seven stress words will be printed in the 1PE11.4 format. Note that in this example, the value indicating the number of force words to be produced is 0; consequently a FORCE card and others to specify FORCE formats are not included.

Using the generated subroutines and BLOCK DATA subprograms generated by Figures 8, 9, and 10, if SORT1 is to be specified in the NASTRAN case control deck and if Transient Response or Frequency Response analyses are not to be run, stress output will consist only of the element ID followed by the specified seven stress words. If a Transient Response analysis is to be run, the output will consist of the time, the element ID, and the seven stress words indicated. In this case, time is an output word. However, since the time is automatically noted by NASTRAN, the user may specify his input (in both the Stress and Force Calculation Packet and the Output Packet) as though time were not an output word. NASTRAN sets up the format for printing the time.

If SORT2 is to be specified in the NASTRAN case control deck, the output will be the element ID, the time, and the stress words indicated. In this case again the user specified his input in the Stress and Force Calculation Packet as though time were not an output word. However, provision must be made in the Output Packet for noting the time as has been done in Figure 10, for example, in which the fourth input card specifies six integers. The second integer (240) is the first output heading format. If SORT2 is to be specified, then the first output heading format integer should be 108, which will cause the element ID to be labeled and printed. The heading TIME should be specified next, instead of specifying a heading for the element ID, as in Figure 10. Finally, the actual format piece corresponding to the time should be an E or F format, rather than the integer format (I7) for the element ID supplied in Figure 10. The DEFINITIONS FINISHED card indicates the end of the Output Packet.

The complexity of the Output Packet reflects the complexity of NASTRAN's Output File Processor. If the user prefers, he may, through a user-supplied insertion subpacket in the Stress and Force Calculation Packet, generate a printout of his results directly via FORTRAN statements in lieu of using the Output Packet. Although unorthodox, this approach saves the user's time. In any case, the number of output words must always be included on the options card of the Stress and Force Calculation Packet.

DIFFERENTIAL STIFFNESS PACKET

The setup of the Differential Packet is identical to that of the Stiffness Matrix Packet. The user should change all references to K and KIJ to D and DIJ, respectively. The variables K11, K12, ..., become D11, D12, ... The COMMON's used by the routine generated by this packet are MATIN, MATOUT, DS1ET, DS1AAA, and DS1ADP, which are listed in Table 3. Table 4 lists the other reserved variables available.

IV. OPERATING INFORMATION

COMPUTER SYSTEM CONTROL CARDS

For the CDC-6000 series computers, cards similar to the following are needed to control the execution of the preprocessor:

```
ATTACH(PREP, PREPROCESSOR)
RFL, 250000.
NOREDUCE.
SNOBOL(PREP,,PUN,INPUT)
7/8/9
   input data 6/7/8/9
```

where

PREP	is the logical file name for the preprocessor source coding
PREPROCESSOR	is the permanent file name under which the preprocessor is catalogued
PUN	is the file containing preprocessor-generated subroutines and BLOCK Data subprograms
SNOBOL	invokes the system SNOBOL interpreter
INPUT	is the file containing the user input data

On the SNOBOL card, the preprocessor name PREP indicates the SNOBOL program to be compiled by the SNOBOL compiler, and INPUT is the name of the standard input file containing the input data. The preprocessor is written in the SNOBOL computer language, Version 3. At present, the preprocessor requires an approximate minimum of 250000_8 words of central memory, although less core can be provided at the user's discretion at a cost of longer running times. Output will be printed and/or punched as the user options specify.

NASTRAN CARD FORMAT FOR PREPROCESSOR-GENERATED STRUCTURAL ELEMENTS

(The reader is assumed to be familiar with NASTRAN Bulk Data cards in general and with element connection and property cards in particular.)

The connection card, and, if necessary, the property card, for a new element will have to be designed by the analyst before the preprocessor may be used. The new bulk data cards may be used in a NASTRAN run as soon as the preprocessor-generated routines and tables have been inserted within NASTRAN.

Names designated for the new connection and property cards must correspond to the name of the element to be generated. Thus, for ELEM39, the connection card name would have to be CELEM39, and the property card name would have to be PELEM39. For the other elements ELEMj, the connection and property card names would be CELEMj and PELEMj, respectively, with j = 40, 41,..., 50. These names are referred to as connection card and property card mnemonics.

When both a connection card and a property card will be needed, the following format must be used:

1	2	3	4	5	6	7	8	9	10
CELEM39	EID	PID	G1	G2	---	GN	CV1	etc.	etc.

PELEM39	PID	MID	PV1	PV2	---	---	etc.		

If only a connection card is needed, a different format must be followed:

CELEM39	EID	G1	G2	---	GN	CV1	etc.	MID	

In these format descriptions, the integers 1 through 10 refer to the different data fields of the bulk data card. The symbols have the following meanings:

EID	Element identification number
PID	Identification number of a PELEM39 property card
G1,G2,---GN	Grid point numbers to which this element connects. N is the number of grid points for this element.
CV1,...,etc.	Connection card variables, as specified in the Preliminary Data Packet.
MID	Identification number of a material property card.
PV1,...,etc.	Property card variables as specified in the Preliminary Data Packet.

Note the following:

- As with element identification numbers for present NASTRAN elements; EID is arbitrary, but must be unique with respect to all other elements in the problem.

- The PID on the CELEM39 connection card refers to the PID on the corresponding property card.

- The MID on the connection or property card refers to a NASTRAN material bulk data card identification number.

- A logical connection or property card may be made up of a number of 10-field cards. However, at the present time this number must be fixed; no open-ended cards are allowed. This number is determined by data supplied to the Preliminary Data Packet. Also, the total number of data items on the connection and property cards combined must not be greater than 97-5N, with N being the number of grid points for the new dummy element.

- If no property card exists, the material identification number must be the last variable on the logical connection card.

- The connection card variables CV1,..., etc., and the property card variables must, for now, be real, single-precision numbers.

- The preprocessor requires that a variable corresponding to an anisotropic material angle be given on the connection card, even though the value of this variable may be zero for every use of the element. The preprocessor will assume that this angle (in degrees) will be the last connection card variable if a property card exists, or the next-to-the-last connection card variable (immediately preceding MID) if no property card exists. This variable must be given and must be included as one of the connection card data items given in the Preliminary Data Packet as specified on page 58.

LINKAGE

NASTRAN consists of 15 separate programs or links, one of which may be termed the super-link. One of the duties of this superlink is to supervise the movements of the other 14 links into and out of central memory. At least four links need to be updated when a new element is to be added. If the analyst makes full use of the preprocessor, six links will require updating. Those individuals wanting to use CDC equipment will want to obtain the Naval Ship Research and Development Center report[4] which describes a linkage-editor that is an extension of the standard NASTRAN CDC linkage editor.

Before a new element may be used, the preprocessor-generated routines and tables must be inserted into NASTRAN. At present, the updating (re-link-editing) of the appropriate NASTRAN links must be performed by the user in a separate computer run after the new routines and tables needed have been generated by the preprocessor. At some future time, this step may be incorporated as an automatic function of the preprocessor. After this link-editing run is complete, the analyst may use his new element with NASTRAN. In all, three computer runs must

[4] Martin, Roger, "A General Purpose Overlay Loader for CDC-6000 Series Computers; Modification of the NASTRAN Linkage Editor," Naval Ship Research & Development Center Report 4062 (Apr 73).

be made before the results with a new element may be obtained: a preprocessor run, a link-editing run, and a NASTRAN run.

The following paragraphs discuss the re-link-editing of the NASTRAN links on the CDC 6000-series computers, and explain just what must be link-edited. Six files are required in the re-link-editing process:

1. A file of the object decks of the preprocessor-generated routines and tables.
2. A file containing the NSRDC-modified CDC linkage editor.
3. A file containing all the object decks in the links being updated.
4. A file of the overlay structure of the links being updated.
5. A file containing NASTRAN in executable form.
6. A file named LINKLIB which contains the CDC system library routines.

The Level 12 NASTRAN file (5) indicated is not the standard release version, but one that contains updates to handle the new elements. Those wishing to use the preprocessor under Level 15's dummy element facilities may use the standard release issued by NASA. Files (1) through (5) for Level 12 may be obtained from the Navy NASTRAN Systems Office (NNSO, Code 1844) of the Computation and Mathematics Department, NSRDC.

A Sample Link-Edit Run Deck:

The cards in the following example have been numbered to facilitate discussion.

```
(1)  ATTACH (NEW, SOURCEDECKS)
(2)  RUN (S,,,NEW,,NEWOBJ,,,1)
(3)  REWIND(NEWOBJ)
(4)  ATTACH(LINKEDT, LINKAGE)
(5)  ATTACH(LINKLIB, LIBRARY)
(6)  ATTACH(NASTRAN, UPDATED)
(7)  ATTACH(NASTOBJ, DECKS)
```

```
(8)  NOREDUCE.
(9)  LINKEDT.
(10) EXTEND,NASTRAN.
(11) 7/8/9
(12) LINKEDIT INFILE=NASTRAN(C), OUTFILE=NASTRAN(C)
(13) LIBRARY NEWOBJ, NASTOBJ
         overlay cards for all links to be updated
         6/7/8/9
```

Card (1) attaches to the job the file containing the output of the preprocessor—the source decks of the new routines and tables. Card (1) may be omitted if the new source decks are in card form, in which case the string NEW in Card (2) would be changed to INPUT. Card (2) invokes the FORTRAN RUN compiler. (NASTRAN is written in RUN FORTRAN. However, the RUN compiler needed is a special one. The necessary updates to the standard RUN compiler are received from COSMIC from which the CDC version of NASTRAN is ordered.) The user may also obtain NASTRAN Level 15 compiled under the FORTRAN EXTENDED compiler from NNSO of the Computation and Mathematics Department. The extended linkage-editor mentioned as Reference 4 may be used.

The input to the compiler consists of the new source decks. The new object decks are written on file NEWOBJ. Card (3) rewinds this object file. The file containing the CDC linkage editor is attached by Card (4). Cards (5) and (6) attach the LINKLIB library and NASTRAN, respectively. Card (7) attaches all the NASTRAN object decks. Although only the file containing the object decks for the links being re-link-edited is required, there is no penalty for attaching all the object decks. Card (8) specifies that the central memory size is not to be reduced after the linkage editor is loaded. Card (9) loads and executes the linkage editor. Card (10) signifies that the changes to NASTRAN are to be permanent. Card (11) indicates the end of the control card record. Cards (12), (13), etc., are the input to the linkage editor. Other options are available on the LINKEDIT card. These options are discussed in the NASTRAN Programmer's

Manual, Section 5.5. Card (9) could be replaced by

LINKEDT(OVERL)

in which OVERL is a file of the source cards (12), (13), etc.

The original NASTRAN file, attached with Card (6), and the overlay cards for the links that pertain to the preprocessor must be obtained from NNSO (Level 12).

When this link-edit run has been successfully completed, the new element will have been fully implemented into NASTRAN. The new structural element may now be used in a NASTRAN analysis run.

NASTRAN OVERLAY

This section will discuss which new routines and tables fit into which link, when a link must be updated, and how to update NASTRAN when previous updates have already been made.

The NASTRAN links 1, 2, 3, 5, 13, and 14 may be updated. Table 9 lists the various routines and tables and the links in which each belongs. The functions of the various routines have already been discussed in Section III.

TABLE 9 - LINKS CONTAINING THE GENERATED ROUTINES AND TABLES

Name of Routine or NASTRAN COMMON Block	Link to Receive Routine or NASTRAN COMMON Block Replacement
IFPCOM, IFSCOM	1
GPTCOM	2
EDSCOM	2
SM1COM	3
SM2COM	3
KLEMi	3
MLEMi	3
VLEMi	3
EDTCOM	5
Mi	5
SDRCOM	13
SELi1	13
SELi2	13
DLEMi	13
DSA1	13
OFPCOM, HEDCOM	4,5,6,12,14

Since a subroutine is generated only if the appropriate data packet is used, no confusion exists as to whether or not a subroutine need by linked into NASTRAN. With the exception of OFPCOM, HEDCOM, and DSA1, all tables are updated each time the preprocessor is executed. Table 10 indicates the circumstances in which the NASTRAN COMMON Blocks must be link-edited into NASTRAN.

TABLE 10 - LINK-EDITING THE NASTRAN COMMON BLOCKS

Block Name	When Re-Link-Editing is Required
IFPCOM, IFSCOM	Each time the preprocessor is run
GPTCOM	Each time the preprocessor is run
EDSCOM	If a new element is to be plotted
SM1COM	After each preprocessor run, since a new stiffness matrix routine is required.
SM2COM	If a mass matrix and/or a viscous-damping matrix routine for the new element to be link-edited.
EDTCOM	If a thermal-loading-vector routine for the new element is to be link-edited, or if the new element is to be used with other elements which will have temperature loading.
SDRCOM	After each preprocessor run, since output of displacements, forces, and stresses make use of this table.
OFPCOM, HEDCOM	If the Output Packet has been used as input to the preprocessor (which is only true if stress and/or force output is desired.
DSA1	If the Differential Stiffness Packet has been used as input to the preprocessor.

From Tables 9 and 10 it may seem that Links 1, 2, 3, and 13 must be updated for every new element. When a new element is to be added to NASTRAN via the preprocessor and a previously added element is to remain, some manipulation of the table updates becomes necessary.

Almost every preprocessor-generated FORTRAN variable in every generated table is initialized to 0, unless the user specifies otherwise—the only exceptions being the two variables IFRMTS and IFMT in table HEDCOM which are initialized to four blanks per computer word (4Hbbbb) by the preprocessor execution. This initialization includes those variables which pertain to the 11 elements not defined in a preprocessor run.

For example, suppose that a version of NASTRAN contains 38 permanent elements, that Element 39 was added via the preprocessor in an earlier run, and that Element 40 is to be added without affecting Element 39. The tables for Element 39 and Element 40 must be hand-merged if these two elements are to reside simultaneously in NASTRAN. When the routines for Element 40 are link-edited into NASTRAN, the tables to be link-edited should be the union of the tables for Element 39 and those for Element 40. The union of a zero and a non-zero number is defined to be the non-zero number. This union must be performed for every variable in every DATA statement in every table and must include all new elements that the user wishes to remain operable. For example, if Element 39, 40, and 43 are all to be operable in the program, the following original DATA statements for the variable NWDEST for the three elements

```
DATA NWDEST/19,11*0/
DATA NWDEST/0,22,10*0/
DATA NWDEST/4*0,21,7*0/
```

must be merged. The variable NWDEST is a dimensioned variable with dimension 12. If Element 39 had already been added to NASTRAN, the addition of ELEMENT 40 would require the DATA statement

```
DATA NWDEST/19,22,10*0/
```

When Element 43 is to be link-edited, this DATA statement must become

```
DATA NWDEST/19,22,0,0,21,7*0/
```

This merging of tables for every variable in every DATA statement for every table is not the formidable task it might seem, since the number of variables is not large, and the same DATA statement appears in several different tables.

The two exceptions in table HEDCOM mentioned earlier are the variables IFRMTS and IFMT. Each element of both arrays is initialized to four BCD blanks, i.e., 4Hbbbb, with b representing the blank. The user need never alter IFMT, since it is not set to other than its initial values by these tables and will thus always appear to the user to be initialized. However, the variable IFRMTS will change. This variable contains the output headings as specified in the Output Packet. Since there are a maximum of ten output headings per element (five for stresses and five for forces), and since each output heading is limited to 133 characters by computer hardware characteristics, each element is allotted 333 words of IFRMTS (four characters per word). Element 39 is allotted words 1-333; Element 40, words 334-666; and so on. Therefore, after the preprocessor run is made for Element 40 in this example, words 334-666 of variable IFRMTS will, in general, be filled; words 1-333 and words 667-3996 will be initialized to 4Hbbbb.

If the user wishes to add Element 40 to NASTRAN without disturbing the previously added Element 39, he will have to merge the DATA statements for variable IFRMTS (assuming of course that Output Packets were specified for both elements). This merger is affected by placing the DATA statements for IFRMTS which were generated for Element 40, into the BLOCK DATA subprogram for Element 39.

The foregoing discussion should enable a user to add a new element into NASTRAN via the preprocessor without disturbing any of the elements already added.

V. OUTPUT CONSIDERATIONS

The output received from a preprocessor run will be a series of FORTRAN subroutines and BLOCK DATA subprograms. The output will be printed, punched, or written on an external storage device, as the user options specify. The subroutines and the output they produce are listed in the following table.

TABLE 11 - PREPROCESSOR-GENERATED SUBROUTINES

Subroutine	Quantity Computed
KLEMi	Element stiffness matrix
MLEMi	Element mass matrix
VLEMi	Element viscous damping matrix
Mi	Element thermal loading vector
SELi1	Element stress matrix
SELi2	Element stresses and forces
DLEMi	Element differential stiffness matrix
Note: i = 39, 40,..., 50. KLEM39 is the stiffness matrix routine for ELEM39.	

As many as ten BLOCK DATA subprograms may be produced. These subprograms contain the table updates which NASTRAN will use to correctly compute results for the new elements. The subprograms and the functional modules which contain them are listed in the table that follows.

TABLE 12 - PREPROCESSOR-GENERATED BLOCK DATA SUBPROGRAMS

Named COMMON	Functional Module Which Employs the Named COMMON
IGPCOM, IFSCOM	Input File Processor
GPTCOM	Geometry Processor
EDSCOM	Plot Set Definition Processor
SM1COM	Structural Matrix Assembler, Phase 1
SM2COM	Structural Matrix Assembler, Phase 2
EDTCOM	Static Solution Generator, Phase 2
OFPCOM, HEDCOM	Output File Processor
DSA1	Differential Stiffness Matrix Generator, Phase 1

ERROR-HANDLING

If a preprocessor run produces no error messages, the user should make an attempt to compile the new subroutines and BLOCK DATA subprograms to make sure that he has not coded "illegal" FORTRAN. If the output compiles correctly, the decks are ready to be linked into NASTRAN.

When a user looks at his generated routines he may find that the dimensions of variables have been reversed, or that calls have been made to unknown subroutines, or that there are some inefficiencies or other seeming peculiarities in the coding. Since the preprocessor only partially checks the FORTRAN coding in term-by-term definitions, it is conceivable, although not likely, that the preprocessor might code incorrect FORTRAN. Some of these situations may be caused by NASTRAN-imposed restrictions; others may be caused by intentional preprocessor shortcuts designed to save computing time. The obvious inefficiencies may be corrected manually by the user, exercising extreme caution. By changing the punched card output directly, the user may correct a simple FORTRAN error in the specification

of a term-by-term definition or make a slight change in the coding. However, changes to correct coding peculiarities resulting from NASTRAN-imposed restrictions should be made only by the experienced NASTRAN programmer.

Error messages returned by the preprocessor are explained in Appendix A. After the user has identified and corrected the error, he resubmits his data for another run. He may not need to resubmit his entire data deck, since some of his packets may have been successfully processed. An error in a particular packet causes the preprocessor to discontinue the processing of that packet and to start the processing of the next logical packet.

RESUBMITTING DATA

When errors are to be corrected or changes are to be made in the various data packets, the appropriate packets must be submitted. For an initial run, the Preliminary Data Packet and the Stiffness Matrix Packet must both be included, among with other optional packets to data to be processed. In successive runs, only the Preliminary Data Packet must be included with those packets to be changed or corrected. The first card in the input data will indicate whether or not the run is an initial one. Table 13 notes the packets needed in changing the various quantities of the Preliminary Data Deck. Note that this table applies only to the Preliminary Data Packet; changes to quantities in other packets might likely involve additional packets. Obviously, if a packet was not needed for an original run, it will not be needed for a repeat run. The same would hold true for the use of the special Subpackets A of the Stress Matrix Packet and the Stress and Force Packet.

If the preprocessor finds an error in the Preliminary Data Packet itself, the entire data deck must be resubmitted when the corrected Preliminary Data Packet is resubmitted, since the preprocessor will make no attempt to process other packets if it finds an error in the Preliminary Data Packet.

When a coding error is detected in a data packet, the user should discard all FORTRAN coding generated by the preprocessor. A complete routine will be generated when the corrected data packet is submitted.

TABLE 13 - INPUT PACKETS NEEDED FOR PRELIMINARY-DATA-DECK CHANGES

Quantity to be Changed	Packets Needed
Element name	Preliminary Data Stress Matrix (Special Subpacket A, only) Stress and Force (Special Subpacket A, only) Output (Note: All subroutine names must be changed manually.)
Number of grid points	All, except for Output
Degrees of freedom	All
Scalar indicator	None. At the present time, new elements may not be scalar elements
Number of permanent elements	Preliminary Data Stress Matrix (Special Subpacket A) Stress and Force (Special Subpacket A) Output
Approach acceptability flag	Preliminary Data Stress Matrix (Special Subpacket A) Stress and Force (Special Subpacket A)
Sequence of digits for connection card; Sequence of digits for property card	
Number of data items on connection card	All, except for Output. (Only the Special Subpacket A of Stress and Force is required.)
User-defined connection card variables	
Number of data items on property card	
User-defined property card variables	

APPENDIX A
ERROR MESSAGES

There are 101 situations which will cause an error message to be printed. Following is a listing of these error messages and the corresponding numbers which appear in the output, together with an explanation of each error.

1 BAD ELEMENT NAME: X

The user has not coded the new element name in the correct manner which is ELEMy, where y is an integer less than 51 and greater than 38. The Preliminary Data Packet generates this error. X is the erroneous element name defined through the input.

2 NUMBER OF GRID POINTS IN ERROR: X

The number of grid points for the new element, the second data item in the preliminary data, is in error. This number must be an integer greater than zero and less than 101. X is the erroneous value obtained through the input data.

3 SEQUENCE OF DIGITS IN ERROR: X

The sequence of integers specifying the degrees of freedom for the new element is in error. This value is the third data item in the preliminary data, and it must be a string of integers each of which is greater than zero and less than seven. No integer may appear twice. X is the erroneous value.

4 SCALAR ELEMENT INDICATOR IN ERROR: X

The integer designating whether the new element is a scalar element or a structural element is in error. As scalar element implementation for the preprocessor does not yet exist, this value must be zero. This error comes from the interpretation of the preliminary data; x is the erroneous value.

5 NUMBER OF PERMANENT ELEMENTS IN NASTRAN ERROR: X

The integer in the preliminary data designating the number of permanent NASTRAN elements is in error. It must be an integer greater than 37 and less than 50. X is the value in error.

6 APPROACH ACCEPTABILITY FLAG IN ERROR: X

The approach acceptability flag, found in the preliminary data, is in error. It must be an integer $y < |3|$; X is the erroneous value.

7 NUMBER OF DATA ITEMS ON CONNECTION CARD IN ERROR: X

The integer in the preliminary data designating the number of variable names on the connection card is in error. It must be an integer greater than zero and less than 97-5N, where N is the number of element grid points. X is the erroneous value. Trying to assume a default also produces this error.

8 NUMBER OF DATA ITEMS ON PROPERTY CARD IN ERROR: X

The integer in the preliminary data designating the number of variable names on the property card is in error. It must be an integer greater than -1 and less than 97-5N, where N is the number of element grid points. An attempt by the user to obtain a default value will also produce this error message. X is the erroneous value.

9 SEQUENCE OF INTEGERS FOR CONNECTION CARD IN ERROR: X

The sequence of integers designating FORTRAN variable type for all connection card variables is in error. Each integer must range between zero and five; there must be an integer for each connection card variable. However, since the preprocessor can presently handle only user-defined integer and real variables, the integers must all presently be 1 or 2. This value is the default value. X is the erroneous value found through interpretation of the preliminary data.

10 SEQUENCE OF INTEGERS FOR PROPERTY CARD IN ERROR: X

The sequence of integers designating FORTRAN variable type for all property card variables is in error. Each integer must range between zero and five, and there must be an integer for each property card variable. However, since the preprocessor can presently handle only user-defined integer and real variables, these integers must all presently be 1 or 2. This value is the default value. X is the erroneous value.

11 LIST OF VARIABLES ON CONNECTION CARD IN ERROR: X

One of the variables found on the connection card does not conform to FORTRAN rules for the construction of variable names, or the name is a NASTRAN common variable. X is the variable name in error.

12 LIST OF VARIABLES ON PROPERTY CARD IN ERROR: X

One of the variables found on the property card does not conform to NASTRAN rules for the construction of variable names, or the name is a NASTRAN common variable. X is the erroneous variable name.

13 THE NUMBER OF CONNECTION AND PROPERTY CARD VARIABLES IS GREATER THAN X.

The total of connection and property card variables is greater than X, the maximum number allowed, which is 97-5N, where N is the number of element grid points.

14 NUMBER OF ITEMS FOR PROPERTY CARDS DOES NOT MATCH ACTUAL NUMBER FOUND.

The preprocessor generates this error message if the number of FORTRAN variable names specified as existing on the property card does not equal the number of names specified earlier as the eighth parameter in the preliminary data.

15 PRELIMINARY ERRORS EXIST, ERROR CHECKING WILL CONTINUE

This error message appears if the user has coded an erroneous Preliminary Data Packet. Error checking for the remainder of the preliminary data will occur, whereupon the program will halt execution.

16 NO DATA AT ALL EXISTS - EXECUTION ABANDONED

The preprocessor prints this error message if execution is attempted with no input data set.

17 INPUT EXHAUSTED WHILE READING PRELIMINARY DATA

The input data deck did not contain enough cards to permit full processing of the preliminary data. The entire set of input data has been read.

18 NUMBER OF DATA ITEMS FOR CONNECTION CARD DOES NOT MATCH ACTUAL NUMBER FOUND

The number of FORTRAN variable names specified as existing on the connection card does not equal the integer specified as the seventh preliminary data parameter.

19 INFLAG/OPTION CARD IN ERROR: X

The user has coded the inflag/option card for a particular packet in an erroneous manner. X is the representation of the bad card.

20 INFLAG/OPTION CARD DOES NOT EXIST

The preprocessor generates this error message if the input data deck has been completely read prior to an attempt to read an INFLAG/OPTION card.

21 ERRORS IN PRELIMINARY DATA PREVENT FURTHER PROCESSING

If any error exists in the Preliminary Data Packet, this error message appears in the output. The preprocessor halts execution and all subsequent input data is ignored.

22 NUMBER OF STRESS WORDS ERROR: X

The first value on the options card for the Stress and Force Calculation Packet is in error. This value must be greater than or equal to zero or less than or equal to 100. X is the erroneous value.

23 NUMBER OF FORCE WORDS ERROR: X

The second value on the options card for the Stress and Force Calculation Packet is in error. This value must be greater than or equal to zero and less than or equal to 100. X is the erroneous value.

24 OPTION ERROR FOR PACKET: X

The second value on the inflag/option card for the Stiffness Matrix, Mass Matrix, Viscous Damping Matrix, Differential Stiffness Matrix, or Thermal Loading Vector Packet is in error. X is the incorrect value.

25 TERMINATION OF EXECUTION REQUESTED

An earlier error could not be rectified. As further processing is futile, the preprocessor halts execution.

26 INPUT EXHAUSTED WHILE READING MAIN EQUATION

All data had been read prior to an attempt to read the main expression (e.g., Stiffness expression). For the Global Variable Packet, this card is the list of global variables. For the Stress Matrix and Stress and Force Calculation Packets, this card should be a list of variables to be loaded into the NASTRAN array PH1OUT.

27 ERROR IN GETVAR

This error message indicates that a terminal error has occurred in the definition of user variables (Subpacket B), and all variables cannot be defined. The usual cause for this error is an absence of input data; preprocessor execution stops at once. Earlier messages should point to the reason for the error.

28 ERROR IN DEFELEM

The input data was exhausted while processing Subpacket C. The error message is fatal; preprocessor execution halts immediately. Earlier error messages should point to the reason for the error.

29 NO OPERATOR IN HIERARCHY-INPUT IS: X

The user has coded an erroneous equation. This error message comes from the infix notation to Polish notation convertor. Error message 30 should appear as a pointer to the bad equation. The error comes from the interpretation of a main expression (e.g., stiffness expression) or an equation in a particular Subpacket C. X serves as a relative pointer to the position of the error in the equation. Typical sources of errors are unbalanced parentheses, unrecognizable operators, and failure to code an operator between variable names (e.g., AbB to mean A+B, where b is a blank character).

30 USER HAS CODED AN ERRONEOUS EQUATION: X

The bad equation is given by X. Something in the equation confuses the Polish convertor.

31 VARIABLE NAME IS TOO LONG: X

The user-defined variable name X, which appears in a Subpacket B, is longer than six characters. X is the bad name.

32 DEFINITIONS FINISHED CARD IS MISSING

The preprocessor attempted to read the next data packet to resolve all undefined variable names. The user should look for the absence of a "DEFINITIONS FINISHED" card at the end of a particular Subpacket B.

33 PREPROCESSOR HAS TRIED TO READ NEXT PACKET TO DEFINE ALL VARIABLES

An undefined variable(s) exists in a particular packet. Look at the table of variables at the end of the packet output prior to the code generation for the undefined variable(s).

34 USER HAS TRIED TO DEFINE THE NULL STRING AS A VARIABLE

Subpacket B contains a mispunched card. A typical error would be trying to assume a default and misplacing the necessary comma. In any case, the first field on the card is incorrect.

35 ILLEGAL ATTEMPT BY USER TO REDEFINE THE RESERVED NAME: X

The user has attempted to redefine a NASTRAN common variable or preprocessor-defined variable. The use of the name appears in Subpacket B for the current packet under consideration. X is the variable name in error.

36 NOT ALL VARIABLES CAN BE DEFINED - INPUT EXHAUSTED

Subpacket B is in error for the current packets. Typical sources of errors are mispunched variable names, undefined variables, and the absence of a "DEFINITIONS FINISHED" card. In any case, all input has been read and processed; this error message forms part of a terminal error message. Error message 27 should also appear.

37 THE FOLLOWING VARIABLES ARE UNDEFINED: X

In the interpretation of the main expression (e.g., stiffness expression) and Subpacket C for the current packet, the user has not defined some packet variables. X is the list of unresolved variable names.

38 THE FOLLOWING VARIABLE MUST APPEAR AND DOES NOT: X

In the Stiffness Matrix, Mass Matrix, Thermal Loading Vector Viscous Damping Matrix, and Differential Stiffness Packets, the main variable (i.e., stiffness matrix, mass matrix, etc.) does not appear. It must appear for the packet to have a meaning. The forms X may assume are as follows:

(1) KIJ - stiffness matrix
(2) MIJ - mass matrix
(3) VIJ - viscous damping matrix
(4) PPI - thermal loading vector
(5) DIJ - differential stiffness matrix

39 DIMENSIONS ARE WRONG FOR VARIABLE X

In the Stiffness Matrix, Mass Matrix, Viscous Damping Matrix, Thermal Loading Vector and Differential Stiffness Packets, the dimensions of the main variable (mass matrix, stiffness matrix, etc.) are in error. For an interpretation of X, see error message 38 above. KIJ, MIJ, VIJ, and DIJ may have the dimensions 3X3, 6X6, or 6NX6N, where N is the number of grid points for the new element. PPI may have the dimensions 3X1, 6X1, or 6NX1. All of the above variables may be defined as a 1X1 scalar.

40 THE VARIABLE UNDEFINED IN THE VARIABLE VECTOR IS: X

In Subpacket C for a particular packet, the user defined a variable which was never used in an equation. X is the variable name in question.

41 THE NUMBER OF UNDEFINED VARIABLES IS: X

In Subpacket C for a particular packet, X variables are undefined either through the NASTRAN COMMON area or through Subpacket B.

42 THIS PACKET IS ALREADY DEFINED: X

The user has defined two packets with the same name, or a card exists in the input data with the string "BEGIN X", where X is a packet name. In the former case, remove one of the packets. For the latter, remove the indicated string. A typical error would be a "COMMENT" control card containing the word "BEGIN".

43 PACKET NAME IN ERROR: X

This header card for a packet has a mispunched packet name. X is the mispunched name. It appears on a "BEGIN" card.

44 PACKET ORDERING IS IN ERROR

The ordering of the various packets within the input data is in error. The correct ordering is:

(1) Preliminary Data Packet
(2) Global Variable Packet
(3) Stiffness Matrix Packet
(4) Mass Matrix Packet
(5) Viscous Damping Matrix Packet
(6) Thermal Loading Vector Packet
(7) Stress Matrix Packet
(8) Stress and Force Calculation Packet
(9) Output Packet
(10) Differential Stiffness Packet

45 DATA EXHAUSTED IN THE GLOBAL PACKET

The preprocessor ran out of data while processing the Global Variable Packet.

46 UNDEFINED VARIABLE FOUND: X

An unresolved variable occurred in an abnormal part of the preprocessor coding. The printout of this error message indicates an error in the preprocessor logic, and the printout should be sent to the authors to determine the cause for the error. X is the erroneous variable name.

47 SUBSCRIPT OF VARIABLE IS IN ERROR - EQUATION OF: X

In the term-by-term definition of variable X, one of the variables used has an illegal subscript. For example, if A had dimensions of 3X1 and was referenced as A(1,3), this error message would occur.

48 NEITHER STRESS NOR FORCE WAS SPECIFIED FOR THE OUTPUT PACKET: X

In the output packet data, the card indicating whether the headings to be produced are for stresses or forces is in error. X is the value in error. A typical error would be:

```
.
.
.
BEGIN OUTPUT
 STRESbSES
.
.
.
```

X must be either the word "STRESS" or the word "FORCE". In this case, X is the bad value "STRESSES".

49 OUTPUT PACKET INDICATORS IN ERROR: X

In the portion of the Output Packet devoted to either stresses or forces, the indicators used to compute variable subscripts are in error. The three fields have the following permissible values:

(1) Field 1: SORT1 OR SORT2

(2) Field 2: REAL or COMPLEX

(3) Field 3: REAL/IMAG or MAGNITUDE/PHASE (if necessary)

Error message 49 will appear if any of these three fields is mispunched. X is the list of the three user-supplied indicators.

50 INPUT EXHAUSTED WHILE READING OUTPUT PACKET

The preprocessor ran out of input data during the processing of the Output Packet.

51 HEADING FORMAT STATEMENT NUMBER IS ERROR: X

In the Output Packet, the user attempted to insert a new format heading which had an erroneous value X. X must fall in the closed interval [217,336] or be either zero or minus one. Sources of this error are mispunched data cards or attempting to define a statement number whose value is not allowed. X appears on the card containing the six integers.

52 FORMAT NUMBER FOR NEW FORMAT IN ERROR: X

While attempting to define one of the new format headings coded on the card containing the six integers, the user coded a format number which did not match one of the six integers. The user has therefore mispunched his heading format number. X is the erroneous value.

53 NEW FORMAT LENGTH IS GREATER THAN 34 WORDS: X

In the specification for a new format heading, the user has coded a format whose length is greater than 34 words (133 characters). X is the number of words used by the erroneous format.

54 FORMAT MISSING LEFT PARENTHESIS AND/OR RIGHT PARENTHESIS: X

The user has left off either the starting left parenthesis or the terminal right parenthesis or both for the specification of a new format heading. All new format headings must begin with a left parenthesis and end with a right parenthesis. X is the erroneous format.

55 FORMAT NUMBER TO BE LOADED IS IN ERROR: X

The user has attempted to append more than five new format heading numbers; only five are allowed. X is the sixth new format number which is not allowed. Note that there may be more than 6 erroneous format header numbers.

56 NEW FORMATS CONSUME MORE THAN 20000 CHARACTERS: X

The total number of characters used by the five new format headings is greater than 20000. X is the erroneous total number of characters.

57 MORE THAN 45 FORMAT PIECES ARE SPECIFIED

The user has attempted to insert more than the maximum number (45) of format pieces.

58 ERROR IN WRFORT

The preprocessor encountered a user-error in the portion of the coding devoted to the generation of FORTRAN coding. Other error messages should point to the actual reasons for the program failure.

59 NUMBER OF PIECES SPECIFIED DO NOT MATCH ACTUAL NUMBER FOUND

The user has defined more format pieces than he coded on the card containing the format pieces to be defined. An example would be 1, 74, -4, 30, 0 on the format piece definition card followed by

```
74
1X, I7
30
1PE11.4, 0PF8.1, 3X
31
1PE10.3, 0PF7.0, 3X
```

Format piece 31 was not coded on the above card; therefore; error message 59 appears in the output.

60 I VALUE FOR FORMAT PIECE IS NOT IN THE PROPER RANGE: X

If a format piece defined by the user has a value I such that

(1) $I < -40$ or
(2) $I > 100$ or
(3) $-32 < I < 29$ or
(4) $39 < I < 74$,

then error message 60 appears in the output. I may range in the two closed intervals [-40, -1] and [1, 100], but the user may define I only for I in [-40, -32], [29, 39] or [74, 100]. Therefore, this error has several meanings. First, an erroneous I value may have been punched by the user, i.e., an I value not in the allowable ranges. Secondly, he may be trying to define an I value which may not be defined. X is the erroneous I value.

61 ERROR FLAG SET - NO CODING WILL BE GENERATED

The preprocessor found a user data error prior to FORTRAN code generation in one of the following packets: Stiffness, Mass, Thermal, Loading, Stress Matrix, Differential Stiffness, or Stress and Force Calculation. No FORTRAN code will appear in the output for this packet until its errors are corrected.

62 USER HAS TRIED TO TRANSPOSE AN ENTIRE MATRIX EQUATION

The user currently may not transpose a matrix equation. An example would be

$$TR\ (B*C)^{A}$$

in Subpacket C of a particular packet where A, B, and C are previously defined matrices. This restriction will not exist in later versions of the preprocessor. This condition is detectable only during FORTRAN code generation. Upon finding the error, code generation stops immediately.

63 ILLEGAL USE OF BINARY OPERATOR IN A UNARY FASHION: X

The user has coded an erroneous equation in a particular Subpacket C. Binary operators such as * and / expect two operands, e.g.,: A*B where A and B are operands. Use of such an operand in a unary fashion (e.g.,: C = *A) will produce this error message. X will contain the operator in error as well as the variable definition under which this error occurs. This error appears during FORTRAN code generation; its appearance halts the code generation at once.

64 USER SPECIFIED WRONG DIMENSIONS FOR: X

In Subpacket B for a particular packet, the user defined a variable with dimensions D1 and D2. However, the preprocessor has determined during FORTRAN code generation that the variable should have dimensions D3 and D4 such that D1 $\neq$ D3 and/or D2 $\neq$ D4. Code generation is therefore halted. X contains the variable in error as well as a printout of D1, D2, D3, and D4.

65 AN UNDEFINED VARIABLE HAS ONLY NOW BEEN FOUND: X

The preprocessor discovered an undefined variable during FORTRAN generation. As all variables should be resolved during the interpretation of Subpacket B, this error indicates an error in the preprocessor logic. The user should return the output to the authors to determine the reason for the error. X is the undefined variable.

66 MULTIPLICATION CANNOT TAKE PLACE AS MATRIX DIMENSIONS DO NOT MATCH: X

During the interpretation of a user-defined matrix equation, the preprocessor attempted to multiply two matrices whose dimensions would result in improper multiplication. An example would be trying to multiply two 3x1 matrices. X contains both matrices and their preprocessor-determined dimensions.

67 ADDITION CANNOT TAKE PLACE AS MATRIX DIMENSIONS DO NOT MATCH: X

During the interpretation of a user-defined matrix equation, the preprocessor attempted to add two matrices whose dimensions would result in improper multiplication. An example would be trying to add a 1x3 matrix to a 3x1 matrice. X contains both matrices and their preprocessor-determined dimensions.

68 EXPONENTIATION CANNOT TAKE PLACE AS MATRIX IS NOT SQUARE: X

The user has attempted to raise a matrix to a power. However, the matrix is not square. X contains the matrix name and its preprocessor-determined dimensions.

69 NEGATIVE OR ZERO EXPONENTIATION CANNOT TAKE PLACE: X

The user has attempted to raise a matrix to the zero or a negative power. X contains the matrix name and its preprocessor-determined dimensions.

70 USER MAY NOT USE G MATRIX WITH AN INFLAG VALUE OF X

The user may use the predefined materials matrix G only with INFLAG values of one, two or three. All other INFLAG values will produce errors.

71 NUMBER OF VALUES TO BE LOADED INTO PH1OUT FROM OPTIONS CARD DOES NOT MATCH ACTUAL NUMBER COMPUTED: X

On the INFLAG/OPTION card in either the Stress Matrix or the Stress and Force Calculation Packet, the user coded an integer specifying how many values were to be loaded into PH1OUT. However, the preprocessor has determined that more variables than this earlier specification will be used. As an example, let the options card for the Stress Matrix Packet show 2, 35 indicating that 35 words are to be inserted into PH1OUT. Now let the card containing the variables to be inserted into PH1OUT be TSUB0, SB1, SB2, SB3, ST, G where

(1) TSUB0 is a scalar
(2) SB1 is a 3X3 matrix
(3) SB2 is a 3X3 matrix
(4) SB3 is a 3X3 matrix
(5) ST is a 3X1 vector
(6) G is a 3X3 matrix

Therefore, 40 variables are to be inserted from this card. Since these two values are not equal, the preprocessor generates error message 71. X is the number of words to be inserted into PH1OUT computed from the card containing the variable names.

72 THERE ARE MORE VALUES TO BE LOADED INTO PH1OUT THAN THERE ARE LOCATIONS AVAILABLE: X

The user may not insert more than 100 variables into PH1OUT from the Stress Matrix Packet and 100 each for stresses and forces from the Stress and Force Calculation Packet. X is the erroneous number of variables which the user is attempting to insert.

73 ILLEGAL USE OF A FORTRAN FUNCTION IN THE MATRIX DEFINITION OF X

The user may not use FORTRAN functions in any matrix equation definition in any Subpacket C. X is the variable being defined in which the error exists.

74 THERE EXIST UNRESOLVED DEFINITIONS IN THE OUTPUT PACKET

The user has coded an undefined format heading or format piece number in the Output Packet. One of the following two cards contains the error.

(1) Card containing the six integers
(2) Card containing sequence of N+1 integers

75 A FORMAT PIECE WITHIN THE RANGE FROM 29 TO 39 IS GREATER THAN 20 CHARACTERS: X

Let X be a format piece under consideration whose value falls into the closed interval [29, 39]. Now the format to be inserted for piece X contains two fields, a standard format plus an alternate format. The preprocessor separates these two fields and adds enough blanks in front of the first to form an eight-character field. If the sum total of characters found from the modified field one and the original field two is greater than twenty, this error message appears in the output.

76 A VARIABLE WAS FLAGGED EARLIER AS GLOBAL BUT DID NOT APPEAR IN THE LIST OF GLOBAL VARIABLES: X

The preprocessor determined that a user-defined variable was global; however, the variable was found not to be a global variable. X is the erroneous variable name. This error indicates an error in preprocessor logic, and the output should be returned to the authors to determine the reason for the error.

77 MATRIX IS NOT SQUARE: X

The user has attempted to define a non-square matrix partition. Depending upon which data packet is under consideration (Stiffness Matrix, Mass Matrix, Viscous Damping Matrix, or Differential Stiffness); the user should check his intermediate equations which produce the partition as to proper dimensions. X is the illegal partition, and the illegal dimensions follow X in parenthesis.

78 FORTRAN INSERTION SUBPACKET NAME NOT FOUND IN THE LIST OF PACKET VARIABLES: X

During the interpretation of the current subpacket, the preprocessor found a user-defined FORTRAN insertion subpacket name which was not defined in the previous Subpacket B. The preprocessor therefore could not determine the ranking of the insertion subpacket relative to the other packet variables. X is the undefined subpacket name.

79 THE SOFTWARE FOR THIS PACKET IS NOT YET AVAILABLE

The user has attempted to use the Piecewise Linear Analysis - Stiffness Matrix Packet or the Piecewise Linear Analysis - Stress Matrix Packet. The use of either of these packets by the analyst is illegal.

80 INPUT EXHAUSTED WHILE READING UPDATE PRELIMINARY DATA PACKET

The input data deck has been exhausted during the reading of the interpreted Preliminary Data Packet punched by the preprocessor from an earlier computer run. The user should check for an extraneous end-of-file or end-of-record terminator in the middle of his data deck.

81 ILLEGAL ATTEMPT TO INVERT A NONSQUARE MATRIX - X

During interpretation of a user-supplied matrix equation, the preprocessor has found an attempt to compute the numerical inverse of a matrix which is not square. X is the illegal matrix in question, and it is followed by the name of the variable being defined whose definition calls for the illegal inversion.

82 ILLEGAL ATTEMPT TO INVERT A SCALAR - X

The user may not try to use the INV function to compute reciprocals. X is the variable name in question.

83 A TERM-BY-TERM DEFINED VARIABLE HAS ALL ZERO ELEMENTS: X

In any Subpacket C supplied by the user, he may not define a variable whose elements are all zero. At least one of the elements of a vector or matrix must be non-zero. All scalars must be non-zero. If the user desires a variable (scalar, vector, or matrix) to be ideally zero, he must use a FORTRAN insertion packet to perform the function. X is the all-zero variable name.

84 A VARIABLE NAME WAS NOT FOUND WHERE EXPECTED

In a typical Subpacket C, the user codes his variable definitions in the following manner. His first card contains the variable name to be defined. His second and subsequent cards (if needed) contain the actual definition. For this error message to be produced, the user must be employing a term-by-term definition. The preprocessor has read the card containing the variable name and a number of subsequent cards to define all elements of that variable. The program then loops to obtain another variable name. However, such a card was not found, and the error message indicates this condition. The typical source of this error is in defining a particular element of a variable more than once or in mispunching the variable name card such that the name is not entirely alphanumeric.

85 ILLEGAL SUBSCRIPT ENCOUNTERED; DATA CARD IGNORED

A term-by-term definition card has three fields: the first subscript, the second subscript, and the definition for the element of the current variable being defined having these subscripts. The first and second fields must each be purely numerical; they may not have alphabetic information. The illegal card number will be included in the error message.

86 ILLEGAL DIMENSION ENCOUNTERED; PACKET PROCESSING STOPPED

In the current Subpacket B under consideration, the user has coded a variable's first or second dimension which is not entirely numeric. As the packet FORTRAN coding produced depends exactly upon these dimensions of the variables, all processing of the current data packet is suspended. The error message will contain the card number of this card containing the illegal dimension.

87 ILLEGAL MANNER OF DEFINITION IN SUBPACKET B: PACKET PROCESSING STOPPED

In Subpacket B, the user has employed an illegal manner-of-definition field. Therefore, the field contains data which is not EQUA, TERM, COMM, or DEFER. All subsequent data in the packet is ignored, and the preprocessor proceeds if possible to the subsequent data packet submitted by the user.

88 PREVIOUS MATRIX EQUATION DIMENSION MISMATCH: X

In a matrix equation definition for a Subpacket B variable, the dimensions defined for that variable in Subpacket B do not match those dimensions dynamically computed by the preprocessor's interpretation of the matrix equation. As an example, let variable X have dimension 2X3 defined in Subpacket B. Let the definition of X be Y*Z in Subpacket C, where Y and Z are 1X3 and 3X1 vectors respectively. The resolution of the equation Y*Z gives X dimensions of 1X1, or a scalar. But X was defined as a 2X3 matrix in Subpacket B. Therefore, the error message

is generated, and generation of the current FORTRAN subroutine is halted. In the generation of a stiffness matrix partition, a mass matrix partition, a thermal loading vector, a viscous damping matrix partition, and a differential stiffness matrix partition, the dynamic dimensions may be less than or equal to those dimensions defined in Subpacket B. X is the ill-defined variable followed by the two pairs of dimensions.

89 TOO MANY PACKET VARIABLES: INCREASE DIMENSION SIZE

In the current packet under consideration, the user has defined more than 75 distinct variable names. The user should employ the PACKET VARIABLES parameter card to increase the number of allowable user-defined variable names. If such a card is already in use, he should increase the integer field of the card.

90 ILLEGAL SPECIFIER FOR MATRIX GENERATION BY PARTITIONS: X

In the matrix equation definition of a matrix by partitions, the data card defining a particular partition does not contain one of the keywords UL, UR, LL, or LR. As an example, let variables A, B, C, and D be previously-defined 2X2 user matrices and E be a 4X4 matrix defined as follows:

$$E = \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right]$$

In Subpacket B, the user would define E to be a 4X4 variable defined through a matrix equation. The user might then code in Subpacket C to define E:

```
E
UL, A
UR, B
LL, C
LR, D
```

Suppose, however, he punches the following:

```
E
UL, A
UR, B
LQ, C
LR, D
```

The card containing LQ, C is in error as LQ does not describe any partition of the matrix E. X is the variable being defined that contains the illegal partition definition.

91 MORE THAN FOUR PARTITIONS SPECIFIED: X

The user has provided more than four partition definitions in the matrix equation definition of X.

92 MATRIX MUST BE SQUARE AND NONSCALAR FOR PARTITION DEFINITION: X

For the user to be able to define the matrix X by the partition method, X must have dimensions NXN where N is a positive even integer.

93 ONLY VARIABLE NAMES AND TRANSPOSES MAY BE USED IN GENERATING PARTITIONS: X

On the cards defining the partitions for user variable X, the matrix equations defining the particular partition must be of the forms A, or TR(A), where A is a previously defined user variable. To reiterate, full matrix equations may not be used to define partitions of a variable. The user should define an intermediate variable with his full matrix equation for a particular partition definition. He would then use that intermediate variable name in the partition definition.

94 DIMENSION MISMATCH DOES NOT ALLOW PARTITION GENERATION: X

Let variable X have dimensions NXN, where N is a positive, even integer. The four partition definitions which make up the definition of X must have resultant dimensions of $\frac{N}{2}$ X $\frac{N}{2}$. Refer to the example described in error message 90 for the following. To define the 4X4 matrix E, the variables A, B, C, and D must be 2X2 matrices. Any other dimensioning for any of the four definition matrices would produce an error condition.

95 USER MAY NOT REDEFINE A COMMON VARIABLE: X

The user is not allowed to define any variable which is declared to be a NASTRAN defined COMMON variable. In addition, any variable described by the user to be COMMON by way of the COMMON keyword in Subpacket B may not be defined in the current Subpacket C or in any subsequent packet definitions. Once a variable name is declared to be a COMMON variable, it retains that status throughout the remainder of the job. However, prior to the variable's declaration as a COMMON variable, it may be redefined and otherwise treated as any other local packet variable.

96 ERRONEOUS ROW/COLUMN SPECIFIED: X

Let variable X have dimensions MXN, where M and N are positive integers. The user may not define a row number greater than M or a column number greater than N. Negative row and column numbers also produce the error message. If X has dimensions 2X3, he may define X by defining rows one and/or two or by defining columns one, two, and/or three. He may not combine row and column definitions to define a particular variable.

97 ROW AND COLUMN DEFINITIONS CANNOT BOTH BE USED: X

The user may not define variable X by using both row and column definitions. He must define X by using only row definitions or by using only column definitions. He would otherwise redefine one or more elements of matrix X, thus producing an error condition.

98 ONLY VARIABLE NAMES AND TRANSPOSES MAY BE USED IN GENERATING BY ROWS OR COLUMNS: X

In the row or column definitions of user variable X, the matrix equations defining the particular rows or columns must be of the form A, or TR(A), where A is a previously defined user variable. Full matrix equations are not allowable. If a particular row or column is defined by a full matrix equation, the user should define an intermediate variable to be the matrix equation and use the intermediate variable name in his row or column definition.

99 DIMENSION MISMATCH DOES NOT ALLOW GENERATION BY ROWS OR COLUMNS: X

Let the user variable X have dimension MXN, where M and N are positive integers. Furthermore, let X be defined by a series of variables A(1), A(2),..., A(M) defined through the row definition manner of matrix equation definition. For the definition of X by A(1), A(2),..., A(M) to be legal; A(1), A(2),.., A(M) must each have one of the two following dimensions:

1XN, or

NX1.

The same argument is true for definition by columns. If the N columns of X are defined by B(1), B(2),..., B(N), each of the B(1), B(2),..., B(N) must have dimension 1XM or MX1. Of course, the user may leave one or more rows or columns undefined, thus defaulting their definitions to zero.

100 ERRONEOUS MAIN EXPRESSION: NO X OR Y

On the main expression card for the Stiffness Matrix, the Mass Matrix, the Viscous Damping, the Thermal Loading, or the Differential Stiffness Matrix Packet, the user has failed to code the main variable name or the general main variable partition. For example, in the Stiffness Matrix Packet he has coded neither K or KIJ on the main expression card. X and Y are the two values from which the user may choose (K and KIJ in the example).

101 ILLEGAL SUBSCRIPTED VARIABLE: X

In the definition of user variable X, the user has referred to a variable Y with subscripts I and J. One or both of the two subscripts, I or J do not conform to the dimensions of Y described in the preceding Subpacket B. For example, let Y have dimensions 2X2; the user may not refer to Y(3,3), as no such element of Y exists.

B L A N K P A G E

APPENDIX B
SAMPLE PROBLEMS

Two separate examples follow. In the first sample problem, all of the data packets supplied in this report as examples in the discussions of the various data packets have been combined to form a complete set of input that will produce the FORTRAN routines and tables that are necessary for adding the triangular membrane (TRMEM) element into NASTRAN. Such an element has actually been incorporated within NASTRAN, using the cards shown in the examples. The input packets provided and the generated routines or tables associated with each are included as Figures 11 through 25. For interpretation, the reader should refer to the descriptions of the packets provided in Section III.

The second sample problem is concerned with the production of the FORTRAN routines and tables necessary to generate a three-dimensional isoparametric thermal element. Figure 26 contains input data needed for the three-dimensional isoparametric thermal element. Figure 27 contains the subroutines and tables generated from the data.

The time used in producing the output of the first sample problem was approximately six minutes, using a CDC 6400 computer and 250000_8 words of central memory. The CPU time used in the second sample problem was eight minutes, even though only two packets of input were used.

```
PARAM = 1
COMMENT
COMMENT  THE FOLLOWING INPUT DATA WILL ADD ELEMENT ELEM40 TO NASTRAN
COMMENT  ELEM40 IS JUST THE TRIANGULAR MEMBRANE ELEMENT(TRMEM) NOW IN
COMMENT  NASTRAN
COMMENT
COMMENT BEGIN PRELIMINARY DATA
COMMENT
ELEM40,3,12,,,,1,2,2,22
COMMENT  CONNECTION CARD VARIABLE
TH
COMMENT  PROPERTY CARD VARIABLES
T,EMU
END PRELIMINARY DATA PACKET
```

Figure 11 - Preliminary Data Packet, Sample Problem 1

```
      BLOCK DATA
      COMMON/IFPCOM/NOELEM,IAPFLG(24),IRANOS(48),IFX7PT(24),IFX7SQ(384)/
     1IFSCOM/NLEM,IFSN(24)
      DATA NOELEM,IAPFLG,NLEM/38,2*0,0,0,20*0,38/
      DATA IRANOS/4*0,8,12,4,8,40*0/
      DATA IFX7PT/2*0,42,1,20*0/
      DATA IFSN/2*0,6,4,20*0/
      END
      BLOCK DATA
      COMMON /GPTCOM/NOELEM,NDATCN(12),NDATPR(12),NGRDPT(12),INDSCA(12),
     1NWDEST(12),IFSTPT(12)
      DATA NOELEM/38/
      DATA NDATCN/0,6,10*0/
      DATA NDATPR/0,4,10*0/
      DATA NGRDPT/0,3,10*0/
      DATA INDSCA/0,0,10*0/
      DATA NWDEST/0,21,10*0/
      DATA IFSTPT/0,3,10*0/
      END
      BLOCK DATA
      COMMON /EDSCOM/NOELEM,NDATCN(12),NGRDPT(12)
      DATA NOELEM/38/
      DATA NDATCN/0,6,10*0/
      DATA NGRDPT/0,3,10*0/
      END
      BLOCK DATA
      COMMON/EDTCOM/NOELEM,NWDEST(12),NGRDPT(12)
      DATA NOELEM/38/
      DATA NWDEST/0,21,10*0/
      DATA NGRDPT/0,3,10*0/
      END
      BLOCK DATA
      COMMON /SM1COM/NOELEM,NWDEST(12)
      DATA NOELEM/38/
      DATA NWDEST/0,21,10*0/
      END
      BLOCK DATA
      COMMON /SM2COM/NOELEM,NWDEST(12)
      DATA NOELEM/38/
      DATA NWDEST/0,21,10*0/
      END
      BLOCK DATA
      COMMON /SDRCOM/NOELEM,NWDEST(12),NGRDPT(12),NWDSTM(12),NWDSTR(12),
     1NWDFOR(12),NPTSTR(12),NPTFOR(12)
      DATA NOELEM/38/
      DATA NWDEST/0,21,10*0/
      DATA NGRDPT/0,3,10*0/
      DATA NWDSTM/0,35,10*0/
      DATA NWDSTR/0,8,10*0/
      DATA NWDFOR/0,0,10*0/
      DATA NPTSTR/0,0,10*0/
      DATA NPTFOR/0,0,10*0/
      END
```

Figure 12 - BLOCK Data Subprograms, Sample Problem 1

```
BEGIN GLOBAL
COMMENT GLOBAL VARIABLE LIST
V12,XLV12,V13,XII,XKK1,XLV13,XKK,XJJ,E1,XX2,XX3,YY3,A,C1,C2,C3
V12,3,1,TERM
XLV12,,,
V13,3,,TERM
XII,3,,
XKK1,3,1,TERM
XLV13,,,
XKK,3,,TERM
XJJ,3,1,TERM
E1,3,2,
XX2,,,
XX3,,,EQUA
YY3,,,
A,,,
C1,3,2,
C2,3,2,
C3,3,2,TERM
DEFINITIONS FINISHED
V12
COMMENT
COMMENT  X1,...,Y1,...,Z1,...  ARE COMMON VARIABLES AVAILABLE FOR USE
COMMENT
1,,X2-X1
2,1,Y2-Y1
3,,Z2-Z1
V13
1,,X3-X1
2,1,Y3-Y1
3,1,Z3-Z1
XLV12
,,DSQRT(V12(1,1)**2+V12(2,1)**2+V12(3,1)**2)
XII
,,V12(1,1)/XLV12(1,1)
2,1,V12(2,1)/XLV12(1,1)
3,1,V12(3,1)/XLV12(1,1)
XKK1
1,1,XII(2,1)*V13(3,1)-XII(3,1)*V13(2,1)
2,1,XII(3,1)*V13(1,1)-XII(1,1)*V13(3,1)
3,1,XII(1,1)*V13(2,1)-XII(2,1)*V13(1,1)
XLV13
1,1,DSQRT(XKK1(1,1)**2+XKK1(2,1)**2+XKK1(3,1)**2)
XKK
1,1,XKK1(1,1)/XLV13(1,1)
2,1,XKK1(2,1)/XLV13(1,1)
3,1,XKK1(3,1)/XLV13(1,1)
XJJ
```

Figure 13 - Global Variable Packet, Sample Problem 1

```
1,1,XKK(2,1)*XII(3,1)-XII(2,1)*XKK(3,1)
2,1,XKK(3,1)*XII(1,1)-XII(3,1)*XKK(1,1)
3,1,XKK(1,1)*XII(2,1)-XII(1,1)*XKK(2,1)
E1
1,1,XII(1,1)
2,1,XII(2,1)
3,1,XII(3,1)
1,2,XJJ(1,1)
2,2,XJJ(2,1)
3,2,XJJ(3,1)
XX2
,,XLV12(1,1)
XX3
COMMENT
COMMENT  V13 IS A 3 X 1 VECTOR---TR(V13) IS A 1 X 3 VECTOR
COMMENT  XII IS A 3 X 1 VECTOR---SO XX3 IS A 1 X 1 VECTOR,
COMMENT  IE, A SCALAR
COMMENT
TR(V13)*XII
YY3
1,1, XLV13(1,1)
A
1,1,.5*XX2(1,1)*YY3(1,1)
C1
1,1,-1./XX2(1,1)
2,2,C1(3,1)
COMMENT  NOTE THAT ELEMENT (2,2) OF THE ARRAY IS DEFINED IN TERMS
COMMENT  OF ELEMENT (3,1)---THE REVERSE WOULD HAVE CAUSED PROBLEMS
3,1,1./YY3(1,1)*(XX3(1,1)/XX2(1,1)-1.)
3,2,C1(1,1)
C2
1,1,-C1(1,1)
2,2,C2(3,1)
3,1,-XX3(1,1)/(XX2(1,1)*YY3(1,1))
3,2,1./XX2(1,1)
C3
2,2,C3(3,1)
3,1,1./YY3(1,1)
END GLOBAL PACKET
```

Figure 13 - Global Variable Packet, Sample Problem 1—Continued

```
BEGIN STIFFNESS
2,KIJ
COMMENT  STIFFNESS MATRIX EXPRESSION FOR THE (I,J)TH PARTITION
4*T*KIJ
COMMENT  BEGIN LISTING OF STIFFNESS MATRIX PACKET VARIABLES
KIJ,3,3,EQUA
DEFINITIONS FINISHED
KIJ
TR(CI*TR(E1)*TI)*G*(CJ*TR(E1)*TJ)
END STIFFNESS PACKET
```

Figure 14 - Stiffness Matrix Packet, Sample Problem 1

STIFFNESS PACKET FORTRAN CODING

```
      SUBROUTINE KLEM40
      DOUBLE PRECISION Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8,Q9,PI,DETERM,TEMPOR,TMPOR
     11,G,T1,T2,T3,V12,XLV12,V13,XII,XKK1,XLV13,XKK,XJJ,E1,XX2,XX3,YY3,A
     1,C1,C2,C3,KI1,KI2,KI3,K11,K21,K31,K12,K22,K32,K13,K23,K33
      DIMENSION ECPT(1)
      DIMENSION TMPOR1(36)
      DIMENSION XYZ1(1,3),XYZ2(1,3),XYZ3(1,3)
      DIMENSION K11(3,3),K12(3,3),K13(3,3),K21(3,3),K22(3,3),K23(3,3),K3
     11(3,3),K32(3,3),K33(3,3)
      COMMON /MATIN/MATID,INFLAG,ELTEMP,STRESS,SINTH,COSTH/SMA1IO/DUM1(1
     10),IFKGG,DUM2(1),IF4GG,DUM3(23)/SMA1CL/IOPT4,K4GGSW,NPVT/SMA1ET/NE
     1CPT(1),NGRID(3),TH,MATID1,T,FMU,ID1,X1,Y1,Z1,ID2,X2,Y2,Z2,ID3,X3,Y
     13,Z3,DUMV(80)/SMA1DP/I,J,IS,IP,I1,I2,PI(1,1),TEMPOR(9),DETERM,Q1(9
     1),Q2(9),Q3(9),Q4(9),Q5(9),Q6(9),Q7(9),Q8(9),Q9(9),G(3,3),T1(3,3),T
     12(3,3),T3(3,3),V12(1,3),XLV12(1,1),V13(1,3),XII(1,3),XKK1(1,3),XLV
     113(1,1),XKK(1,3),XJJ(1,3),E1(2,3),XX2(1,1),XX3(1,1),YY3(1,1),A(1,1
     1),C1(2,3),C2(2,3),C3(2,3),INDX(4,3),KI1(3,3),KI2(3,3),KI3(3,3)/MAT
     1OUT/G11,G12,G13,G22,G23,G33,RHO,ALPHA1,ALPHA2,ALP12,TSUB0,GSUBE,SI
     1GTEN,SIGCOM,SIGSHE
      EQUIVALENCE (ECPT,NECPT)
      EQUIVALENCE (X1,XYZ1(1,1)),(X2,XYZ2(1,1)),(X3,XYZ3(1,1))
      EQUIVALENCE (K11(1,1),KI1(1,1)),(K12(1,1),KI2(1,1)),(K13(1,1),KI3(
     11,1)),(K21(1,1),KI1(1,1)),(K22(1,1),KI2(1,1)),(K23(1,1),KI3(1,1)),
     1(K31(1,1),KI1(1,1)),(K32(1,1),KI2(1,1)),(K33(1,1),KI3(1,1))
      DATA TMPOR1/36*0.0D0/
      PI(1,1)=3.14159265
      INFLAG=2
      DO 1 IP=1,3
      IF(NGRID(IP).EQ.NPVT)GO TO 2
    1 CONTINUE
      CALL MESAGE(-30,34,ECPT(1))
    2 CONTINUE
      MATID=MATID1
      ELTEMP=DUMV(1)
      SINTH=DSIN(PI(1,1)/180.*TH)
      COSTH=DCOS(PI(1,1)/180.*TH)
      CALL MAT(NECPT(1))
      G(1,1)=G11
      G(1,2)=G12
      G(2,1)=G12
      G(1,3)=G13
      G(3,1)=G13
      G(2,2)=G22
      G(2,3)=G23
      G(3,2)=G23
      G(3,3)=G33
      CALL TRANSD(ID1,T1)
      CALL TRANSD(ID2,T2)
      CALL TRANSD(ID3,T3)
      V12(1,1)=X2-X1
      V12(1,2)=Y2-Y1
      V12(1,3)=Z2-Z1
```

Figure 15 - Stiffness Matrix Subroutine, Sample Problem 1

```
      XLV12(1,1)=DSQRT(V12(1,1)**2+V12(1,2)**2+V12(1,3)**2)
      V13(1,1)=X3-X1
      V13(1,2)=Y3-Y1
      V13(1,3)=Z3-Z1
      XII(1,1)=V12(1,1)/XLV12(1,1)
      XII(1,2)=V12(1,2)/XLV12(1,1)
      XII(1,3)=V12(1,3)/XLV12(1,1)
      XKK1(1,1)=XII(1,2)*V13(1,3)-XII(1,3)*V13(1,2)
      XKK1(1,2)=XII(1,3)*V13(1,1)-XII(1,1)*V13(1,3)
      XKK1(1,3)=XII(1,1)*V13(1,2)-XII(1,2)*V13(1,1)
      XLV13(1,1)=DSQRT(XKK1(1,1)**2+XKK1(1,2)**2+XKK1(1,3)**2)
      XKK(1,1)=XKK1(1,1)/XLV13(1,1)
      XKK(1,2)=XKK1(1,2)/XLV13(1,1)
      XKK(1,3)=XKK1(1,3)/XLV13(1,1)
      XJJ(1,1)=XKK(1,2)*XII(1,3)-XII(1,2)*XKK(1,3)
      XJJ(1,2)=XKK(1,3)*XII(1,1)-XII(1,3)*XKK(1,1)
      XJJ(1,3)=XKK(1,1)*XII(1,2)-XII(1,1)*XKK(1,2)
      E1(1,1)=XII(1,1)
      E1(1,2)=XII(1,2)
      E1(1,3)=XII(1,3)
      E1(2,1)=XJJ(1,1)
      E1(2,2)=XJJ(1,2)
      E1(2,3)=XJJ(1,3)
      XX2(1,1)=XLV12(1,1)
      CALL GMMATD(V13,3,1,1,XII,3,1,0,XX3)
      YY3(1,1)=XLV13(1,1)
      A(1,1)=.5*XX2(1,1)*YY3(1,1)
      C1(1,1)=-1./XX2(1,1)
      C1(1,2)=0.0
      C1(1,3)=1./YY3(1,1)*(XX3(1,1)/XX2(1,1)-1.)
      C1(2,1)=0.0
      C1(2,2)=C1(1,3)
      C1(2,3)=C1(1,1)
      C2(1,1)=-C1(1,1)
      C2(1,2)=0.0
      C2(1,3)=-XX3(1,1)/(XX2(1,1)*YY3(1,1))
      C2(2,1)=0.0
      C2(2,2)=C2(1,3)
      C2(2,3)=1./XX2(1,1)
      C3(1,1)=0.0
      C3(1,2)=0.0
      C3(1,3)=1./YY3(1,1)
      C3(2,1)=0.0
      C3(2,2)=C3(1,3)
      C3(2,3)=0.0
      GO TO(100,200,300),IP
C     GENERATE THE MAIN VARIABLE
```

Figure 15 - Stiffness Matrix Subroutine, Sample Problem 1—Continued

```
  100 CONTINUE
      CALL GMMATD(C1,3,2,0,E1,3,2,1,Q1)
      CALL GMMATD(Q1,3,3,0,T1,3,3,0,Q2)
      CALL GMMATD(Q2,3,3,1,G,3,3,0,TEMPOR)
      CALL GMMATD(C1,3,2,0,E1,3,2,1,Q1)
      CALL GMMATD(Q1,3,3,0,T1,3,3,0,Q2)
      CALL GMMATD(TEMPOR,3,3,0,Q2,3,3,0,K11)
      CALL GMMATD(C2,3,2,0,E1,3,2,1,Q1)
      CALL GMMATD(Q1,3,3,0,T2,3,3,0,Q2)
      CALL GMMATD(TEMPOR,3,3,0,Q2,3,3,0,K12)
      CALL GMMATD(C3,3,2,0,E1,3,2,1,Q1)
      CALL GMMATD(Q1,3,3,0,T3,3,3,0,Q2)
      CALL GMMATD(TEMPOR,3,3,0,Q2,3,3,0,K13)
      GO TO 400
  200 CONTINUE
      CALL GMMATD(C2,3,2,0,E1,3,2,1,Q1)
      CALL GMMATD(Q1,3,3,0,T2,3,3,0,Q2)
      CALL GMMATD(Q2,3,3,1,G,3,3,0,TEMPOR)
      CALL GMMATD(C1,3,2,0,E1,3,2,1,Q1)
      CALL GMMATD(Q1,3,3,0,T1,3,3,0,Q2)
      CALL GMMATD(TEMPOR,3,3,0,Q2,3,3,0,K21)
      CALL GMMATD(C2,3,2,0,E1,3,2,1,Q1)
      CALL GMMATD(Q1,3,3,0,T2,3,3,0,Q2)
      CALL GMMATD(TEMPOR,3,3,0,Q2,3,3,0,K22)
      CALL GMMATD(C3,3,2,0,E1,3,2,1,Q1)
      CALL GMMATD(Q1,3,3,0,T3,3,3,0,Q2)
      CALL GMMATD(TEMPOR,3,3,0,Q2,3,3,0,K23)
      GO TO 400
  300 CONTINUE
      CALL GMMATD(C3,3,2,0,E1,3,2,1,Q1)
      CALL GMMATD(Q1,3,3,0,T3,3,3,0,Q2)
      CALL GMMATD(Q2,3,3,1,G,3,3,0,TEMPOR)
      CALL GMMATD(C1,3,2,0,E1,3,2,1,Q1)
      CALL GMMATD(Q1,3,3,0,T1,3,3,0,Q2)
      CALL GMMATD(TEMPOR,3,3,0,Q2,3,3,0,K31)
      CALL GMMATD(C2,3,2,0,E1,3,2,1,Q1)
      CALL GMMATD(Q1,3,3,0,T2,3,3,0,Q2)
      CALL GMMATD(TEMPOR,3,3,0,Q2,3,3,0,K32)
      CALL GMMATD(C3,3,2,0,E1,3,2,1,Q1)
      CALL GMMATD(Q1,3,3,0,T3,3,3,0,Q2)
      CALL GMMATD(TEMPOR,3,3,0,Q2,3,3,0,K33)
  400 GO TO(500,600,700),IP
C     GENERATE THE MAIN EXPRESSION
  500 CONTINUE
      TEMPOR(1)=A(1,1)*T
      DO 501 I=1,3
      DO 501 J=1,3
```

Figure 15 - Stiffness Matrix Subroutine, Sample Problem 1—Continued

```
  501 K11(I,J)=TEMPOR(1)*K11(I,J)
      DO 502 I=1,3
      DO 502 J=1,3
  502 K12(I,J)=TEMPOR(1)*K12(I,J)
      DO 503 I=1,3
      DO 503 J=1,3
  503 K13(I,J)=TEMPOR(1)*K13(I,J)
      GO TO 800
  600 CONTINUE
      TEMPOR(1)=A(1,1)*T
      DO 601 I=1,3
      DO 601 J=1,3
  601 K21(I,J)=TEMPOR(1)*K21(I,J)
      DO 602 I=1,3
      DO 602 J=1,3
  602 K22(I,J)=TEMPOR(1)*K22(I,J)
      DO 603 I=1,3
      DO 603 J=1,3
  603 K23(I,J)=TEMPOR(1)*K23(I,J)
      GO TO 800
  700 CONTINUE
      TEMPOR(1)=A(1,1)*T
      DO 701 I=1,3
      DO 701 J=1,3
  701 K31(I,J)=TEMPOR(1)*K31(I,J)
      DO 702 I=1,3
      DO 702 J=1,3
  702 K32(I,J)=TEMPOR(1)*K32(I,J)
      DO 703 I=1,3
      DO 703 J=1,3
  703 K33(I,J)=TEMPOR(1)*K33(I,J)
  800 CONTINUE
C     INSERT STIFFNESS PARTITION
      DO 801 I=1,3
      TMPOR1(I)=KI1(I,1)
      TMPOR1(I+6)=KI1(I,2)
  801 TMPOR1(I+12)=KI1(I,3)
      CALL SMA1B(TMPOR1,NGRID(1),-1,IFKGG,0.00)
      IF(IOPT4.EQ.0.OR.GSUBE.EQ.0.)GO TO 802
      TEMPOR(1)=GSUBE
      CALL SMA1B(TMPOR1,NGRID(1),-1,IF4GG,TEMPOR)
      K4GGSW=1
  802 CONTINUE
      DO 803 I=1,3
      TMPOR1(I)=KI2(I,1)
      TMPOR1(I+6)=KI2(I,2)
```

Figure 15 - Stiffness Matrix Subroutine, Sample Problem 1—Continued

```
803 TMPOR1(I+12)=KI2(I,3)
    CALL SMA1B(TMPOR1,NGRID(2),-1,IFKGG,0.D0)
    IF(IOPT4.EQ.0.OR.GSUBE.EQ.0.)GO TO 804
    TEMPOR(1)=GSUBE
    CALL SMA1B(TMPOR1,NGRID(2),-1,IF4GG,TEMPOR)
    K4GGSW=1
804 CONTINUE
    DO 805 I=1,3
    TMPOR1(I)=KI3(I,1)
    TMPOR1(I+6)=KI3(I,2)
805 TMPOR1(I+12)=KI3(I,3)
    CALL SMA1B(TMPOR1,NGRID(3),-1,IFKGG,0.D0)
    IF(IOPT4.EQ.0.OR.GSUBE.EQ.0.)GO TO 806
    TEMPOR(1)=GSUBE
    CALL SMA1B(TMPOR1,NGRID(3),-1,IF4GG,TEMPOR)
    K4GGSW=1
806 CONTINUE
    RETURN
    END
```

Figure 15 - Stiffness Matrix Subroutine, Sample Problem 1—Continued

```
BEGIN MASS
COMMENT  SUBPACKET A
2,M
COMMENT MASS MATRIX MATRIX EQUATION
MIJ
COMMENT  SUBPACKET B
COMMENT BEGIN LISTING OF MASS MATRIX PACKET VARIABLES
XMASS,,,
M11,3,3,TERM
M22,3,3,
M33,3,3,TERM
DEFINITIONS FINISHED
COMMENT  SUBPACKET C
COMMENT  PARTITIONS M12,M13,M21,M23,M31,AND M32 ARE IDENTICALLY ZERO
XMASS
1,1,A*(RHO*T+EMU)/3.
M11
1,1,XMASS(1,1)
2,2,XMASS(1,1)
3,3,XMASS(1,1)
M22
1,1,XMASS(1,1)
2,2,XMASS(1,1)
3,3,XMASS(1,1)
M33
1,1,XMASS(1,1)
2,2,XMASS(1,1)
3,3,XMASS(1,1)
END MASS PACKET
```

Figure 16 - Mass Matrix Packet, Sample Problem 1

MASS PACKET FORTRAN CODING

```
      SUBROUTINE MLEM40
      DOUBLE PRECISION Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8,Q9,PI,DETERM,TEMPOR,TMPOR
     11,XMASS,V12,XLV12,V13,XII,XKK1,XLV13,XX2,YY3,A,MI1,MI2,MI3,M11,M21
     1,M31,M12,M22,M32,M13,M23,M33
      DIMENSION ECPT(1)
      DIMENSION TMPOR1(36)
      DIMENSION XYZ1(1,3),XYZ2(1,3),XYZ3(1,3)
      DIMENSION M11(3,3),M12(3,3),M13(3,3),M21(3,3),M22(3,3),M23(3,3),M3
     11(3,3),M32(3,3),M33(3,3)
      COMMON /MATIN/MATID,INFLAG,ELTEMP,STRESS,SINTH,COSTH/SMA2IO/DUM1(1
     10),IFMGG,IGMGG,IFBGG,DUM2(23)/SMA2CL/IOPT4,BGGIND,NPVT,DUM3(157)/S
     1MA2ET/NECPT(1),NGRID(3),TH,MATID1,T,FMU,ID1,X1,Y1,Z1,ID2,X2,Y2,Z2,
     1ID3,X3,Y3,Z3,DUMV(80)/SMA2DP/I,J,IS,IP,I1,I2,PI(1,1),TEMPOR(9),DET
     1ERM,Q1(9),Q2(9),Q3(9),Q4(9),Q5(9),Q6(9),Q7(9),Q8(9),Q9(9),XMASS(1,
     11),V12(1,3),XLV12(1,1),V13(1,3),XII(1,3),XKK1(1,3),XLV13(1,1),XX2(
     11,1),YY3(1,1),A(1,1),INDX(4,3),MI1(3,3),MI2(3,3),MI3(3,3)/MATOUT/G
     111,G12,G13,G22,G23,G33,RHO,ALPHA1,ALPHA2,ALP12,TSUB0,GSUBE,SIGTEN,
     1SIGCOM,SIGSHE
      EQUIVALENCE (ECPT,NECPT)
      EQUIVALENCE (X1,XYZ1(1,1)),(X2,XYZ2(1,1)),(X3,XYZ3(1,1))
      EQUIVALENCE (M11(1,1),MI1(1,1)),(M12(1,1),MI2(1,1)),(M13(1,1),MI3(
     11,1)),(M21(1,1),MI1(1,1)),(M22(1,1),MI2(1,1)),(M23(1,1),MI3(1,1)),
     1(M31(1,1),MI1(1,1)),(M32(1,1),MI2(1,1)),(M33(1,1),MI3(1,1))
      DATA TMPOR1/36*0.0D0/
      PI(1,1)=3.14159265
      INFLAG=2
      DO 1 IP=1,3
      IF(NGRID(IP).EQ.NPVT)GO TO 2
    1 CONTINUE
      CALL MESAGE(-30,34,ECPT(1))
    2 CONTINUE
      CALL VLEM40
      MATID=MATID1
      ELTEMP=DUMV(1)
      SINTH=DSIN(PI(1,1)/180.*TH)
      COSTH=DCOS(PI(1,1)/180.*TH)
      CALL MAT(NECPT(1))
      V12(1,1)=X2-X1
      V12(1,2)=Y2-Y1
      V12(1,3)=Z2-Z1
      XLV12(1,1)=DSQRT(V12(1,1)**2+V12(1,2)**2+V12(1,3)**2)
      V13(1,1)=X3-X1
      V13(1,2)=Y3-Y1
      V13(1,3)=Z3-Z1
      XII(1,1)=V12(1,1)/XLV12(1,1)
      XII(1,2)=V12(1,2)/XLV12(1,1)
      XII(1,3)=V12(1,3)/XLV12(1,1)
      XKK1(1,1)=XII(1,2)*V13(1,3)-XII(1,3)*V13(1,2)
      XKK1(1,2)=XII(1,3)*V13(1,1)-XII(1,1)*V13(1,3)
      XKK1(1,3)=XII(1,1)*V13(1,2)-XII(1,2)*V13(1,1)
```

Figure 17 - Mass Matrix Subroutine, Sample Problem 1

```
      XLV13(1,1)=DSQRT(XKK1(1,1)**2+XKK1(1,2)**2+XKK1(1,3)**2)
      XX2(1,1)=XLV12(1,1)
      YY3(1,1)=XLV13(1,1)
      A(1,1)=.5*XX2(1,1)*YY3(1,1)
      XMASS(1,1)=A(1,1)*(RHO*T+FMU)/3.
      GO TO(100,200,300),IP
C     GENERATE THE MAIN VARIABLE
  100 CONTINUE
      M11(1,1)=XMASS(1,1)
      M11(1,2)=0.0
      M11(1,3)=0.0
      M11(2,1)=0.0
      M11(2,2)=XMASS(1,1)
      M11(2,3)=0.0
      M11(3,1)=0.0
      M11(3,2)=0.0
      M11(3,3)=XMASS(1,1)
      DO 101 I=1,3
      DO 101 J=1,3
  101 M12(I,J)=0.0
      DO 102 I=1,3
      DO 102 J=1,3
  102 M13(I,J)=0.0
      GO TO 400
  200 CONTINUE
      DO 201 I=1,3
      DO 201 J=1,3
  201 M21(I,J)=0.0
      M22(1,1)=XMASS(1,1)
      M22(1,2)=0.0
      M22(1,3)=0.0
      M22(2,1)=0.0
      M22(2,2)=XMASS(1,1)
      M22(2,3)=0.0
      M22(3,1)=0.0
      M22(3,2)=0.0
      M22(3,3)=XMASS(1,1)
      DO 202 I=1,3
      DO 202 J=1,3
  202 M23(I,J)=0.0
      GO TO 400
  300 CONTINUE
      DO 301 I=1,3
      DO 301 J=1,3
  301 M31(I,J)=0.0
      DO 302 I=1,3
      DO 302 J=1,3
```

Figure 17 - Mass Matrix Subroutine, Sample Problem 1—Continued

```
  302 M32(I,J)=0.0
      M33(1,1)=XMASS(1,1)
      M33(1,2)=0.0
      M33(1,3)=0.0
      M33(2,1)=0.0
      M33(2,2)=XMASS(1,1)
      M33(2,3)=0.0
      M33(3,1)=0.0
      M33(3,2)=0.0
      M33(3,3)=XMASS(1,1)
C     NO EXPRESSION-MAIN VARIABLE IS MAIN EXPRESSION
  400 CONTINUE
C     INSERT MASS PARTITION
      DO 401 I=1,3
      TMPOR1(I)=MI1(I,1)
      TMPOR1(I+6)=MI1(I,2)
  401 TMPOR1(I+12)=MI1(I,3)
      CALL SMA2B(TMPOR1,NGRID(1),-1,IFMGG,0.0D0)
      DO 402 I=1,3
      TMPOR1(I)=MI2(I,1)
      TMPOR1(I+6)=MI2(I,2)
  402 TMPOR1(I+12)=MI2(I,3)
      CALL SMA2B(TMPOR1,NGRID(2),-1,IFMGG,0.0D0)
      DO 403 I=1,3
      TMPOR1(I)=MI3(I,1)
      TMPOR1(I+6)=MI3(I,2)
  403 TMPOR1(I+12)=MI3(I,3)
      CALL SMA2B(TMPOR1,NGRID(3),-1,IFMGG,0.0D0)
      RETURN
      END
```

Figure 17 - Mass Matrix Subroutine, Sample Problem 1—Continued

```
BEGIN THERMAL LOADING
2,PPI
COMMENT THERMAL LOADING EQUATION
A*T*PPI*TBAR
ALPHV,3,1,TERM
TBAR,,,
PPI,3,,EQUA
DEFINITIONS FINISHED
ALPHV
1,1,ALPHA1
2,1,ALPHA2
3,1,ALP12
TBAR
1,1,(TTI(1)+TTI(2)+TTI(3))/3.-TSUB0
COMMENT  ALPHA1,ALPHA2,ALP12,TSUB0 COME FROM COMMON MATOUT
COMMENT  TTI IS THE VECTOR OF GRID POINT TEMPERATURES
PPI
TR(TI)*E1*TR(CI)*G*ALPHV
END THERMAL LOADING
```

Figure 18 - Thermal Loading Packet, Sample Problem 1

```
      SUBROUTINE M40(TTI,PG)
      REAL G,T1,T2,T3,V12,TBAR,ALPHV,XLV12,V13,XII,XKK1,XLV13,XKK,XJJ,E1
     11,XX2,XX3,YY3,A,C1,C2,C3,PP1,PP2,PP3
      DIMENSION ECPT(1),PG(1),TTI(1)
      DIMENSION TMPOR1(36)
      DIMENSION XYZ1(1,3),XYZ2(1,3),XYZ3(1,3)
      COMMON /MATIN/MATID,INFLAG,ELTEMP,STRESS,SINTH,COSTH/TRIMEX/NECPT(
     111),NGRID(3),TH,MATID1,T,FMU,ID1,X1,Y1,Z1,ID2,X2,Y2,Z2,ID3,X3,Y3,Z3
     11,DUMV(80)/EDTSP/I,J,IS,IP,I1,I2,PI(1,1),TEMPOR(9),DETERM,Q1(9),Q2(
     19),Q3(9),Q4(9),Q5(9),Q6(9),Q7(9),Q8(9),Q9(9),G(3,3),T1(3,3),T2(3,3
     11),T3(3,3),V12(1,3),TBAR(1,1),ALPHV(1,3),XLV12(1,1),V13(1,3),XII(1,
     113),XKK1(1,3),XLV13(1,1),XKK(1,3),XJJ(1,3),E1(2,3),XX2(1,1),XX3(1,1
     111),YY3(1,1),A(1,1),C1(2,3),C2(2,3),C3(2,3),INDX(4,3),PP1(1,3),PP2(1
     111,3),PP3(1,3)/MATOUT/G11,G12,G13,G22,G23,G33,RHO,ALPHA1,ALPHA2,ALP1
     112,TSUB0,GSUBE,SIGTEN,SIGCOM,SIGSHE
      EQUIVALENCE (ECPT,NECPT)
      EQUIVALENCE (X1,XYZ1(1,1)),(X2,XYZ2(1,1)),(X3,XYZ3(1,1))
      DATA TMPOR1/36*0.0/
      PI(1,1)=3.14159265
      INFLAG=2
      MATID=MATID1
      ELTEMP=DUMV(1)
      SINTH=SIN(PI(1,1)/180.*TH)
      COSTH=COS(PI(1,1)/180.*TH)
      CALL MAT(NECPT(1))
      G(1,1)=G11
      G(1,2)=G12
      G(2,1)=G12
      G(1,3)=G13
      G(3,1)=G13
      G(2,2)=G22
      G(2,3)=G23
      G(3,2)=G23
      G(3,3)=G33
      CALL TRAN(ID1,T1)
      CALL TRAN(ID2,T2)
      CALL TRAN(ID3,T3)
      V12(1,1)=X2-X1
      V12(1,2)=Y2-Y1
      V12(1,3)=Z2-Z1
C     FOLLOWING CARD CHANGED TO REFLECT PACKET PRECISION
      XLV12(1,1)=SQRT(V12(1,1)**2+V12(1,2)**2+V12(1,3)**2)
      V13(1,1)=X3-X1
      V13(1,2)=Y3-Y1
      V13(1,3)=Z3-Z1
      XII(1,1)=V12(1,1)/XLV12(1,1)
      XII(1,2)=V12(1,2)/XLV12(1,1)
      XII(1,3)=V12(1,3)/XLV12(1,1)
```

Figure 19 - Thermal Loading Subroutine, Sample Problem 1

```
      XKK1(1,1)=XII(1,2)*V13(1,3)-XII(1,3)*V13(1,2)
      XKK1(1,2)=XII(1,3)*V13(1,1)-XII(1,1)*V13(1,3)
      XKK1(1,3)=XII(1,1)*V13(1,2)-XII(1,2)*V13(1,1)
C     FOLLOWING CARD CHANGED TO REFLECT PACKET PRECISION
      XLV13(1,1)=SQRT(XKK1(1,1)**2+XKK1(1,2)**2+XKK1(1,3)**2)
      XKK(1,1)=XKK1(1,1)/XLV13(1,1)
      XKK(1,2)=XKK1(1,2)/XLV13(1,1)
      XKK(1,3)=XKK1(1,3)/XLV13(1,1)
      XJJ(1,1)=XKK(1,2)*XII(1,3)-XII(1,2)*XKK(1,3)
      XJJ(1,2)=XKK(1,3)*XII(1,1)-XII(1,3)*XKK(1,1)
      XJJ(1,3)=XKK(1,1)*XII(1,2)-XII(1,1)*XKK(1,2)
      E1(1,1)=XII(1,1)
      E1(1,2)=XII(1,2)
      E1(1,3)=XII(1,3)
      E1(2,1)=XJJ(1,1)
      E1(2,2)=XJJ(1,2)
      E1(2,3)=XJJ(1,3)
      XX2(1,1)=XLV12(1,1)
      CALL GMMATS(V13,3,1,1,XII,3,1,0,XX3)
      YY3(1,1)=XLV13(1,1)
      A(1,1)=.5*XX2(1,1)*YY3(1,1)
      C1(1,1)=-1./XX2(1,1)
      C1(1,2)=0.0
      C1(1,3)=1./YY3(1,1)*(XX3(1,1)/XX2(1,1)-1.)
      C1(2,1)=0.0
      C1(2,2)=C1(1,3)
      C1(2,3)=C1(1,1)
      C2(1,1)=-C1(1,1)
      C2(1,2)=0.0
      C2(1,3)=-XX3(1,1)/(XX2(1,1)*YY3(1,1))
      C2(2,1)=0.0
      C2(2,2)=C2(1,3)
      C2(2,3)=1./XX2(1,1)
      C3(1,1)=0.0
      C3(1,2)=0.0
      C3(1,3)=1./YY3(1,1)
      C3(2,1)=0.0
      C3(2,2)=C3(1,3)
      C3(2,3)=0.0
      TBAR(1,1)=(TTI(1)+TTI(2)+TTI(3))/3.-TSUB0
      ALPHV(1,1)=ALPHA1
      ALPHV(1,2)=ALPHA2
      ALPHV(1,3)=ALP12
C     GENERATE THE MAIN VARIABLE
      CALL GMMATS(T1,3,3,1,E1,3,2,0,Q1)
      CALL GMMATS(Q1,3,2,0,C1,3,2,1,Q2)
      CALL GMMATS(Q2,3,3,0,G,3,3,0,Q3)
      CALL GMMATS(Q3,3,3,1,ALPHV,3,1,0,PP1)
      CALL GMMATS(T2,3,3,1,E1,3,2,0,Q1)
```

Figure 19 - Thermal Loading Subroutine, Sample Problem 1—Continued

```
      CALL GMMATS(Q1,3,2,0,C2,3,2,1,Q2)
      CALL GMMATS(Q2,3,3,0,G,3,3,0,Q3)
      CALL GMMATS(Q3,3,3,0,ALPHV,3,1,0,PP2)
      CALL GMMATS(T3,3,3,1,E1,3,2,0,Q1)
      CALL GMMATS(Q1,3,2,0,C3,3,2,1,Q2)
      CALL GMMATS(Q2,3,3,0,G,3,3,0,Q3)
      CALL GMMATS(Q3,3,3,0,ALPHV,3,1,0,PP3)
C     GENERATE THE MAIN EXPRESSION
      TEMPOR(1)=A(1,1)*T
      DO 1 I=1,3
    1 Q1(I)=TEMPOR(1)*PP1(1,I)
      DO 2 I=1,3
    2 PP1(1,I)=TBAR(1,1)*Q1(I)
      DO 3 I=1,3
    3 Q1(I)=TEMPOR(1)*PP2(1,I)
      DO 4 I=1,3
    4 PP2(1,I)=TBAR(1,1)*Q1(I)
      DO 5 I=1,3
    5 Q1(I)=TEMPOR(1)*PP3(1,I)
      DO 6 I=1,3
    6 PP3(1,I)=TBAR(1,1)*Q1(I)
C     INSERT THERMALLOADING PARTITION
      DO 7 I=1,3
    7 TMPOR1(I)=PP1(1,I)
      DO 8 I=1,6
      L=NGRID(1)+I-1
    8 PG(L)=PG(L)+TMPOR1(I)
      DO 9 I=1,3
    9 TMPOR1(I)=PP2(1,I)
      DO 10 I=1,6
      L=NGRID(2)+I-1
   10 PG(L)=PG(L)+TMPOR1(I)
      DO 11 I=1,3
   11 TMPOR1(I)=PP3(1,I)
      DO 12 I=1,6
      L=NGRID(3)+I-1
   12 PG(L)=PG(L)+TMPOR1(I)
      RETURN
      END
```

Figure 19 - Thermal Loading Subroutine, Sample Problem 1--Continued

```
BEGIN STRESS
2,35
COMMENT  35 WORDS WILL BE PASSED INTO THE STRESS AND FORCE
COMMENT  CALCULATION PACKET--THE 35 WORDS ARE
COMMENT  ELEMENT ID(1),GRID POINTS(3),TSUB0(1),SB1(3 X 3 =9),
COMMENT  SB2(3 X 3 =9),SB3(3 X 3 =9),AND ST(3 X 1 =3)
TSUB0,SB1,SB2,SB3,ST
SB1,3,3,EQUA
SB2,3,3,EQUA
SB3,3,3,EQUA
ALPHV,3,1,TERM
ST,3,1,EQUA
DEFINITIONS FINISHED
SB1
G*C1*TR(E1)*T1
SB2
G*C2*TR(E1)*T2
SB3
G*C3*TR(E1)*T3
ALPHV
1,1,ALPHA1
2,1,ALPHA2
3,1,ALPHA3
ST
-G*ALPHV
END STRESS
```

Figure 20 - Stress Matrix Packet, Sample Problem 1

STRESS PACKET FORTRAN CODING

```
      SUBROUTINE SEL401
      REAL G,T1,T2,T3,SB1,SB2,SB3,ST,ALPHV,V12,XLV12,V13,XII,XKK1,XLV13,
     1XKK,XJJ,E1,XX2,XX3,YY3,C1,C2,C3
      DIMENSION ECPT(1),GRID(1)
      DIMENSION XYZ1(1,3),XYZ2(1,3),XYZ3(1,3),SB1(3,3),SB2(3,3),SB3(3,3)
     1,ST(1,3)
      COMMON /MATIN/MATID,INFLAG,ELTEMP,STRESS,SINTH,COSTH/SDR2X5/NECPT(
     11),NGRID(3),TH,MATID1,T,FMU,ID1,X1,Y1,Z1,ID2,X2,Y2,Z2,ID3,X3,Y3,Z3
     1,DUMV(80),PH1OUT(300)/SDR2X6/I,J,IS,IP,I1,I2,PI(1,1),TEMPOR(9),DET
     1ERM,Q1(9),Q2(9),Q3(9),Q4(9),Q5(9),Q6(9),Q7(9),Q8(9),Q9(9),G(3,3),T
     11(3,3),T2(3,3),T3(3,3),ALPHV(1,3),V12(1,3),XLV12(1,1),V13(1,3),XII
     1(1,3),XKK1(1,3),XLV13(1,1),XKK(1,3),XJJ(1,3),E1(2,3),XX2(1,1),XX3(
     11,1),YY3(1,1),C1(2,3),C2(2,3),C3(2,3),INDX(4,3)/MATOUT/G11,G12,G13
     1,G22,G23,G33,RHO,ALPHA1,ALPHA2,ALP12,TSUB0,GSUBE,SIGTEN,SIGCOM,SIG
     1SHE
      EQUIVALENCE (ECPT,NECPT)
      EQUIVALENCE (GRID,NGRID)
      EQUIVALENCE (X1,XYZ1(1,1)),(X2,XYZ2(1,1)),(X3,XYZ3(1,1))
      EQUIVALENCE (PH1OUT(6),SB1(1,1)),(PH1OUT(15),SB2(1,1)),(PH1OUT(24)
     1,SB3(1,1)),(PH1OUT(33),ST(1,1))
      PH1OUT(1)=ECPT(1)
      DO 1 I=1,3
    1 PH1OUT(I+1)=GRID(I)
      PI(1,1)=3.14159265
      INFLAG=2
      MATID=MATID1
      ELTEMP=DUMV(1)
      SINTH=SIN(PI(1,1)/180.*TH)
      COSTH=COS(PI(1,1)/180.*TH)
      CALL MAT(NECPT(1))
      G(1,1)=G11
      G(1,2)=G12
      G(2,1)=G12
      G(1,3)=G13
      G(3,1)=G13
      G(2,2)=G22
      G(2,3)=G23
      G(3,2)=G23
      G(3,3)=G33
      PH1OUT(5)=TSUB0
      CALL TRANSS(ID1,T1)
      CALL TRANSS(ID2,T2)
      CALL TRANSS(ID3,T3)
      V12(1,1)=X2-X1
      V12(1,2)=Y2-Y1
      V12(1,3)=Z2-Z1
```

Figure 21 - Stress Matrix Subroutine, Sample Problem 1

```
C       FOLLOWING CARD CHANGED TO REFLECT PACKET PRECISION
        XLV12(1,1)=SQRT(V12(1,1)**2+V12(1,2)**2+V12(1,3)**2)
        V13(1,1)=X3-X1
        V13(1,2)=Y3-Y1
        V13(1,3)=Z3-Z1
        XII(1,1)=V12(1,1)/XLV12(1,1)
        XII(1,2)=V12(1,2)/XLV12(1,1)
        XII(1,3)=V12(1,3)/XLV12(1,1)
        XKK1(1,1)=XII(1,2)*V13(1,3)-XII(1,3)*V13(1,2)
        XKK1(1,2)=XII(1,3)*V13(1,1)-XII(1,1)*V13(1,3)
        XKK1(1,3)=XII(1,1)*V13(1,2)-XII(1,2)*V13(1,1)
C       FOLLOWING CARD CHANGED TO REFLECT PACKET PRECISION
        XLV13(1,1)=SQRT(XKK1(1,1)**2+XKK1( ,2)**2+XKK1(1,3)**2)
        XKK(1,1)=XKK1(1,1)/XLV13(1,1)
        XKK(1,2)=XKK1(1,2)/XLV13(1,1)
        XKK(1,3)=XKK1(1,3)/XLV13(1,1)
        XJJ(1,1)=XKK(1,2)*XII(1,3)-XII(1,2)*XKK(1,3)
        XJJ(1,2)=XKK(1,3)*XII(1,1)-XII(1,3)*XKK(1,1)
        XJJ(1,3)=XKK(1,1)*XII(1,2)-XII(1,1)*XKK(1,2)
        E1(1,1)=XII(1,1)
        E1(1,2)=XII(1,2)
        E1(1,3)=XII(1,3)
        E1(2,1)=XJJ(1,1)
        E1(2,2)=XJJ(1,2)
        E1(2,3)=XJJ(1,3)
        XX2(1,1)=XLV12(1,1)
        CALL GMMATS(V13,3,1,1,XII,3,1,0,XX3)
        YY3(1,1)=XLV13(1,1)
        C1(1,1)=-1./XX2(1,1)
        C1(1,2)=0.0
        C1(1,3)=1./YY3(1,1)*(XX3(1,1)/XX2(1,1)-1.)
        C1(2,1)=0.0
        C1(2,2)=C1(1,3)
        C1(2,3)=C1(1,1)
        C2(1,1)=-C1(1,1)
        C2(1,2)=0.0
        C2(1,3)=-XX3(1,1)/(XX2(1,1)*YY3(1,1))
        C2(2,1)=0.0
        C2(2,2)=C2(1,3)
        C2(2,3)=1./XX2(1,1)
        C3(1,1)=0.0
        C3(1,2)=0.0
        C3(1,3)=1./YY3(1,1)
        C3(2,1)=0.0
        C3(2,2)=C3(1,3)
        C3(2,3)=0.0
        ALPHV(1,1)=ALPHA1
        ALPHV(1,2)=ALPHA2
        ALPHV(1,3)=ALP12
```

Figure 21 - Stress Matrix Subroutine, Sample Problem 1—Continued

```
      CALL GMMATS(G,3,3,0,C1,3,2,0,Q1)
      CALL GMMATS(Q1,3,2,0,E1,3,2,1,Q2)
      CALL GMMATS(Q2,3,3,0,T1,3,3,0,SB1)
      CALL GMMATS(G,3,3,0,C2,3,2,0,Q1)
      CALL GMMATS(Q1,3,2,0,E1,3,2,1,Q2)
      CALL GMMATS(Q2,3,3,0,T2,3,3,0,SB2)
      CALL GMMATS(G,3,3,0,C3,3,2,0,Q1)
      CALL GMMATS(Q1,3,2,0,E1,3,2,1,Q2)
      CALL GMMATS(Q2,3,3,0,T3,3,3,0,SB3)
      CALL GMMATS(G,3,3,0,ALPHV,3,1,0,Q1)
      DO 2 I=1,3
    2 ST(1,I)=-Q1(I)
      RETURN
      END
```

Figure 21 - Stress Matrix Subroutine, Sample Problem 1—Continued

```
BEGIN STRESS AND FORCE
8,0,0,0
COMMENT  8 STRESS WORDS WILL BE OUTPUT  ---  THESE ARE
COMMENT  ELEMENT ID (1),SIG(3 X 1 =3),THETA(1),SIGP1(1),
COMMENT  SIGP2(1),AND TAU(1)
SIG,INSERT A,THETA,INSERT B,SIGP1,SIGP2,TAU
SIG,1,3;EQUA
SAV,1,1,DEFER
TAU,,,TERM
SIGP1,,,TERM
SIGP2,,,TERM
THETA,1,1,TERM
DEFINITIONS FINISHED
SIG
COMMENT  HERE DISPJ IS THE 3 X 1 TRANSLATION VECTOR FOR GRID POINT J
COMMENT  TEMP IS ELEMENT TEMPERATURE
SB1*DISP1+SB2*DISP2+SB3*DISP3+ST*(TEMP-TSUB0)
TAU
,,SQRT((SAV(1,1)/2.)**2+                                $
SIG(1,3)**2)
SIGP1
1,1,(SIG(1,1)+SIG(1,2))/2.+TAU(1,1)
SIGP2
1,1,(SIG(1,1)+SIG(1,2))/2.-TAU(1,1)
INSERT A
      SAV(1,1)=SIG(1,1)-SIG(2,1)
      IF(ABS(SAV(1,1)).LT.1.E-15.AND.ABS(2.*SIG(3,1)).LT.1.E-15)GO TO 90
     10
      IF(ABS(SAV(1,1)).LT.1.E-15)GO TO 9100
DEFINITIONS FINISHED
THETA
1,1,ATAN(2.*SIG(1,3)/SAV(1,1))*28.64789
INSERT B
      GO TO 9200
 9000 THETA(1,1)=0.
      GO TO 9200
 9100 THETA(1,1)=45.
 9200 CONTINUE
DEFINITIONS FINISHED
END STRESS AND FORCE
```

Figure 22 - Stress and Force Calculation Packet, Sample Problem 1

STRESSANDFORCE PACKET FORTRAN CODING

```
      SUBROUTINE SEL402
      REAL SIG,THETA,SIGP1,SIGP2,TAU,SAV
      DIMENSION NGRID(3),STRESS(8)
      DIMENSION XYZ1(1,3),XYZ2(1,3),XYZ3(1,3),SB1(3,3),SB2(3,3),SB3(3,3)
     1,ST(1,3)
      DIMENSION SIG(3,1),THETA(1,1),SIGP1(1,1),SIGP2(1,1),TAU(1,1)
      COMMON/SDR2XX/Z(1)/SDR2X4/DUM(35),IVEC,IVECN,TEMP,DEFORM/SDR2X7/PH
     11OUT(200),FORVEC(100)/SDR2X8/I,J,IS,IP,I1,I2,PI(1,1),TEMPOR(9),DET
     1ERM,Q1(9),Q2(9),Q3(9),Q4(9),Q5(9),Q6(9),Q7(9),Q8(9),Q9(9),DISP1(1,
     13),DISP2(1,3),DISP3(1,3),SAV(1,1),INDX(4,3)
      EQUIVALENCE (NPH1OU,PH1OUT),(NGRID(1),PH1OUT(2)),(STRESS(1),PH1OUT
     1(101))
      EQUIVALENCE (X1,XYZ1(1,1)),(X2,XYZ2(1,1)),(X3,XYZ3(1,1))
      EQUIVALENCE (PH1OUT(6),SB1(1,1)),(PH1OUT(15),SB2(1,1)),(PH1OUT(24)
     1,SB3(1,1)),(PH1OUT(33),ST(1,1))
      EQUIVALENCE (PH1OUT(102),SIG(1,1)),(PH1OUT(105),THETA(1,1)),(PH1OU
     1T(106),SIGP1(1,1)),(PH1OUT(107),SIGP2(1,1)),(PH1OUT(108),TAU(1,1))
      EQUIVALENCE (PH1OUT(5),TSUB0)
      PH1OUT(101)=PH1OUT(1)
      PH1OUT(201)=PH1OUT(1)
      IP=IVEC+NGRID(1)-1
      J=IP+2
      DO 1 I=IP,J
      IS=I-IP+1
    1 DISP1(1,IS)=Z(I)
      IP=IVEC+NGRID(2)-1
      J=IP+2
      DO 2 I=IP,J
      IS=I-IP+1
    2 DISP2(1,IS)=Z(I)
      IP=IVEC+NGRID(3)-1
      J=IP+2
      DO 3 I=IP,J
      IS=I-IP+1
    3 DISP3(1,IS)=Z(I)
      PI(1,1)=3.14159265
      CALL GMMATS(SB1,3,3,0,DISP1,3,1,0,Q1)
      CALL GMMATS(SB2,3,3,0,DISP2,3,1,0,Q2)
      DO 4 I=1,3
    4 Q3(I)=Q1(I)+Q2(I)
      CALL GMMATS(SB3,3,3,0,DISP3,3,1,0,Q4)
      DO 5 I=1,3
    5 Q5(I)=Q3(I)+Q4(I)
      Q6(1)=TEMP-TSUB0
      DO 6 I=1,3
    6 Q7(I)=Q6(1)*ST(1,I)
      DO 7 I=1,3
```

Figure 23 - Stress and Force Calculation Subroutine, Sample Problem 1

```
    7 SIG(I,1)=Q5(I)+Q7(I)
      SAV(1,1)=SIG(1,1)-SIG(2,1)
      IF(ABS(SAV(1,1)).LT.1.E-15.AND.ABS(2.*SIG(3,1)).LT.1.E-15)GO TO 90
     100
      IF(ABS(SAV(1,1)).LT.1.E-15)GO TO 9100
      THETA(1,1)=ATAN(2.*SIG(3,1)/SAV(1,1))*28.64789
      GO TO 9200
 9000 THETA(1,1)=0.
      GO TO 9200
 9100 THETA(1,1)=45.
 9200 CONTINUE
      SIGP1(1,1)=(SIG(1,1)+SIG(2,1))/2.+TAU(1,1)
      SIGP2(1,1)=(SIG(1,1)+SIG(2,1))/2.-TAU(1,1)
      TAU(1,1)=SQRT((SAV(1,1)/2.)**2+SIG(3,1)**2)
      RETURN
      END
```

Figure 23 - Stress and Force Calculation Subroutine, Sample Problem 1—Continued

```
BEGIN OUTPUT
STRESS

COMMENT  WE ARE DEFINING FORMATS FOR SORT1,REAL OUTPUT
COMMENT  NEW HEADING FORMATS ARE 240,220,AND 225
115C,240,0,220,-1,225
COMMENT  THE FOLLOWING ARE THE I VALUES FOR THE FORMAT PIECES
1,74,-4,30,-33,39,-4,30,-33,30,-4,39,-33,39,   $
-33,30,0
COMMENT  WE WILL NOW SPECIFY THE NEW FORMAT HEADINGS
240
(40X,51HE L E M E N T  S T R E S S E S  F O R  E L E M 4 0 )
220
(1X,7HELEMENT)
225
(3X,3HID.,9X,6HSIG(1),10X,6HSIG(2),10X,6HSIG(3),11X,5HTHETA,$
11X,5HSIGP1,11X,5HSIGP2,12X,3HTAU)
COMMENT  WE WILL NOW SPECIFY THE NEW FORMAT PIECES
74
1X,I7
30
1PE11.4,0PF8.1,3X
-33
5X
39
1PE11.4,0PF8.1,3X
DEFINITIONS FINISHED
COMMENT  ACCORDING TO THE STRESS AND FORCE CALCULATION PACKET
COMMENT  8 VALUES WILL BE OUTPUT, THE FIRST BEING THE ELEMNT ID ---
COMMENT  COMBINING THE SPECIFIED FORMAT PIECES,ADDING IN THE APPROPRIATE
COMMENT  COMMAS,AND THE BEGINNING AND ENDING PARENTHESES(WHICH NASTRAN
COMMENT  DOES) WE GET
COMMENT  (1X,I7,5X,1PE11.4,5X,1PE11.4,5X,1PE11.4,5X,1PE11.4,5X,1PE11.4,
COMMENT  5X,1PE11.4,5X,1PE11.4)
END OUTPUT
INPUT FINISHED
```

Figure 24 - Output Packet, Sample Problem 1

OUTPUT PACKET

```
      BLOCK DATA
      COMMON/OFPCOM/NOELEM,IPOINT(24),IOFP2(144),IOFP1(250),IOFP5(300),I
     1PTFP1(24)/HEDCOM/ISTNO(3,120),IFRMTS(5000),IFMT(180)
      DATA NOELEM/38/
      DATA IPOINT,IOFP2,IOFP1,IOFP5,IPTFP1,ISTNO,IFRMTS,IFMT/1102*0,5180
     1*10H          /
      DATA IOFP2(19),IOFP2(20),IOFP2(21),IOFP2(22),IOFP2(23),IOFP2(24)/1
     1150,240,0,220,-1,225/
      DATA IPOINT(4)/571/
      DATA ISTNO(1,16),ISTNO(2,16),ISTNO(3,16)/240,6,334/
      DATA IFRMTS(334)/10H(40X,51HE /,IFRMTS(335)/10HL E M E N /,IFRMTS(
     1336)/10HT  S T R E/,IFRMTS(337)/10H S S E S  /,IFRMTS(338)/10HF O
     1R  E L/,IFRMTS(339)/10H E M 4 0 )/
      DATA ISTNO(1,17)/0/
      DATA ISTNO(1,18),ISTNO(2,18),ISTNO(3,18)/220,2,340/
      DATA IFRMTS(340)/10H(1X,7HELEM/,IFRMTS(341)/10HENT)      /
      DATA ISTNO(1,19)/-1/
      DATA ISTNO(1,20),ISTNO(2,20),ISTNO(3,20)/225,10,342/
      DATA IFRMTS(342)/10H(3X,3HID.,/,IFRMTS(343)/10H9X,6HSIG(1/,IFRMTS(
     1344)/10H),10X,6HSI/,IFRMTS(345)/10HG(2),10X,6/,IFRMTS(346)/10HHSIG
     1(3),11/,IFRMTS(347)/10HX,5HTHETA,/,IFRMTS(348)/10H11X,5HSIGP/,IFRM
     1TS(349)/10H1,11X,5HSI/,IFRMTS(350)/10HGP2,12X,3H/,IFRMTS(351)/10HT
     1AU)      /
      DATA IPTFP1(4)/1/
      DATA IOFP1(1)/1/
      DATA IOFP1(2),IOFP5(1),IOFP5(2),IOFP5(3),IOFP5(4),IOFP5(5),IOFP5(6
     1)/2*74,10H1X,I7     ,10H          ,10H          ,10H          ,10H
     1          /
      DATA IOFP1(3)/-4/
      DATA IOFP1(4),IOFP5(7),IOFP5(8),IOFP5(9),IOFP5(10),IOFP5(11),IOFP5
     1(12)/2*30,10H          ,10H   1PE11.4,10HOPF8.1,3X ,10H          ,
     110H          /
      DATA IOFP1(5),IOFP5(13),IOFP5(14),IOFP5(15),IOFP5(16),IOFP5(17),IO
     1FP5(18)/2*-33,10H5X        ,10H          ,10H          ,10H
     1    ,10H          /
      DATA IOFP1(6),IOFP5(19),IOFP5(20),IOFP5(21),IOFP5(22),IOFP5(23),IO
     1FP5(24)/2*39,10H          ,10H   1PE11.4,10HOPF8.1,3X ,10H
     1   ,10H          /
      DATA IOFP1(7)/-4/
      DATA IOFP1(8)/30/
      DATA IOFP1(9)/-33/
      DATA IOFP1(10)/30/
      DATA IOFP1(11)/-4/
      DATA IOFP1(12)/39/
      DATA IOFP1(13)/-33/
      DATA IOFP1(14)/39/
      DATA IOFP1(15)/-33/
      DATA IOFP1(16)/30/
      DATA IOFP1(17)/0/
      END
```

Figure 25 - Output Packet BLOCK DATA Subprogram, Sample Problem 1

```
PARAM = 1
PACKET VARIABLES = 90
ELEM50,8,1,0,38,,3,0,121
JD1,ANGLE,JD2
BEGIN STIFFNESS
2,KIJ
KIJ
INSERT COMMON A
INSERT EQUIVALENCE A
INSERT A
XX,8,3,EQUA
INSERT B
PT,2,1,TERM
DKX,1,1,TERM
DKY,1,1,TERM
DKZ,1,1,TERM
DN,1,3,TERM
DNA,3,8,TERM
DNB,3,8,TERM
DNC,3,8,TERM
DND,3,8,TERM
DNE,3,8,TERM
DNF,3,8,TERM
DNG,3,8,TERM
DNH,3,8,TERM
KX,1,1,COMM
KY,1,1,COMM
KZ,1,1,COMM
DNLA,3,8,EQUA
DETA,,,EQUA
DNLB,3,8,EQUA
DETB,,,EQUA
DNLC,3,8,EQUA
DETC,,,EQUA
DNLD,3,8,EQUA
DETD,,,EQUA
DNLE,3,8,EQUA
DETE,,,EQUA
DNLF,3,8,EQUA
DETF,,,EQUA
DNLG,3,8,EQUA
DETG,,,EQUA
DNLH,3,8,EQUA
DETH,,,EQUA
BTA,1,3,TERM
BTB,1,3,TERM
BTC,1,3,TERM
BTD,1,3,TERM
BTE,1,3,TERM
BTF,1,3,TERM
BTG,1,3,TERM
BTH,1,3,TERM
SAVEA,1,8,EQUA
```

Figure 26 - Preliminary Data Packet and Stiffness Matrix Packet, Sample Problem 2

```
SAVEB,1,8,EQUA
SAVEC,1,8,EQUA
SAVEI,1,8,EQUA
SAVEE,1,8,EQUA
SAVEF,1,8,EQUA
SAVEG,1,8,EQUA
SAVEH,1,8,EQUA
SUM1,,,TERM
SUM2,,,TERM
SUM3,,,TERM
SUM4,,,TERM
SUM5,,,TERM
SUM6,,,TERM
SUM7,,,TERM
SUM8,,,TERM
KIJ,,,EQUA
DEFINITIONS FINISHED
INSERT COMMON A
      DIMENSION IZ(1)
      DIMENSION TA(9),TOFF(3)
      DIMENSION XX1(3),XX2(3),XX3(3),XX4(3),XX5(3),XX6(3)
      DIMENSION XX7(3),XX8(3)
      COMMON /SYSTEM/DUMM(36),ISOP
      COMMON /SMA1X/Z(1)
      COMMON /SMA1BK/ICSTM,NCSTM
DEFINITIONS FINISHED
INSERT EQUIVALENCE A
      EQUIVALENCE (Z,IZ)
DEFINITIONS FINISHED
INSERT A
      IF(ISOP.EQ.-1)CALL MESAGE(-30,154,ECPT(1))
      IF(JD1.EQ.0)GO TO 1065
      IF(NCSTM.EQ.0)GO TO 1010
      DO 1020 I=1,NCSTM,14
      I1=ICSTM+I
      IF(JD1.NE.IZ(I1))GO TO 1000
      IF(IZ(I+1).EQ.1)GO TO 1030
      GO TO 1020
 1000 CONTINUE
 1010 CALL MESAGE(-30,25,ID1)
 1020 CALL MESAGE(-30,155,ID1)
 1030 DO 1040 J=1,9
      I1=I+4+J
 1040 TA(J)=Z(I1)
      DO 1050 J=1,3
      I1=I+1+J
 1050 TOFF(J)=Z(I1)
      DO 1060 I=1,3
      XX1(I)=XYZ1(1,I)-TOFF(I)
      XX2(I)=XYZ2(1,I)-TOFF(I)
      XX3(I)=XYZ3(1,I)-TOFF(I)
      XX4(I)=XYZ4(1,I)-TOFF(I)
```

Figure 26 - Preliminary Data Packet and Stiffness Matrix Packet, Sample Problem 2—Continued

```
      XX5(I)=XYZ5(1,I)-TOFF(I)
      XX6(I)=XYZ6(1,I)-TOFF(I)
      XX7(I)=XYZ7(1,I)-TOFF(I)
 1060 XX8(I)=XYZ8(1,I)-TOFF(I)
      CALL GMMATD(TA,3,3,1,XX1,3,1,0,XX(1,1))
      CALL GMMATD(TA,3,3,1,XX2,3,1,0,XX(1,2))
      CALL GMMATD(TA,3,3,1,XX3,3,1,0,XX(1,3))
      CALL GMMATD(TA,3,3,1,XX4,3,1,0,XX(1,4))
      CALL GMMATD(TA,3,3,1,XX5,3,1,0,XX(1,5))
      CALL GMMATD(TA,3,3,1,XX6,3,1,0,XX(1,6))
      CALL GMMATD(TA,3,3,1,XX7,3,1,0,XX(1,7))
      CALL GMMATD(TA,3,3,1,XX8,3,1,0,XX(1,8))
      GO TO 1070
 1065 CONTINUE
DEFINITIONS FINISHED
XX
ROW 1,XYZ1
ROW 2,XYZ2
ROW 3,XYZ3
ROW 4,XYZ4
ROW 5,XYZ5
ROW 6,XYZ6
ROW 7,XYZ7
ROW 8,XYZ8
INSERT B
 1070 CONTINUE
DEFINITIONS FINISHED
PT
1,1,-0.577350270
2,1,-PT(1,1)
DKX
1,1,1.0
DKY
1,1,1.0
DKZ
1,1,1.0
DN
1,1,0.125D0*(1.D0+PT(1,1))**2
1,2,0.125D0*(1.D0-PT(1,1))**2
1,3,0.125D0*(1.D0+PT(1,1))*(1.D0-PT(1,1))
DNA
1,1,-DN(1,2)
2,1,-DN(1,2)
3,1,-DN(1,2)
1,2,+DN(1,2)
2,2,-DN(1,3)
3,2,-DN(1,3)
1,3,+DN(1,3)
2,3,-DN(1,1)
3,3,+DN(1,3)
1,4,-DN(1,3)
2,4,-DN(1,3)
3,4,+DN(1,2)
```

Figure 26 - Preliminary Data Packet and Stiffness Matrix Packet, Sample Problem 2—Continued

```
1,5,-DN(1,3)
2,5,+DN(1,2)
3,5,-DN(1,3)
1,6,+DN(1,3)
2,6,+DN(1,3)
3,6,-DN(1,1)
1,7,+DN(1,1)
2,7,+DN(1,1)
3,7,+DN(1,1)
1,8,-DN(1,1)
2,8,+DN(1,3)
3,8,+DN(1,3)
DNB
1,1,-DN(1,3)
2,1,-DN(1,3)
3,1,-DN(1,2)
1,2,+DN(1,3)
2,2,-DN(1,1)
3,2,-DN(1,3)
1,3,+DN(1,2)
2,3,-DN(1,3)
3,3,+DN(1,3)
1,4,-DN(1,2)
2,4,-DN(1,2)
3,4,+DN(1,2)
1,5,-DN(1,1)
2,5,+DN(1,3)
3,5,-DN(1,3)
1,6,+DN(1,1)
2,6,+DN(1,1)
3,6,-DN(1,1)
1,7,+DN(1,3)
2,7,+DN(1,3)
3,7,+DN(1,1)
1,8,-DN(1,3)
2,8,+DN(1,2)
3,8,+DN(1,3)
DNC
1,1,-DN(1,3)
2,1,-DN(1,2)
3,1,-DN(1,3)
1,2,+DN(1,3)
2,2,-DN(1,3)
3,2,-DN(1,1)
1,3,+DN(1,1)
2,3,-DN(1,1)
3,3,+DN(1,1)
1,4,-DN(1,1)
2,4,-DN(1,3)
3,4,+DN(1,3)
```

Figure 26 - Preliminary Data Packet and Stiffness Matrix Packet, Sample Problem 2—Continued

```
1,5,-DN(1,2)
2,5,+DN(1,2)
3,5,-DN(1,2)
1,6,+DN(1,2)
2,6,+DN(1,3)
3,6,-DN(1,3)
1,7,+DN(1,3)
2,7,+DN(1,1)
3,7,+DN(1,3)
1,8,-DN(1,3)
2,8,+DN(1,3)
3,8,+DN(1,2)
DND
1,1,-DN(1,1)
2,1,-DN(1,3)
3,1,-DN(1,3)
1,2,+DN(1,1)
2,2,-DN(1,1)
3,2,-DN(1,1)
1,3,+DN(1,3)
2,3,-DN(1,3)
3,3,+DN(1,1)
1,4,-DN(1,3)
2,4,-DN(1,2)
3,4,+DN(1,3)
1,5,-DN(1,3)
2,5,+DN(1,3)
3,5,-DN(1,2)
1,6,+DN(1,3)
2,6,+DN(1,1)
3,6,-DN(1,3)
1,7,+DN(1,2)
2,7,+DN(1,3)
3,7,+DN(1,3)
1,8,-DN(1,2)
2,8,+DN(1,2)
3,8,+DN(1,2)
DNE
1,1,-DN(1,2)
2,1,-DN(1,3)
3,1,-DN(1,3)
1,2,+DN(1,2)
2,2,-DN(1,2)
3,2,-DN(1,2)
1,3,+DN(1,3)
2,3,-DN(1,3)
3,3,+DN(1,2)
1,4,-DN(1,3)
2,4,-DN(1,1)
3,4,+DN(1,3)
```

Figure 26 - Preliminary Data Packet and Stiffness Matrix Packet, Sample Problem 2—Continued

```
1,5,-DN(1,3)
2,5,+DN(1,3)
3,5,-DN(1,1)
1,6,+DN(1,3)
2,6,+DN(1,2)
3,6,-DN(1,3)
1,7,+DN(1,1)
2,7,+DN(1,3)
3,7,+DN(1,3)
1,8,-DN(1,1)
2,8,+DN(1,1)
3,8,+DN(1,1)
DNF
1,1,-DN(1,3)
2,1,-DN(1,1)
3,1,-DN(1,3)
1,2,+DN(1,3)
2,2,-DN(1,3)
3,2,-DN(1,2)
1,3,+DN(1,2)
2,3,-DN(1,2)
3,3,+DN(1,2)
1,4,-DN(1,2)
2,4,-DN(1,3)
3,4,+DN(1,3)
1,5,-DN(1,1)
2,5,+DN(1,1)
3,5,-DN(1,1)
1,6,+DN(1,1)
2,6,+DN(1,3)
3,6,-DN(1,3)
1,7,+DN(1,3)
2,7,+DN(1,2)
3,7,+DN(1,3)
1,8,-DN(1,3)
2,8,+DN(1,3)
3,8,+DN(1,1)
DNG
1,1,-DN(1,3)
2,1,-DN(1,3)
3,1,-DN(1,1)
1,2,+DN(1,3)
2,2,-DN(1,2)
3,2,-DN(1,3)
1,3,+DN(1,1)
2,3,-DN(1,3)
3,3,+DN(1,3)
1,4,-DN(1,1)
2,4,-DN(1,1)
3,4,+DN(1,1)
1,5,-DN(1,2)
2,5,+DN(1,3)
3,5,-DN(1,3)
```

Figure 26 - Preliminary Data Packet and Stiffness Matrix Packet, Sample Problem 2—Continued

```
1,6,+DN(1,2)
2,6,+DN(1,2)
3,6,-DN(1,2)
1,7,+DN(1,3)
2,7,+DN(1,3)
3,7,+DN(1,2)
1,8,-DN(1,3)
2,8,+DN(1,1)
3,8,+DN(1,3)
DNH
1,1,-DN(1,1)
2,1,-DN(1,1)
3,1,-DN(1,1)
1,2,+DN(1,1)
2,2,-DN(1,3)
3,2,-DN(1,3)
1,3,+DN(1,3)
2,3,-DN(1,2)
3,3,+DN(1,3)
1,4,-DN(1,3)
2,4,-DN(1,3)
3,4,+DN(1,1)
1,5,-DN(1,3)
2,5,+DN(1,1)
3,5,-DN(1,3)
1,6,+DN(1,3)
2,6,+DN(1,3)
3,6,-DN(1,2)
1,7,+DN(1,2)
2,7,+DN(1,2)
3,7,+DN(1,2)
1,8,-DN(1,2)
2,8,+DN(1,3)
3,8,+DN(1,3)
DNLA
INV(DNA*XX)*DNA
DETA
DETERM
DNLB
INV(DNB*XX)*DNB
DETB
DETERM
DNLC
INV(DNC*XX)*DNC
DETC
DETERM
DNLD
INV(DND*XX)*DND
DETD
DETERM
DNLE
```

Figure 26 - Preliminary Data Packet and Stiffness Matrix Packet, Sample Problem 2—Continued

```
INV(DNE*XX)*DNE
DETE
DETERM
DNLF
INV(DNF*XX)*DNF
DETF
DETERM
DNLG
INV(DNG*XX)*DNG
DETG
DETERM
DNLH
INV(DNH*XX)*DNH
DETH
DETERM
BTA
1,1,DKX(1,1)*DNLA(1,IP)
1,2,DKY(1,1)*DNLA(2,IP)
1,3,DKZ(1,1)*DNLA(3,IP)
BTB
1,1,DKX(1,1)*DNLB(1,IP)
1,2,DKY(1,1)*DNLB(2,IP)
1,3,DKZ(1,1)*DNLB(3,IP)
BTC
1,1,DKX(1,1)*DNLC(1,IP)
1,2,DKY(1,1)*DNLC(2,IP)
1,3,DKZ(1,1)*DNLC(3,IP)
BTD
1,1,DKX(1,1)*DNLD(1,IP)
1,2,DKY(1,1)*DNLD(2,IP)
1,3,DKZ(1,1)*DNLD(3,IP)
BTE
1,1,DKX(1,1)*DNLE(1,IP)
1,2,DKY(1,1)*DNLE(2,IP)
1,3,DKZ(1,1)*DNLE(3,IP)
BTF
1,1,DKX(1,1)*DNLF(1,IP)
1,2,DKY(1,1)*DNLF(2,IP)
1,3,DKZ(1,1)*DNLF(3,IP)
BTG
1,1,DKX(1,1)*DNLG(1,IP)
1,2,DKY(1,1)*DNLG(2,IP)
1,3,DKZ(1,1)*DNLG(3,IP)
BTH
1,1,DKX(1,1)*DNLH(1,IP)
1,2,DKY(1,1)*DNLH(2,IP)
1,3,DKZ(1,1)*DNLH(3,IP)
SAVEA
BTA*DNLA*DETA
SAVEB
BTB*DNLB*DETB
SAVEC
```

Figure 26 - Preliminary Data Packet and Stiffness Matrix Packet, Sample Problem 2—Continued

```
BTC*DNLC*DETC
SAVEI
BTD*DNLD*DETD
SAVEE
BTE*DNLE*DETE
SAVEF
BTF*DNLF*DETF
SAVEG
BTG*DNLG*DETG
SAVEH
BTH*DNLH*DETH
SUM1
1,1,$
SAVEA(1,1)+SAVEB(1,1)+SAVEC(1,1)+SAVEI(1,1)+SAVEE(1,1)+SAVEF(1,1)+$
SAVEG(1,1)+SAVEH(1,1)
SUM2
1,1,$
SAVEA(1,2)+SAVEB(1,2)+SAVEC(1,2)+SAVEI(1,2)+SAVEE(1,2)+SAVEF(1,2)+$
SAVEG(1,2)+SAVEH(1,2)
SUM3
1,1,$
SAVEA(1,3)+SAVEB(1,3)+SAVEC(1,3)+SAVEI(1,3)+SAVEE(1,3)+SAVEF(1,3)+$
SAVEG(1,3)+SAVEH(1,3)
SUM4
1,1,$
SAVEA(1,4)+SAVEB(1,4)+SAVEC(1,4)+SAVEI(1,4)+SAVEE(1,4)+SAVEF(1,4)+$
SAVEG(1,4)+SAVEH(1,4)
SUM5
1,1,$
SAVEA(1,5)+SAVEB(1,5)+SAVEC(1,5)+SAVEI(1,5)+SAVEE(1,5)+SAVEF(1,5)+$
SAVEG(1,5)+SAVEH(1,5)
SUM6
1,1,$
SAVEA(1,6)+SAVEB(1,6)+SAVEC(1,6)+SAVEI(1,6)+SAVEE(1,6)+SAVEF(1,6)+$
SAVEG(1,6)+SAVEH(1,6)
SUM7
1,1,$
SAVEA(1,7)+SAVEB(1,7)+SAVEC(1,7)+SAVEI(1,7)+SAVEE(1,7)+SAVEF(1,7)+$
SAVEG(1,7)+SAVEH(1,7)
SUM8
1,1,$
SAVEA(1,8)+SAVEB(1,8)+SAVEC(1,8)+SAVEI(1,8)+SAVEE(1,8)+SAVEF(1,8)+$
SAVEG(1,8)+SAVEH(1,8)
KIJ
SUMJ
END STIFFNESS
INPUT FINISHED
```

Figure 26 - Preliminary Data Packet and Stiffness Matrix Packet, Sample Problem 2—Continued

```
      SUBROUTINE KLEM50
      DOUBLE PRECISION Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8,Q9,PI,DETERM,TEMPOR,TMPOR
     11,XX,PT,DKX,DKY,DKZ,DN,DNA,DNB,DNC,DND,DNE,DNF,DNG,DNH,DNLA,DETA,D
     1NLB,DETB,DNLC,DETC,DNLD,DETD,DNLE,DETE,DNLF,DETF,DNLG,DETG,DNLH,DE
     1TH,BTA,BTB,BTC,BTD,BTE,BTF,BTG,BTH,SAVEA,SAVEB,SAVEC,SAVEI,SAVEE,S
     1AVEF,SAVEG,SAVEH,SUM1,SUM2,SUM3,SUM4,SUM5,SUM6,SUM7,SUM8,KI1,KI2,K
     1I3,KI4,KI5,KI6,KI7,KI8,K11,K21,K31,K41,K51,K61,K71,K81,K12,K22,K32
     1,K42,K52,K62,K72,K82,K13,K23,K33,K43,K53,K63,K73,K83,K14,K24,K34,K
     144,K54,K64,K74,K84,K15,K25,K35,K45,K55,K65,K75,K85,K16,K26,K36,K46
     1,K56,K66,K76,K86,K17,K27,K37,K47,K57,K67,K77,K87,K18,K28,K38,K48,K
     158,K68,K78,K88
      DIMENSION ECPT(1)
      DIMENSION TMPOR1(36)
      DIMENSION Q4(64),Q5(64),Q6(64),Q7(64),Q8(64),Q9(64),XYZ1(1,3),XYZ2
     1(1,3),XYZ3(1,3),XYZ4(1,3),XYZ5(1,3),XYZ6(1,3),XYZ7(1,3),XYZ8(1,3),
     1XX(3,8),PT(1,2),DKX(1,1),DKY(1,1),DKZ(1,1),DN(3,1),DNA(8,3),DNB(8,
     13),DNC(8,3),DND(8,3),DNE(8,3),DNF(8,3),DNG(8,3),DNH(8,3),DNLA(8,3)
     1,DETA(1,1),DNLB(8,3),DETB(1,1),DNLC(8,3),DETC(1,1),DNLD(8,3),DETD(
     11,1),DNLE(8,3),DETE(1,1),DNLF(8,3),DETF(1,1),DNLG(8,3),DETG(1,1),D
     1NLH(8,3),DETH(1,1),BTA(3,1),BTB(3,1),BTC(3,1),BTD(3,1),BTE(3,1),BT
     1F(3,1),BTG(3,1),BTH(3,1),SAVEA(8,1),SAVEB(8,1),SAVEC(8,1),SAVEI(8,
     11),SAVEE(8,1),SAVEF(8,1),SAVEG(8,1),SAVEH(8,1),SUM1(1,1),SUM2(1,1)
     1,SUM3(1,1),SUM4(1,1),SUM5(1,1),SUM6(1,1),SUM7(1,1),SUM8(1,1),INDX(
     18,3),KI1(1,1),KI2(1,1),KI3(1,1),KI4(1,1),KI5(1,1),KI6(1,1),KI7(1,1
     1),KI8(1,1)
      DIMENSION IZ(1)
      DIMENSION TA(9),TOFF(3)
      DIMENSION XX1(3),XX2(3),XX3(3),XX4(3),XX5(3),XX6(3)
      DIMENSION XX7(3),XX8(3)
      COMMON /SYSTEM/DUMM(36),ISOP
      COMMON /SMA1X/Z(1)
      COMMON /SMA1BK/ICSTM,NCSTM
      DIMENSION K11(1,1),K12(1,1),K13(1,1),K14(1,1),K15(1,1),K16(1,1),K1
     17(1,1),K18(1,1),K21(1,1),K22(1,1),K23(1,1),K24(1,1),K25(1,1),K26(1
     1,1),K27(1,1),K28(1,1),K31(1,1),K32(1,1),K33(1,1),K34(1,1),K35(1,1)
     1,K36(1,1),K37(1,1),K38(1,1),K41(1,1),K42(1,1),K43(1,1),K44(1,1),K4
     15(1,1),K46(1,1),K47(1,1),K48(1,1),K51(1,1),K52(1,1),K53(1,1),K54(1
     1,1),K55(1,1),K56(1,1),K57(1,1),K58(1,1),K61(1,1),K62(1,1),K63(1,1)
     1,K64(1,1),K65(1,1),K66(1,1),K67(1,1),K68(1,1),K71(1,1),K72(1,1),K7
     13(1,1),K74(1,1),K75(1,1),K76(1,1),K77(1,1),K78(1,1),K81(1,1),K82(1
     1,1),K83(1,1),K84(1,1),K85(1,1),K86(1,1),K87(1,1),K88(1,1)
      COMMON /MATIN/MATID,INFLAG,ELTEMP,STRESS,SINTH,COSTH/SMA1IO/DUM1(1
     10),IFKGG,DUM2(1),IF4GG,DUM3(23)/SMA1CL/IOPT4,K4GGSW,NPVT/SMA1ET/NE
     1CPT(1),NGRID(8),JD1,ANGLE,JD2,MATID1,ID1,X1,Y1,Z1,ID2,X2,Y2,Z2,ID3
```

Figure 27 - Stiffness Matrix Subroutine and BLOCK DATA Subprograms, Sample Problem 2

```
     1,X3,Y3,Z3,ID4,X4,Y4,Z4,ID5,X5,Y5,Z5,ID6,X6,Y6,Z6,ID7,X7,Y7,Z7,ID8,
     1X8,Y8,Z8,DUMV(55)/SMA1DP/I,J,IS,IP,I1,I2,PI(1,1),TEMPOR(64),DETERM
     1,Q1(64),Q2(64),Q3(64)/MATOUT/G11,G12,G13,G22,G23,G33,RHO,ALPHA1,AL
     1PHA2,ALP12,TSUB0,GSUBE,SIGTEN,SIGCOM,SIGSHE
      EQUIVALENCE (ECPT,NECPT)
      EQUIVALENCE (X1,XYZ1(1,1)),(X2,XYZ2(1,1)),(X3,XYZ3(1,1)),(X4,XYZ4(
     11,1)),(X5,XYZ5(1,1)),(X6,XYZ6(1,1)),(X7,XYZ7(1,1)),(X8,XYZ8(1,1))
      EQUIVALENCE (K11(1,1),KI1(1,1)),(K12(1,1),KI2(1,1)),(K13(1,1),KI3(
     11,1)),(K14(1,1),KI4(1,1)),(K15(1,1),KI5(1,1)),(K16(1,1),KI6(1,1)),
     1(K17(1,1),KI7(1,1)),(K18(1,1),KI8(1,1)),(K21(1,1),KI1(1,1)),(K22(1
     1,1),KI2(1,1)),(K23(1,1),KI3(1,1)),(K24(1,1),KI4(1,1)),(K25(1,1),KI
     15(1,1)),(K26(1,1),KI6(1,1)),(K27(1,1),KI7(1,1)),(K28(1,1),KI8(1,1)
     1),(K31(1,1),KI1(1,1)),(K32(1,1),KI2(1,1)),(K33(1,1),KI3(1,1)),(K34
     1(1,1),KI4(1,1)),(K35(1,1),KI5(1,1)),(K36(1,1),KI6(1,1)),(K37(1,1),
     1KI7(1,1)),(K38(1,1),KI8(1,1)),(K41(1,1),KI1(1,1)),(K42(1,1),KI2(1,
     11)),(K43(1,1),KI3(1,1)),(K44(1,1),KI4(1,1)),(K45(1,1),KI5(1,1)),(K
     146(1,1),KI6(1,1)),(K47(1,1),KI7(1,1)),(K48(1,1),KI8(1,1)),(K51(1,1
     1),KI1(1,1)),(K52(1,1),KI2(1,1)),(K53(1,1),KI3(1,1)),(K54(1,1),KI4(
     11,1)),(K55(1,1),KI5(1,1)),(K56(1,1),KI6(1,1)),(K57(1,1),KI7(1,1)),
     1(K58(1,1),KI8(1,1)),(K61(1,1),KI1(1,1)),(K62(1,1),KI2(1,1)),(K63(1
     1,1),KI3(1,1)),(K64(1,1),KI4(1,1)),(K65(1,1),KI5(1,1)),(K66(1,1),KI
     16(1,1)),(K67(1,1),KI7(1,1)),(K68(1,1),KI8(1,1)),(K71(1,1),KI1(1,1)
     1),(K72(1,1),KI2(1,1)),(K73(1,1),KI3(1,1)),(K74(1,1),KI4(1,1)),(K75
     11(1,1),KI5(1,1)),(K76(1,1),KI6(1,1)),(K77(1,1),KI7(1,1)),(K78(1,1),
     1KI8(1,1)),(K81(1,1),KI1(1,1)),(K82(1,1),KI2(1,1)),(K83(1,1),KI3(1,
     11)),(K84(1,1),KI4(1,1)),(K85(1,1),KI5(1,1)),(K86(1,1),KI6(1,1)),(K
     187(1,1),KI7(1,1)),(K88(1,1),KI8(1,1))
      DATA TMPOR1/36*0.000/
      EQUIVALENCE (Z,IZ)
      PI(1,1)=3.14159265
      INFLAG=2
      DO 1 IP=1,8
      IF(NGRID(IP).EQ.NPVT)GO TO 2
    1 CONTINUE
      CALL MESAGE(-30,34,ECPT(1))
    2 CONTINUE
      MATID=MATID1
      ELTEMP=DUMV(1)
      SINTH=DSIN(PI(1,1)/180.*ANGLE)
      COSTH=DCOS(PI(1,1)/180.*ANGLE)
      CALL MAT(NECPT(1))
      IF(ISOP.EQ.-1)CALL MESAGE(-30,154,ECPT(1))
      IF(JD1.EQ.0)GO TO 1065
      IF(NCSTM.EQ.0)GO TO 1010
      DO 1020 I=1,NCSTM,14
      I1=ICSTM+I
      IF(JD1.NE.IZ(I1))GO TO 1020
      IF(IZ(I+1).EQ.1)GO TO 1030
      GO TO 1020
```

Figure 27 - Stiffness Matrix Subroutine and BLOCK DATA Subprograms, Sample Problem 2—Continued

```
 1000 CONTINUE
 1010 CALL MESAGE(-30,25,ID1)
 1020 CALL MESAGE(-30,155,ID1)
 1030 DO 1040 J=1,9
      I1=I+4+J
 1040 TA(J)=Z(I1)
      DO 1050 J=1,3
      I1=I+1+J
 1050 TOFF(J)=Z(I1)
      DO 1060 I=1,3
      XX1(I)=XYZ1(1,I)-TOFF(I)
      XX2(I)=XYZ2(1,I)-TOFF(I)
      XX3(I)=XYZ3(1,I)-TOFF(I)
      XX4(I)=XYZ4(1,I)-TOFF(I)
      XX5(I)=XYZ5(1,I)-TOFF(I)
      XX6(I)=XYZ6(1,I)-TOFF(I)
      XX7(I)=XYZ7(1,I)-TOFF(I)
 1060 XX8(I)=XYZ8(1,I)-TOFF(I)
      CALL GMMATD(TA,3,3,1,XX1,3,1,0,XX(1,1))
      CALL GMMATD(TA,3,3,1,XX2,3,1,0,XX(1,2))
      CALL GMMATD(TA,3,3,1,XX3,3,1,0,XX(1,3))
      CALL GMMATD(TA,3,3,1,XX4,3,1,0,XX(1,4))
      CALL GMMATD(TA,3,3,1,XX5,3,1,0,XX(1,5))
      CALL GMMATD(TA,3,3,1,XX6,3,1,0,XX(1,6))
      CALL GMMATD(TA,3,3,1,XX7,3,1,0,XX(1,7))
      CALL GMMATD(TA,3,3,1,XX8,3,1,0,XX(1,8))
      GO TO 1070
 1065 CONTINUE
      DO 3 I=1,3
    3 XX(I,1)=XYZ1(1,I)
      DO 4 I=1,3
    4 XX(I,2)=XYZ2(1,I)
      DO 5 I=1,3
    5 XX(I,3)=XYZ3(1,I)
      DO 6 I=1,3
    6 XX(I,4)=XYZ4(1,I)
      DO 7 I=1,3
    7 XX(I,5)=XYZ5(1,I)
      DO 8 I=1,3
    8 XX(I,6)=XYZ6(1,I)
      DO 9 I=1,3
    9 XX(I,7)=XYZ7(1,I)
      DO 10 I=1,3
   10 XX(I,8)=XYZ8(1,I)
 1070 CONTINUE
      PT(1,1)=-0.57735027D0
      PT(1,2)=-PT(1,1)
```

Figure 27 - Stiffness Matrix Subroutine and BLOCK DATA Subprograms, Sample Problem 2—Continued

```
      DKX(1,1)=1.0
      DKY(1,1)=1.0
      DKZ(1,1)=1.0
      DN(1,1)=0.125D0*(1.D0+PT(1,1))**2
      DN(2,1)=0.125D0*(1.D0-PT(1,1))**2
      DN(3,1)=0.125D0*(1.D0+PT(1,1))*(1.D0-PT(1,1))
      DNA(1,1)=-DN(2,1)
      DNA(1,2)=-DN(2,1)
      DNA(1,3)=-DN(2,1)
      DNA(2,1)=+DN(2,1)
      DNA(2,2)=-DN(3,1)
      DNA(2,3)=-DN(3,1)
      DNA(3,1)=+DN(3,1)
      DNA(3,2)=-DN(1,1)
      DNA(3,3)=+DN(3,1)
      DNA(4,1)=-DN(3,1)
      DNA(4,2)=-DN(3,1)
      DNA(4,3)=+DN(2,1)
      DNA(5,1)=-DN(3,1)
      DNA(5,2)=+DN(2,1)
      DNA(5,3)=-DN(3,1)
      DNA(6,1)=+DN(3,1)
      DNA(6,2)=+DN(3,1)
      DNA(6,3)=-DN(1,1)
      DNA(7,1)=+DN(1,1)
      DNA(7,2)=+DN(1,1)
      DNA(7,3)=+DN(1,1)
      DNA(8,1)=-DN(1,1)
      DNA(8,2)=+DN(3,1)
      DNA(8,3)=+DN(3,1)
      DNB(1,1)=-DN(3,1)
      DNB(1,2)=-DN(3,1)
      DNB(1,3)=-DN(2,1)
      DNB(2,1)=+DN(3,1)
      DNB(2,2)=-DN(1,1)
      DNB(2,3)=-DN(3,1)
      DNB(3,1)=+DN(2,1)
      DNB(3,2)=-DN(3,1)
      DNB(3,3)=+DN(3,1)
      DNB(4,1)=-DN(2,1)
      DNB(4,2)=-DN(2,1)
      DNB(4,3)=+DN(2,1)
      DNB(5,1)=-DN(1,1)
      DNB(5,2)=+DN(3,1)
      DNB(5,3)=-DN(3,1)
      DNB(6,1)=+DN(1,1)
      DNB(6,2)=+DN(1,1)
      DNB(6,3)=-DN(1,1)
```

Figure 27 - Stiffness Matrix Subroutine and BLOCK DATA Subprograms, Sample Problem 2—Continued

```
      DNB(7,1)=+DN(3,1)
      DNB(7,2)=+DN(3,1)
      DNB(7,3)=+DN(1,1)
      DNB(8,1)=-DN(3,1)
      DNB(8,2)=+DN(2,1)
      DNB(8,3)=+DN(3,1)
      DNC(1,1)=-DN(3,1)
      DNC(1,2)=-DN(2,1)
      DNC(1,3)=-DN(3,1)
      DNC(2,1)=+DN(3,1)
      DNC(2,2)=-DN(3,1)
      DNC(2,3)=-DN(1,1)
      DNC(3,1)=+DN(1,1)
      DNC(3,2)=-DN(1,1)
      DNC(3,3)=+DN(1,1)
      DNC(4,1)=-DN(1,1)
      DNC(4,2)=-DN(3,1)
      DNC(4,3)=+DN(3,1)
      DNC(5,1)=-DN(2,1)
      DNC(5,2)=+DN(2,1)
      DNC(5,3)=-DN(2,1)
      DNC(6,1)=+DN(2,1)
      DNC(6,2)=+DN(3,1)
      DNC(6,3)=-DN(3,1)
      DNC(7,1)=+DN(3,1)
      DNC(7,2)=+DN(1,1)
      DNC(7,3)=+DN(3,1)
      DNC(8,1)=-DN(3,1)
      DNC(8,2)=+DN(3,1)
      DNC(8,3)=+DN(2,1)
      DND(1,1)=-DN(1,1)
      DND(1,2)=-DN(3,1)
      DND(1,3)=-DN(3,1)
      DND(2,1)=+DN(1,1)
      DND(2,2)=-DN(1,1)
      DND(2,3)=-DN(1,1)
      DND(3,1)=+DN(3,1)
      DND(3,2)=-DN(3,1)
      DND(3,3)=+DN(1,1)
      DND(4,1)=-DN(3,1)
      DND(4,2)=-DN(2,1)
      DND(4,3)=+DN(3,1)
      DND(5,1)=-DN(3,1)
      DND(5,2)=+DN(3,1)
      DND(5,3)=-DN(2,1)
      DND(6,1)=+DN(3,1)
      DND(6,2)=+DN(1,1)
      DND(6,3)=-DN(3,1)
      DND(7,1)=+DN(2,1)
      DND(7,2)=+DN(3,1)
      DND(7,3)=+DN(3,1)
      DND(8,1)=-DN(2,1)
      DND(8,2)=+DN(2,1)
      DND(8,3)=+DN(2,1)
```

Figure 27 - Stiffness Matrix Subroutine and BLOCK DATA Subprograms, Sample Problem 2—Continued

```
      DNE(1,1)=-DN(2,1)
      DNE(1,2)=-DN(3,1)
      DNE(1,3)=-DN(3,1)
      DNE(2,1)=+DN(2,1)
      DNE(2,2)=-DN(2,1)
      DNE(2,3)=-DN(2,1)
      DNE(3,1)=+DN(3,1)
      DNE(3,2)=-DN(3,1)
      DNE(3,3)=+DN(2,1)
      DNE(4,1)=-DN(3,1)
      DNE(4,2)=-DN(1,1)
      DNE(4,3)=+DN(3,1)
      DNE(5,1)=-DN(3,1)
      DNE(5,2)=+DN(3,1)
      DNE(5,3)=-DN(1,1)
      DNE(6,1)=+DN(3,1)
      DNE(6,2)=+DN(2,1)
      DNE(6,3)=-DN(3,1)
      DNE(7,1)=+DN(1,1)
      DNE(7,2)=+DN(3,1)
      DNE(7,3)=+DN(3,1)
      DNE(8,1)=-DN(1,1)
      DNE(8,2)=+DN(1,1)
      DNE(8,3)=+DN(1,1)
      DNF(1,1)=-DN(3,1)
      DNF(1,2)=-DN(1,1)
      DNF(1,3)=-DN(3,1)
      DNF(2,1)=+DN(3,1)
      DNF(2,2)=-DN(3,1)
      DNF(2,3)=-DN(2,1)
      DNF(3,1)=+DN(2,1)
      DNF(3,2)=-DN(2,1)
      DNF(3,3)=+DN(2,1)
      DNF(4,1)=-DN(2,1)
      DNF(4,2)=-DN(3,1)
      DNF(4,3)=+DN(3,1)
      DNF(5,1)=-DN(1,1)
      DNF(5,2)=+DN(1,1)
      DNF(5,3)=-DN(1,1)
      DNF(6,1)=+DN(1,1)
      DNF(6,2)=+DN(3,1)
      DNF(6,3)=-DN(3,1)
      DNF(7,1)=+DN(3,1)
      DNF(7,2)=+DN(2,1)
      DNF(7,3)=+DN(3,1)
      DNF(8,1)=-DN(3,1)
      DNF(8,2)=+DN(3,1)
      DNF(8,3)=+DN(1,1)
      DNG(1,1)=-DN(3,1)
      DNG(1,2)=-DN(3,1)
      DNG(1,3)=-DN(1,1)
      DNG(2,1)=+DN(3,1)
```

Figure 27 - Stiffness Matrix Subroutine and BLOCK DATA Subprograms, Sample Problem 2—Continued

```
      DNG(2,2)=-DN(2,1)
      DNG(2,3)=-DN(3,1)
      DNG(3,1)=+DN(1,1)
      DNG(3,2)=-DN(3,1)
      DNG(3,3)=+DN(3,1)
      DNG(4,1)=-DN(1,1)
      DNG(4,2)=-DN(1,1)
      DNG(4,3)=+DN(1,1)
      DNG(5,1)=-DN(2,1)
      DNG(5,2)=+DN(3,1)
      DNG(5,3)=-DN(3,1)
      DNG(6,1)=+DN(2,1)
      DNG(6,2)=+DN(2,1)
      DNG(6,3)=-DN(2,1)
      DNG(7,1)=+DN(3,1)
      DNG(7,2)=+DN(3,1)
      DNG(7,3)=+DN(2,1)
      DNG(8,1)=-DN(3,1)
      DNG(8,2)=+DN(1,1)
      DNG(8,3)=+DN(3,1)
      DNH(1,1)=-DN(1,1)
      DNH(1,2)=-DN(1,1)
      DNH(1,3)=-DN(1,1)
      DNH(2,1)=+DN(1,1)
      DNH(2,2)=-DN(3,1)
      DNH(2,3)=-DN(3,1)
      DNH(3,1)=+DN(3,1)
      DNH(3,2)=-DN(2,1)
      DNH(3,3)=+DN(3,1)
      DNH(4,1)=-DN(3,1)
      DNH(4,2)=-DN(3,1)
      DNH(4,3)=+DN(1,1)
      DNH(5,1)=-DN(3,1)
      DNH(5,2)=+DN(1,1)
      DNH(5,3)=-DN(3,1)
      DNH(6,1)=+DN(3,1)
      DNH(6,2)=+DN(3,1)
      DNH(6,3)=-DN(2,1)
      DNH(7,1)=+DN(2,1)
      DNH(7,2)=+DN(2,1)
      DNH(7,3)=+DN(2,1)
      DNH(8,1)=-DN(2,1)
      DNH(8,2)=+DN(3,1)
      DNH(8,3)=+DN(3,1)
      CALL GMMATD(DNA,3,8,J,XX,8,3,0,Q1)
      IS=J
      DO 11 I=1,3
      DO 11 J=1,3
      IS=IS+1
      IJ=3*(J-1)+I
```

Figure 27 - Stiffness Matrix Subroutine and BLOCK DATA Subprograms, Sample Problem 2—Continued

```
   11 Q2(IS)=Q1(IJ)
      CALL INVERD(3,Q2,3,Q2,0,DETERM,IS,INDX)
      IF(IS.EQ.1)GO TO 12
      CALL MESAGE(-30,156,ECPT(1))
      CALL DUMP
      STOP
   12 IS=0
      DO 13 I=1,3
      DO 13 J=1,3
      IS=IS+1
      IJ=3*(J-1)+I
   13 Q3(IS)=Q2(IJ)
      CALL GMMATD(Q3,3,3,0,DNA,3,8,0,DNLA)
      DETA(1,1)=DETERM
      CALL GMMATD(DNB,3,8,0,XX,8,3,0,Q1)
      IS=0
      DO 14 I=1,3
      DO 14 J=1,3
      IS=IS+1
      IJ=3*(J-1)+I
   14 Q2(IS)=Q1(IJ)
      CALL INVERD(3,Q2,3,Q2,0,DETERM,IS,INDX)
      IF(IS.EQ.1)GO TO 15
      CALL MESAGE(-30,156,ECPT(1))
      CALL DUMP
      STOP
   15 IS=0
      DO 16 I=1,3
      DO 16 J=1,3
      IS=IS+1
      IJ=3*(J-1)+I
   16 Q3(IS)=Q2(IJ)
      CALL GMMATD(Q3,3,3,0,DNB,3,8,0,DNLB)
      DETB(1,1)=DETERM
      CALL GMMATD(DNC,3,8,0,XX,8,3,0,Q1)
      IS=0
      DO 17 I=1,3
      DO 17 J=1,3
      IS=IS+1
      IJ=3*(J-1)+I
   17 Q2(IS)=Q1(IJ)
      CALL INVERD(3,Q2,3,Q2,0,DETERM,IS,INDX)
      IF(IS.EQ.1)GO TO 18
      CALL MESAGE(-30,156,ECPT(1))
      CALL DUMP
      STOP
   18 IS=0
      DO 19 I=1,3
      DO 19 J=1,3
      IS=IS+1
      IJ=3*(J-1)+I
```

Figure 27 - Stiffness Matrix Subroutine and BLOCK DATA Subprograms, Sample Problem 2—Continued

```
 19 Q3(IS)=Q2(IJ)
    CALL GMMATD(Q3,3,3,0,DNC,3,8,0,DNLC)
    DETC(1,1)=DETERM
    CALL GMMATD(DND,3,8,0,XX,8,3,0,Q1)
    IS=0
    DO 20 I=1,3
    DO 20 J=1,3
    IS=IS+1
    IJ=3*(J-1)+I
 20 Q2(IS)=Q1(IJ)
    CALL INVERD(3,Q2,3,Q2,0,DETERM,IS,INDX)
    IF(IS.EQ.1)GO TO 21
    CALL MESAGE(-30,156,ECPT(1))
    CALL DUMP
    STOP
 21 IS=0
    DO 22 I=1,3
    DO 22 J=1,3
    IS=IS+1
    IJ=3*(J-1)+I
 22 Q3(IS)=Q2(IJ)
    CALL GMMATD(Q3,3,3,0,DND,3,8,0,DNLD)
    DETD(1,1)=DETERM
    CALL GMMATD(DNE,3,8,0,XX,8,3,0,Q1)
    IS=0
    DO 23 I=1,3
    DO 23 J=1,3
    IS=IS+1
    IJ=3*(J-1)+I
 23 Q2(IS)=Q1(IJ)
    CALL INVERD(3,Q2,3,Q2,0,DETERM,IS,INDX)
    IF(IS.EQ.1)GO TO 24
    CALL MESAGE(-30,156,ECPT(1))
    CALL DUMP
    STOP
 24 IS=0
    DO 25 I=1,3
    DO 25 J=1,3
    IS=IS+1
    IJ=3*(J-1)+I
 25 Q3(IS)=Q2(IJ)
    CALL GMMATD(Q3,3,3,0,DNE,3,8,0,DNLE)
    DETE(1,1)=DETERM
    CALL GMMATD(DNF,3,8,0,XX,8,3,0,Q1)
    IS=0
    DO 26 I=1,3
    DO 26 J=1,3
    IS=IS+1
    IJ=3*(J-1)+I
```

Figure 27 - Stiffness Matrix Subroutine and BLOCK DATA Subprograms, Sample Problem 2—Continued

```
   26 Q2(IS)=Q1(IJ)
      CALL INVERD(3,Q2,3,Q2,0,DETERM,IS,INDX)
      IF(IS.EQ.1)GO TO 27
      CALL MESAGE(-30,156,ECPT(1))
      CALL DUMP
      STOP
   27 IS=0
      DO 28 I=1,3
      DO 28 J=1,3
      IS=IS+1
      IJ=3*(J-1)+I
   28 Q3(IS)=Q2(IJ)
      CALL GMMATD(Q3,3,3,0,DNF,3,3,0,DNLF)
      DETF(1,1)=DETERM
      CALL GMMATD(DNG,3,3,0,XX,3,3,0,Q1)
      IS=0
      DO 29 I=1,3
      DO 29 J=1,3
      IS=IS+1
      IJ=3*(J-1)+I
   29 Q2(IS)=Q1(IJ)
      CALL INVERD(3,Q2,3,Q2,0,DETERM,IS,INDX)
      IF(IS.EQ.1)GO TO 30
      CALL MESAGE(-30,156,ECPT(1))
      CALL DUMP
      STOP
   30 IS=0
      DO 31 I=1,3
      DO 31 J=1,3
      IS=IS+1
      IJ=3*(J-1)+I
   31 Q3(IS)=Q2(IJ)
      CALL GMMATD(Q3,3,3,0,DNG,3,3,0,DNLG)
      DETG(1,1)=DETERM
      CALL GMMATD(DNH,3,3,0,XX,3,3,0,Q1)
      IS=0
      DO 32 I=1,3
      DO 32 J=1,3
      IS=IS+1
      IJ=3*(J-1)+I
   32 Q2(IS)=Q1(IJ)
      CALL INVERD(3,Q2,3,Q2,0,DETERM,IS,INDX)
      IF(IS.EQ.1)GO TO 33
      CALL MESAGE(-30,156,ECPT(1))
      CALL DUMP
      STOP
   33 IS=0
      DO 34 I=1,3
      DO 34 J=1,3
      IS=IS+1
      IJ=3*(J-1)+I
```

Figure 27 - Stiffness Matrix Subroutine and BLOCK DATA Subprograms, Sample Problem 2—Continued

```
 34 Q3(IS)=Q2(IJ)
    CALL GMMATD(Q3,3,3,0,DNH,3,8,0,DNLH)
    DETH(1,1)=DETERM
    BTA(1,1)=DKX(1,1)*DNLA(IP,1)
    BTA(2,1)=DKY(1,1)*DNLA(IP,2)
    BTA(3,1)=DKZ(1,1)*DNLA(IP,3)
    BTB(1,1)=DKX(1,1)*DNLB(IP,1)
    BTB(2,1)=DKY(1,1)*DNLB(IP,2)
    BTB(3,1)=DKZ(1,1)*DNLB(IP,3)
    BTC(1,1)=DKX(1,1)*DNLC(IP,1)
    BTC(2,1)=DKY(1,1)*DNLC(IP,2)
    BTC(3,1)=DKZ(1,1)*DNLC(IP,3)
    BTD(1,1)=DKX(1,1)*DNLD(IP,1)
    BTD(2,1)=DKY(1,1)*DNLD(IP,2)
    BTD(3,1)=DKZ(1,1)*DNLD(IP,3)
    BTE(1,1)=DKX(1,1)*DNLE(IP,1)
    BTE(2,1)=DKY(1,1)*DNLE(IP,2)
    BTE(3,1)=DKZ(1,1)*DNLE(IP,3)
    BTF(1,1)=DKX(1,1)*DNLF(IP,1)
    BTF(2,1)=DKY(1,1)*DNLF(IP,2)
    BTF(3,1)=DKZ(1,1)*DNLF(IP,3)
    BTG(1,1)=DKX(1,1)*DNLG(IP,1)
    BTG(2,1)=DKY(1,1)*DNLG(IP,2)
    BTG(3,1)=DKZ(1,1)*DNLG(IP,3)
    BTH(1,1)=DKX(1,1)*DNLH(IP,1)
    BTH(2,1)=DKY(1,1)*DNLH(IP,2)
    BTH(3,1)=DKZ(1,1)*DNLH(IP,3)
    CALL GMMATD(BTA,1,3,0,DNLA,3,8,0,Q1)
    DO 35 I=1,8
 35 SAVEA(I,1)=DETA(1,1)*Q1(I)
    CALL GMMATD(BTB,1,3,0,DNLB,3,8,0,Q1)
    DO 36 I=1,8
 36 SAVEB(I,1)=DETB(1,1)*Q1(I)
    CALL GMMATD(BTC,1,3,0,DNLC,3,8,0,Q1)
    DO 37 I=1,8
 37 SAVEC(I,1)=DETC(1,1)*Q1(I)
    CALL GMMATD(BTD,1,3,0,DNLD,3,8,0,Q1)
    DO 38 I=1,8
 38 SAVEI(I,1)=DETD(1,1)*Q1(I)
    CALL GMMATD(BTE,1,3,0,DNLE,3,8,0,Q1)
    DO 39 I=1,8
 39 SAVEE(I,1)=DETE(1,1)*Q1(I)
    CALL GMMATD(BTF,1,3,0,DNLF,3,8,0,Q1)
    DO 40 I=1,8
 40 SAVEF(I,1)=DETF(1,1)*Q1(I)
    CALL GMMATD(BTG,1,3,0,DNLG,3,8,0,Q1)
    DO 41 I=1,8
 41 SAVEG(I,1)=DETG(1,1)*Q1(I)
    CALL GMMATD(BTH,1,3,0,DNLH,3,8,0,Q1)
    DO 42 I=1,8
```

Figure 27 - Stiffness Matrix Subroutine and BLOCK DATA Subprograms, Sample Problem 2——Continued

```
   42 SAVEH(I,1)=DETH(1,1)*Q1(I)
      SUM1(1,1)=SAVEA(1,1)+SAVEB(1,1)+SAVEC(1,1)+SAVEI(1,1)+SAVEE(1,1)+S
     1AVEF(1,1)+SAVEG(1,1)+SAVEH(1,1)
      SUM2(1,1)=SAVEA(2,1)+SAVEB(2,1)+SAVEC(2,1)+SAVEI(2,1)+SAVEE(2,1)+S
     1AVEF(2,1)+SAVEG(2,1)+SAVEH(2,1)
      SUM3(1,1)=SAVEA(3,1)+SAVEB(3,1)+SAVEC(3,1)+SAVEI(3,1)+SAVEE(3,1)+S
     1AVEF(3,1)+SAVEG(3,1)+SAVEH(3,1)
      SUM4(1,1)=SAVEA(4,1)+SAVEB(4,1)+SAVEC(4,1)+SAVEI(4,1)+SAVEE(4,1)+S
     1AVEF(4,1)+SAVEG(4,1)+SAVEH(4,1)
      SUM5(1,1)=SAVEA(5,1)+SAVEB(5,1)+SAVEC(5,1)+SAVEI(5,1)+SAVEE(5,1)+S
     1AVEF(5,1)+SAVEG(5,1)+SAVEH(5,1)
      SUM6(1,1)=SAVEA(6,1)+SAVEB(6,1)+SAVEC(6,1)+SAVEI(6,1)+SAVEE(6,1)+S
     1AVEF(6,1)+SAVEG(6,1)+SAVEH(6,1)
      SUM7(1,1)=SAVEA(7,1)+SAVEB(7,1)+SAVEC(7,1)+SAVEI(7,1)+SAVEE(7,1)+S
     1AVEF(7,1)+SAVEG(7,1)+SAVEH(7,1)
      SUM8(1,1)=SAVEA(8,1)+SAVEB(8,1)+SAVEC(8,1)+SAVEI(8,1)+SAVEE(8,1)+S
     1AVEF(8,1)+SAVEG(8,1)+SAVEH(8,1)
      GO TO(100,200,300,400,500,600,700,800),IP
C     GENERATE THE MAIN VARIABLE
  100 CONTINUE
      K11(1,1)=SUM1(1,1)
      K12(1,1)=SUM2(1,1)
      K13(1,1)=SUM3(1,1)
      K14(1,1)=SUM4(1,1)
      K15(1,1)=SUM5(1,1)
      K16(1,1)=SUM6(1,1)
      K17(1,1)=SUM7(1,1)
      K18(1,1)=SUM8(1,1)
      GO TO 900
  200 CONTINUE
      K21(1,1)=SUM1(1,1)
      K22(1,1)=SUM2(1,1)
      K23(1,1)=SUM3(1,1)
      K24(1,1)=SUM4(1,1)
      K25(1,1)=SUM5(1,1)
      K26(1,1)=SUM6(1,1)
      K27(1,1)=SUM7(1,1)
      K28(1,1)=SUM8(1,1)
      GO TO 900
  300 CONTINUE
      K31(1,1)=SUM1(1,1)
      K32(1,1)=SUM2(1,1)
      K33(1,1)=SUM3(1,1)
      K34(1,1)=SUM4(1,1)
      K35(1,1)=SUM5(1,1)
      K36(1,1)=SUM6(1,1)
      K37(1,1)=SUM7(1,1)
      K38(1,1)=SUM8(1,1)
      GO TO 900
```

Figure 27 - Stiffness Matrix Subroutine and BLOCK DATA Subprograms, Sample Problem 2—Continued

```
  400 CONTINUE
      K41(1,1)=SUM1(1,1)
      K42(1,1)=SUM2(1,1)
      K43(1,1)=SUM3(1,1)
      K44(1,1)=SUM4(1,1)
      K45(1,1)=SUM5(1,1)
      K46(1,1)=SUM6(1,1)
      K47(1,1)=SUM7(1,1)
      K48(1,1)=SUM8(1,1)
      GO TO 900
  500 CONTINUE
      K51(1,1)=SUM1(1,1)
      K52(1,1)=SUM2(1,1)
      K53(1,1)=SUM3(1,1)
      K54(1,1)=SUM4(1,1)
      K55(1,1)=SUM5(1,1)
      K56(1,1)=SUM6(1,1)
      K57(1,1)=SUM7(1,1)
      K58(1,1)=SUM8(1,1)
      GO TO 900
  600 CONTINUE
      K61(1,1)=SUM1(1,1)
      K62(1,1)=SUM2(1,1)
      K63(1,1)=SUM3(1,1)
      K64(1,1)=SUM4(1,1)
      K65(1,1)=SUM5(1,1)
      K66(1,1)=SUM6(1,1)
      K67(1,1)=SUM7(1,1)
      K68(1,1)=SUM8(1,1)
      GO TO 900
  700 CONTINUE
      K71(1,1)=SUM1(1,1)
      K72(1,1)=SUM2(1,1)
      K73(1,1)=SUM3(1,1)
      K74(1,1)=SUM4(1,1)
      K75(1,1)=SUM5(1,1)
      K76(1,1)=SUM6(1,1)
      K77(1,1)=SUM7(1,1)
      K78(1,1)=SUM8(1,1)
      GO TO 900
  800 CONTINUE
      K81(1,1)=SUM1(1,1)
      K82(1,1)=SUM2(1,1)
      K83(1,1)=SUM3(1,1)
      K84(1,1)=SUM4(1,1)
      K85(1,1)=SUM5(1,1)
      K86(1,1)=SUM6(1,1)
      K87(1,1)=SUM7(1,1)
      K88(1,1)=SUM8(1,1)
C     NO EXPRESSION-MAIN VARIABLE IS MAIN EXPRESSION
```

Figure 27 - Stiffness Matrix Subroutine and BLOCK DATA Subprograms, Sample Problem 2—Continued

```
  900 CONTINUE
C     INSERT STIFFNESS PARTITION
      TMPOR1(1)=KI1(1,1)
      CALL SMA1B(TMPOR1,NGRID(1),-1,IFKGG,0.D0)
      IF(IOPT4.EQ.0.OR.GSUBE.EQ.0.)GO TO 901
      TEMPOR(1)=GSUBE
      CALL SMA1B(TMPOR1,NGRID(1),-1,IF4GG,TEMPOR)
      K4GGSW=1
  901 CONTINUE
      TMPOR1(1)=KI2(1,1)
      CALL SMA1B(TMPOR1,NGRID(2),-1,IFKGG,0.D0)
      IF(IOPT4.EQ.0.OR.GSUBE.EQ.0.)GO TO 902
      TEMPOR(1)=GSUBE
      CALL SMA1B(TMPOR1,NGRID(2),-1,IF4GG,TEMPOR)
      K4GGSW=1
  902 CONTINUE
      TMPOR1(1)=KI3(1,1)
      CALL SMA1B(TMPOR1,NGRID(3),-1,IFKGG,0.D0)
      IF(IOPT4.EQ.0.OR.GSUBE.EQ.0.)GO TO 903
      TEMPOR(1)=GSUBE
      CALL SMA1B(TMPOR1,NGRID(3),-1,IF4GG,TEMPOR)
      K4GGSW=1
  903 CONTINUE
      TMPOR1(1)=KI4(1,1)
      CALL SMA1B(TMPOR1,NGRID(4),-1,IFKGG,0.D0)
      IF(IOPT4.EQ.0.OR.GSUBE.EQ.0.)GO TO 904
      TEMPOR(1)=GSUBE
      CALL SMA1B(TMPOR1,NGRID(4),-1,IF4GG,TEMPOR)
      K4GGSW=1
  904 CONTINUE
      TMPOR1(1)=KI5(1,1)
      CALL SMA1B(TMPOR1,NGRID(5),-1,IFKGG,0.D0)
      IF(IOPT4.EQ.0.OR.GSUBE.EQ.0.)GO TO 905
      TEMPOR(1)=GSUBE
      CALL SMA1B(TMPOR1,NGRID(5),-1,IF4GG,TEMPOR)
      K4GGSW=1
  905 CONTINUE
      TMPOR1(1)=KI6(1,1)
      CALL SMA1B(TMPOR1,NGRID(6),-1,IFKGG,0.D0)
      IF(IOPT4.EQ.0.OR.GSUBE.EQ.0.)GO TO 906
      TEMPOR(1)=GSUBE
      CALL SMA1B(TMPOR1,NGRID(6),-1,IF4GG,TEMPOR)
      K4GGSW=1
  906 CONTINUE
      TMPOR1(1)=KI7(1,1)
      CALL SMA1B(TMPOR1,NGRID(7),-1,IFKGG,0.D0)
      IF(IOPT4.EQ.0.OR.GSUBE.EQ.0.)GO TO 907
      TEMPOR(1)=GSUBE
      CALL SMA1B(TMPOR1,NGRID(7),-1,IF4GG,TEMPOR)
      K4GGSW=1
```

Figure 27 - Stiffness Matrix Subroutine and BLOCK DATA Subprograms, Sample Problem 2—Continued

```
  907 CONTINUE
      TMPOR1(1)=KI8(1,1)
      CALL SMA1B(TMPOR1,NGRID(8),-1,IFKGG,0.D0)
      IF(IOPT4.EQ.0.OR.GSUBE.EQ.0.)GO TO 908
      TEMPOR(1)=GSUBE
      CALL SMA1B(TMPOR1,NGRID(8),-1,IF4GG,TEMPOR)
      K4GGSW=1
  908 CONTINUE
      RETURN
      END
```

PRELIMINARY PACKET FORTRAN CODING

```
      BLOCK DATA
      COMMON/IFPCOM/NOELEM,IAPFLG(24),IRANOS(48),IFX7PT(24),IFX7SQ(384)/
     1IFSCOM/NLEM,IFSN(24)
      DATA NOELEM,IAPFLG,NLEM/38,22*0,0,0,38/
      DATA IRANOS/44*0,16,20,0,4/
      DATA IFX7PT/22*0,802,0/
      DATA IFX7SQ/1,1,1,1,1,1,1,1,1,1,2,1,1,371*0/
      DATA IFSN/22*0,13,0/
      END
      BLOCK DATA
      COMMON /GPTCOM/NOELEM,NDATCN(12),NDATPR(12),NGRDPT(12),INDSCA(12),
     1NWDEST(12),IFSTPT(12)
      DATA NOELEM/38/
      DATA NDATCN/11*0,13/
      DATA NDATPR/11*0,0/
      DATA NGRDPT/11*0,8/
      DATA INDSCA/11*0,0/
      DATA NWDEST/11*0,46/
      DATA IFSTPT/11*0,2/
      END
      BLOCK DATA
      COMMON /EDSCOM/NOELEM,NDATCN(12),NGRDPT(12)
      DATA NOELEM/38/
      DATA NDATCN/11*0,13/
      DATA NGRDPT/11*0,8/
      END
      BLOCK DATA
      COMMON/EDTCOM/NOELEM,NWDEST(12),NGRDPT(12)
      DATA NOELEM/38/
      DATA NWDEST/11*0,46/
      DATA NGRDPT/11*0,8/
      END
```

Figure 27 - Stiffness Matrix Subroutine and BLOCK DATA Subprograms, Sample Problem 2—Continued

```
      BLOCK DATA
      COMMON /SM1COM/NOELEM,NWDEST(12)
      DATA NOELEM/38/
      DATA NWDEST/11*0,46/
      END
      BLOCK DATA
      COMMON /SM2COM/NOELEM,NWDEST(12)
      DATA NOELEM/38/
      DATA NWDEST/11*0,46/
      END
      BLOCK DATA
      COMMON /SDRCOM/NOELEM,NWDEST(12),NGRDPT(12),NWDSTM(12),NWDSTR(12),
     1NWDFOR(12),NPTSTR(12),NPTFOR(12)
      DATA NOELEM/38/
      DATA NWDEST/11*0,46/
      DATA NGRDPT/11*0,8/
      DATA NWDSTM/11*0,0/
      DATA NWDSTR/11*0,0/
      DATA NWDFOR/11*0,0/
      DATA NPTSTR/11*0,0/
      DATA NPTFOR/11*0,0/
      END
```

Figure 27 - Stiffness Matrix Subroutine and BLOCK DATA Subprograms, Sample Problem 2—Continued

REFERENCES

1. "The NASTRAN Theoretical Manual, Level 15," Edited by R.H. MacNeal, NASAN SP-221(01) (Apr 1972).

2. "The NASTRAN User's Manual, Level 15," Edited by C. W. McCormick, NASA SP-222(01) (Jun 1972).

3. "The NASTRAN Programmer;s Manual, Level 15," Edited by F.J. Douglas, NASA SP-223(01) (Sep 1972).

4. Martin, R., "A General Purpose Overlay Loader for CDC6000-Series Computers; Modification of the NASTRAN Linkage Editor," Naval Ship Research and Development Center Report 4062 (Apr 1973).

INITIAL DISTRIBUTION

Copies:	
1	ARMY FRANKFORD ARSENAL/D.FREDERICK
1	HARRY DIAMOND LABS/P.EMMERMAN
1	USAPA/W. BOLTE
1	USA WATERVLIET ARSENAL/G. O'HARA
1	DNL
1	CHONR/N. BASDEKAS
1	CHONR/S. BRODSKY
1	ONR 439/N. PERRONE
1	ONR CHICAGO/R. BUCHAL
1	MAT 03P2/S. ATCHISON
1	NRL/R. PERLUT
1	USNA LIB
1	USNA DEPT MATH
1	NAVPGSCOL LIB
1	NROTC & NAVADMINU, MIT
1	NAVWARCOL
1	SHIPS 2052
1	SHIPS 031/J. HUTH
1	SHIPS 0311/B. ORLEANS
1	ORD/L. PASIUK
1	NAVAIRDEVCEN/S. HUANG
1	NELC/E. NISSAN
1	NAVUSEACEN/D. BARACH
1	NAVUSEACEN/J. HUNT
1	NAVWPNSCEN/J. SERPANOS
1	CIVENGRLAB/J. CRAWFORD
1	NOL/R. EDWARDS
1	NWL/C. BLACKMON

Copies:	
1	NLONLAB NUSC/A. CARLSON
1	NAVSHIPYD BREM/LIB
1	NAVSHIPYD BSN/LIB
1	NAVSHIPYD CHASN/LIB
1	NAVSHIPYD MARE/LIB
1	NAVSHIPYD NORVA/LIB
1	NAVSHIPYD PEARL/LIB
1	NAVSHIPYD PHILA 240
1	NAVSHIPYD PTSMH/LIB
1	SEC/R. KELTIE
1	SEC/R. SIELSKI
1	SEC 6034B
1	ASD/ENJEA/R. KIELB
1	KIRTLAND AFB/CAPT HANSEN
12	DDC
1	HQS NASA/D. MICHEL
1	NASA AMES RES CEN/P. POLENTZ
1	NASA GODDARD SFC/J. MASON
1	NASA JFK SPACE CEN/H. HARRIS
1	NASA LANGLEY RES CEN/D. WEIDMAN
1	NASA MARSHALL SFC/R. MCCOMAS
1	JHU APPL PHYS LAB/W. CAYWOOD
1	JHU APPL PHYS LAB/G. DAILEY
1	JHU APPL PHYS LAB/J. O'CONNOR
1	JHU APPL PHYS LAB/R. RIVELLO
1	PENN STATE U ARL/J. CONWAY
1	U VA CIV ENGR/R. EPPINK
1	BELL AEROSPACE C-6/A. WOEHR

CENTER DISTRIBUTION

Copies:	
1	0000 NELSON PERRY W
1	0100 POWELL ALAN
1	1720 KRENZKE MARTIN A
1	1725 JONES REMBERT F JR
1	1727 KIERNAN THOMAS J
1	1730 STAVOVY ALEXANDER B
1	1800 GLEISSNER GENE H
1	1805 CUTHILL ELIZABETH H
1	1840 WRENCH JOHN W JR

Copies:	
1	1844 DHIR SURENDRA K
12	1844 GOLDEN MICHAEL E
12	1844 HURWITZ MYLES M
1	2744 LU YEH-PEI
30	5614 REPORTS DISTRIBUTION
1	5641 LIBRARY
1	5642 LIBRARY

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)*

Naval Ship Research and Development Center
Bethesda, Maryland 20034

2a. REPORT SECURITY CLASSIFICATION

UNCLASSIFIED

2b. GROUP

3. REPORT TITLE

DEFINIT—A NEW ELEMENT DEFINITION CAPABILITY FOR NASTRAN: USER'S MANUAL

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

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

Michael E. Golden
Myles M. Hurwitz

6. REPORT DATE

December 1973

7a. TOTAL NO. OF PAGES

200

7b. NO. OF REFS

4

8a. CONTRACT OR GRANT NO.

b. PROJECT NO. ZR 1040201

c. Work Unit 1-1844-009

d.

9a. ORIGINATOR'S REPORT NUMBER(S)

Report 4250

9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*

10. DISTRIBUTION STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY

NAVSHIPS 0311

13. ABSTRACT

To relieve the user of NASTRAN—the National Aeronautics and Space Administration's general purpose finite element, structural analysis computer program—from the necessity of becoming involved with internal aspects of NASTRAN when he adds a new element, a new element definition capability has been developed. This capability takes the form of a preprocessor which will generate, according to user specifications, the FORTRAN routines and tables required by NASTRAN for a new element.

This manual contains details and instructions on the use of the preprocessor, and provides numerous examples.

DD FORM 1 NOV 65 1473 (PAGE 1)

S/N 0101-807-6801

UNCLASSIFIED

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
NASTRAN New Element Definition Capability Finite Element Method Structural Analysis						