NON-NUMERICAL APPLICATIONS OF COMPUTER PROGRAMMING IN THE CONSTRUCTION OF PROBLEM ORIENTED LANGUAGES

Lawrence Bruce Elliott

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93940

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

NON-NUMERICAL APPLICATIONS OF COMPUTER PROGRAMMING IN THE CONSTRUCTION OF PROBLEM ORIENTED LANGUAGES

by

Lawrence Bruce Elliott

December 1979

Thesis Advisor: Gilles Cantin

Approved for public release; distribution unlimited.

T1942

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Non-Numerical Applications of Computer Programming in the Construction of Problem Oriented Languages		5. TYPE OF REPORT & PERIOD COVERED Master's & Engineer's Thesis; December 1979
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Lawrence Bruce Elliott		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, CA 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, CA 93940		12. REPORT DATE December 1979
		13. NUMBER OF PAGES 138
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report)
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

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

18. SUPPLEMENTARY NOTES

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

Computer Programming, Problem Oriented Language, Linear Algebra, Finite Element Analysis

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

A discussion of several non-numerical techniques that are useful in FORTRAN programming is presented. The use of these techniques is then illustrated with a problem oriented language called CAL-NPS. This last program is a derivation of a code named CAL written by Professor E. L. Wilson of the University of California, Berkeley, California.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-014-6601 |

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Approved for public release; distribution unlimited

Non-Numerical Applications of Computer Programming in the Construction of Problem Oriented Languages

by

Lawrence Bruce Elliott
Lieutenant Commander, United States Navy
B.S., United States Naval Academy, 1968

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

and the degree of

MECHANICAL ENGINEER

from the Naval Postgraduate School
December 1979

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA

ABSTRACT

A discussion of several non-numerical techniques that are useful in FORTRAN programming is presented. The use of these techniques is then illustrated with a problem oriented language called CAL-NPS. This last program is a derivation of a code named CAL written by Professor E. L. Wilson of the University of California, Berkeley, California.

TABLE OF CONTENTS

I. INTRODUCTION

In writing Problem Oriented Computer Languages several non numerical applications of computer programming must be mastered and the literature about such activities is nearly nil for the FORTRAN language. It is to these problems that this work addresses itself. The professional programmer would probably elect to write these codes in machine language but the user who only knows FORTRAN cannot afford time to learn the various machine languages in use today. All of the features discussed below are programmable in FORTRAN and illustrations are given in each use.

The CAL language is then used to illustrate how these features are incorporated in a problem oriented language. The CAL program was developed by Professor Edward L. Wilson of the University of California, Berkeley, California, for Structural Mechanics and Structural Engineering.

CAL combines matrix manipulation routines with direct stiffness computation options to produce a program for the automated analysis of structures. CAL also has significant capabilities as an instructional tool in linear algebra.

In order to use any structural analysis program it is necessary to idealize the structure into a series of joints connected by structural members. The joints are commonly referred to as nodal points and the structural members as elements. The program input consists of:

a) nodal point locations;

b) information on how nodes are connected;

c) boundary conditions that are applied to the nodes;

d) properties of the elements;

e) the loading to be analyzed.

The program output, at a minimum, consists of nodal point

deflection values and normally includes nodal forces and/or element stress values. CAL breaks this procedure down into a series of simple steps under the control of the user. It was designed to execute rapidly on small problems, so students can quickly see results of an analysis. There are options which allow the user to debug data without printing previously obtained results. Looping operations are also available. This allows a user to program iterative numerical algorithms, greatly expanding the usefulness of the program. In this version, CAL will solve problems with about 50 degrees of freedom (DOF).

The primary aim of the thesis work presented here was to modify CAL for use on the IBM 360/67 computer at the Naval Postgraduate School (NPS). The NPS version of CAL has both interactive and batch operating modes, and is nearly machine independent.

Most of this thesis effort was not involved in the numerical procedures used for calculations, but rather in the non-numerical areas of data input and processing. For this reason, numerical procedures will not be discussed. Subsequent chapters will provide:

a) a substantive discussion of some non-numerical techniques useful in writing scientific programs such as CAL;

b) a discussion of the internal organization of CAL-NPS;

c) instructions for the implementation of CAL on a different computer.

The appendices contain a listing of the FORTRAN code and an Instruction Manual for CAL-NPS.

Modifications to the program were carfully made to avoid changes in the instruction manual written by Professor Wilson. Ten commands have been added and three commands

have been extended for more flexibility in the interactive use of the program. The original instruction manual has been modified to reflect these changes and is reproduced in Appendix B. The author gratefully acknowledges Professor Wilson's permission to use the manual.

The non-numerical techniques discussed in Chapter II are this author's effort to explain how several state-of-the-art programming techniques work. It is hoped that readers will find this section beneficial in their own programming efforts. The CAL organization discussed in Chapter III is an excellent outline for general scientific programs. Additionaly, Chapter III provides details for writing user supplied subroutines. For readers who are interested only in how to use CAL, Chapter II and IV may be omitted.

II. NON-NUMERICAL TECHNIQUES IN FORTRAN

Chapter II is used to discuss FORTRAN programming techniques for writing general scientific programs. The reader is assumed to have some familiarity with the FORTRAN language and computer programming (i.e., completed CS2700 FORTRAN Programming or equivalent at NPS). After introductory remarks on the attributes of a scientific program, dynamic dimensioning is discussed in Section A, free and object time formats are discussed in Section B, data management is discussed in Section C, and finally, looping is discussed in Section D. The terms will be defined as encountered in these sections.

A general purpose program for the computer solution of a class of scientific problems should be simple to use, flexible and reliable. This requires a program that:

a) is modular, i.e., sections of program may be deleted, inserted or modified with ease;

b) has numerical methods that are stable and accurate for the type of problem encountered;

c) may require the user to have a knowledge of numerical methods necessary for the selection of the appropriate solution technique or boundary conditions but does not require knowledge of computer programming beyond the ability to prepare the input data for a program, initiate execution of a program, and create data files;

d) is not limited by array sizes in dimension or common statements;

e) permits iteration, i.e., the repeated execution of a user selected sequence of operations;

f) requires a minimum of central processor time and internal computer memory space (core) for execution;

A very useful method for meeting these requirements is to structure the program around the use of a problem oriented language. Such a language consists of a set of mnemonics representing each possible step in the general solution of the problem.

A mnemonic is a word that serves as a memory aid to identify some thing or some action. In FORTRAN, the mnemonic COS is used to represent the process of computing the cosine of an angle. All mnemonics can be thought of as representing a function subprogram, which can be very complex. The use of mnemonics to represent various operations in the formulation and solution of a particular problem is a very attractive method of writing a program for the general solution of a class of scientific problems. This technique promotes modularity and greatly simplifies use of the program. The mnemonic used for a particular operation or computation should be representative of the operation. For instance, a programmer might select the mnemonic SOLVE for the operation of computing the vector x from the matrix equation Ax = B where A and B are known. The mnemonic identifying a particular operation together with a list of arguments necessary for the computation is read from one data card. This card is followed by a series of data cards to provide data, not currently in core storage, for computation. The ability to enter the information in free format or to allow object time format specification dramatically reduces the work required to use the program.

References (1), (2), and (3) provide examples of problem oriented language programming. Apart from these few examples, there is very little written on the non-numerical techniques needed to write a program for the general solution of a scientific problem using a problem oriented

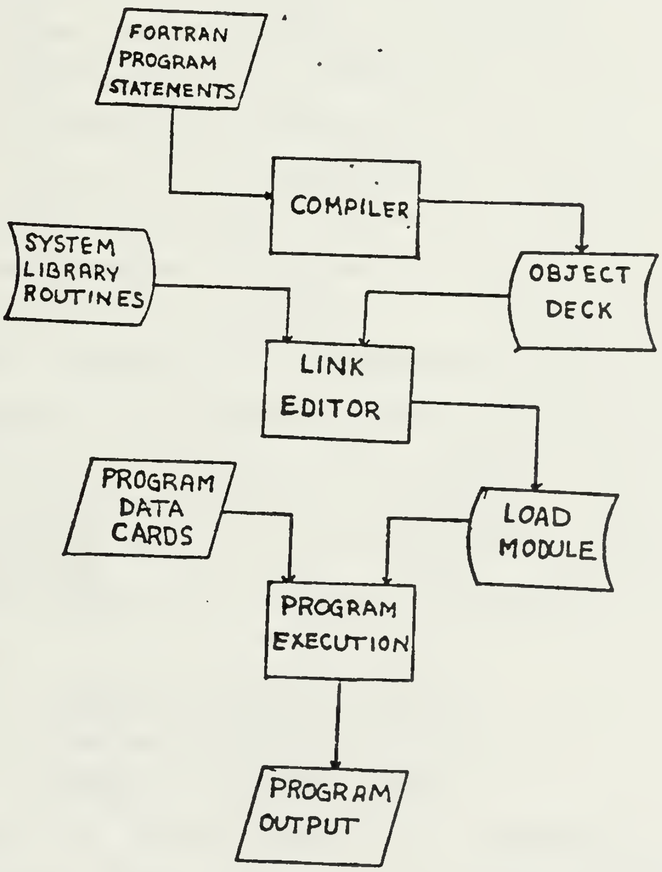
language for the solution. References (4) and (5) are useful exceptions and provide some very valuable information. The numerical techniques for execution time array dimensioning, execution time and free format, data management and the automated repetition of a user selected sequence of instructions will now be discussed in some detail.

A. DYNAMIC DIMENSIONING

In classical programming, array storage is reserved either in common or in dimension statements. Either way, the dimensions are usually set to the largest values envisioned for the type of problem under solution. When small or medium sized problems are solved, a considerable amount of core space is reserved but not used. Avoiding the wasted space normally requires re-doing the dimension and/or common statements in the entire program and then re-compiling the whole program.

Compilation is the first step in preparing a program for execution. In the compile step, the FORTRAN statements are interpreted to produce a set of machine language instructions, called an object deck, that the computer is capable of following. The next step is to link-edit the object deck. The link-editor collects all of the machine language instructions needed to read data, write, and evaluate functions called by the program; puts them in logical order with the object deck; and prepares an index to locate various sections of the program. The output from the link-editor is called the load module. The final step is the actual execution of the program. This is called the go step and consists of storing the load module in core then executing the instructions in sequence. Figure 1

demonstrates this process graphically. Usually it is possible to use a compiler that will optimize the machine language code to minimize the run time for a program in execution. Optimizing compilers take more time and require more core space to compile than do non-optimizing compilers. It is possible to compile and link-edit a program then store the load module on a mass storage device (magnetic tape or disk). This always results in saving central processor (CPU) time for subsequent program runs and can result in a significant reduction of the amount of core required.



Program Processing

FIGURE 1

The alternative to the classical style of programming is dynamic dimensioning. In order to understand the concept of dynamic dimensioning, it is necessary to have some appreciation for how a computer handles a common statement. Inside the computer, the common statement will cause the reservation of sufficient bytes of memory to store listed variables. A byte is a segment of a computer word which can contain either one alpha (hollerith) character or about two decimal digits. If the same common requires a different number of bytes in different subroutines, the largest requirement will be the one reserved. Any type of variable may occupy the reserved storage as long as the address of the starting byte of the variable is consistent with the type of variable. Different types of variables frequently have a different number of bytes in a word. For instance, a complex variable will normally have twice as many bytes per word as a real variable. To illustrate this point, consider a program where all variable names beginning with A are double precision (8 bytes), all variable names beginning with Z are complex double precision (16 bytes), and variables beginning with N are integers (4 bytes). The named common EXAMPLE might be listed as follows in different subroutines:

```
COMMON /EXAMPLE/ N(12)          (1)
COMMON /EXAMPLE/ A(3,2)         (2)
COMMON /EXAMPLE/ Z(3)           (3)
```

In each case the same number of bytes of storage are reserved (48) but the common has a significantly different type of variable! Introductory FORTRAN courses normally teach the student that a particular common should always be an exact duplicate each time it appears in the program

without much other discussion of the uses of common statements. The example lines above contrast sharply with this cautious style but can prove very useful in actual problem solving. Table 1 gives the address of the starting byte for each variable in EXAMPLE.

BYTE	VARIABLES
1	N(1),A(1,1),Z(1)
5	N(2)
9	N(3),A(2,1)
13	N(4)
17	N(5),A(3,1),Z(2)
21	N(6)
25	N(7),A(1,2)
29	N(8)
33	N(9),A(2,2),Z(3)
37	N(10)
41	N(11),A(3,2)
45	N(12)

Byte Addresses for Variables

TABLE 1

Note that N(3), A(2,1) and the imaginary portion of Z(1) all have the same byte as their starting address. Additionally, 48 bytes would have been reserved even if the array dimension of both (2) and (3) had been set at one since (1) still requires 48 bytes. One more feature of importance is that A(2,1), A(1,2) and A(3,2) contain the same information as the corresponding imaginary portions of the Z vector. This fact is quite useful in programming with complex variables.

The common EXAMPLE above demonstrates how a computer stores a two dimensional array but amplification may prove helpful. The right most subscript in a multiply dimensioned array is always the one that increases slowest as the array is going into computer storage. The address in memory of an element of a linear array is computed much faster than the address of an element from an array with multiple

subscripts. The address of the element A(I,J) of a two dimensional array can be computed in a corresponding linear array as follows:

$$L = ((J-1)*NR + I - 1)*IPR + 1 \quad (4)$$

where

NR is the number of rows in the matrix

IPR is the ratio of bytes per word of the matrix to bytes per word of the linear array.

For example, compute the index in N of A(3,2) from the common EXAMPLE:

here I=3, NR=3, J=2 and IPR = 8/4 = 2

$$L = ((2-1)*3 + 3 - 1)*2 + 1 = 11$$

Compare this answer with the results in table 1.

Dynamic dimensioning is a process that utilizes the computer's ability to store different types of variables in the same common array. Array storage is reserved in blank common with a large, one-dimensional, integer array (hereafter called the main array) for all subscripted variables needed for a particular computation step. This is normally done in a very short main program. Problem size can be controlled by the dimension of the main array in the main program. The dimension of the main array in all subroutines is set at 1 and never changed. Typically, to change a problem size, two cards of a six card main program need to be altered. See figure 2 for an example. The main program needs to be compiled, but there is no need to use an optimizing compiler. The remainder of the program can be in

a load module on a mass storage device.

```
C-----MAIN PROGRAM
      COMMON MAX,L(6000)
      MAX = 6000
      CALL DRIVER
      STOP
      END
C-----SUBPROGRAM TO CONTROL EXECUTION
      SUBROUTINE DRIVER
      COMMON MAX,L(1)
      .
      .
      .
      CALL COMPUTE(L(N1),L(N2),...,N3,N4)
      .
      .
      .
      RETURN
      END
C-----SUBROUTINE FOR COMPUTATION
      SUBROUTINE COMPUTE(A,B,...,NROW,ICOL)
      DIMENSION A(NROW,1),B(1)
      .
      .
      .
      RETURN
      END
```

Dynamic Dimensioning Sample Program

FIGURE 2

The main program calls a subprogram which controls execution, reserves storage in the main array for subscripted variables, and provides the address of array arguments in the main array to the computation subroutines. This subprogram must also ensure that there is sufficient room in the main array before a subscripted variable is allowed to be stored. Boundary misalignment occurs if the type of variable being stored requires more than one integer word for storage, and the starting address of the subscripted variable in the main array is not an address that is compatible with allowable addresses for the variable being stored. Some compiliers automatically correct for

boundary alignment but often this is accomplished with a considerable penalty in the time required for execution. Consider, for example, that a real double precision variable on an IBM 360 computer requires two integer words for storage. A boundary misalignment will occur unless the double precision variable is at an odd integer word address counting from the first integer word in blank common. In figure 2, MAX is the first integer word and L(1) is the second word in blank common. L(2) is, therefore, the first address available for the storage of the double precision element. L(2) and L(3) are required for the storage of one double precision number. If the double precision variable started in L(1), it would be out of proper alignment for this type of variable. A simple way to compute a proper starting address is:

N1 = NSIZE + MOD(NVAR + NSIZE , IPR) + 1

where

N1 is the starting address to be computed.

NVAR is the number of integer words used by the variables in blank common preceeding the main array, L.

NSIZE is the number of words in the main array that have been used.

IPR is the number of integer words required to store the variable being processed.

MOD(I,J) is an ANSI standard function which returns the remainder from (I / J).

A method to keep track of the starting addresses for subscripted variables will be discussed later. The amount of storage available may be compared with the amount required for storage of a subscripted variable as follows:

LEFT = MAX - (N1 + NEL * IPR) + 1

where

LEFT is the number of integer words remaining.

MAX is the number of integer words in the main array.

N1 is the starting address in the main array for the variable.

NEL is the number of elements in the variable. Note that when blanks are left between the end of an array and the start of the next array to compensate for boundary alignment, NEL of the leading array is effectively changed.

IPR is the number of integer words required per element.

Positive steps must be taken to prevent LEFT from becoming negative. One method to compute the number of words that have been used in the main array is

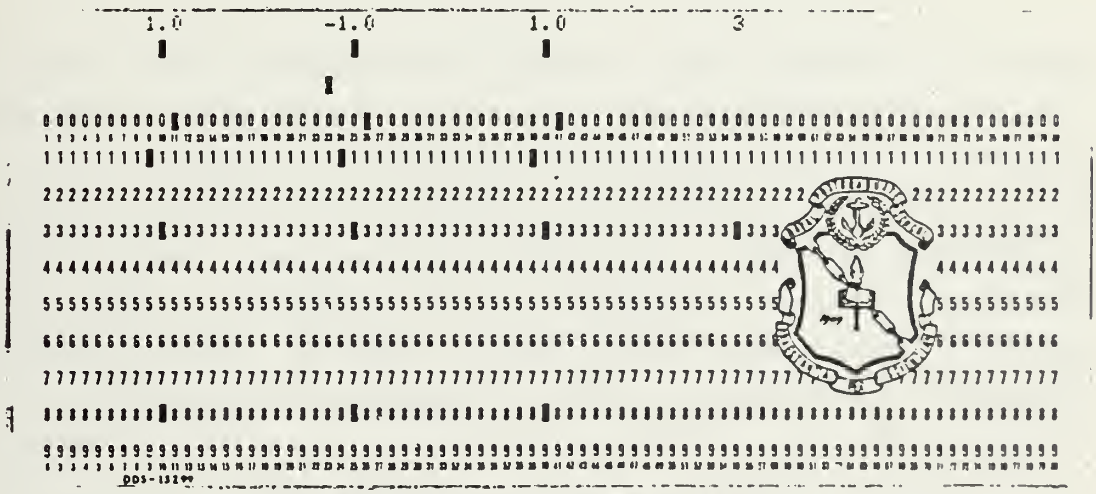
NSIZE = MAX - LEFT

where all variables are used as defined above.

Figure 2 is an example of some programming necessary to implement the dynamic dimensioning concept. The variable MAX and the dimension of the L array must have the same value in the main program. Outside the main program, the L array needs only to be dimensioned one since adequate storage is already reserved. In the control subprogram, N1 and N2 are used to provide starting addresses for subscripted variables to the computation subroutine. N3 and N4 are used to provide array dimensions. The dimension statement in the computation subroutines can use variables for array subscripts. Linear arrays and the final subscript of multiply subscripted arrays need only be set to one as shown in figure 2.

B. FREE AND OBJECT TIME FORMATS

Recall that in classical programming, data entered into the computer must be entered exactly in accordance with the format statement that has been compiled with the program. For instance, suppose that the coordinates and a reference number are required for a point in 3-space in format(3F15.5,I10). Figure 3 is the data card required for the point (1.0,-1.0,1.0) assigned reference number 3. Note that if the 3 is not in card column 55, the reference number assigned to the point will be wrong. When the input device is a terminal, the probability of making an error on entering data with this type of fixed format is high. Errors in entering data frequently require the user to restart the program and re-enter all the data to correct a small mistake.



Sample Fixed Format Data

FIGURE 3

Object time format (also called variable format) permits the user to enter the format for subsequent input or output operations as a part of the data deck at the time the

program enters execution. This means that it is possible to read an array prepared in (F10.0) format and an array prepared in (5E15.7) format without changing the format statement and recompiling the program.

Object time formatting is accomplished by reading a card containing the format for the data to follow, then reading or writing the data as specified in the format. A sample of the required programming to read data is:

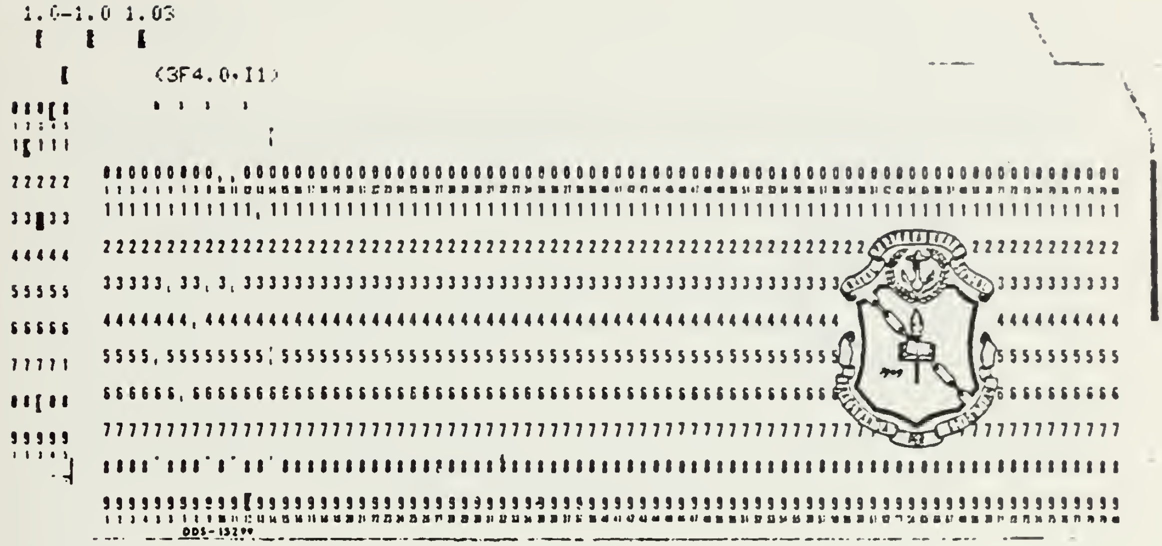
```
        DIMENSION FOR(20),A(3), ...
        .
        .
        .
    100 READ(5,200)FOR
    200 FORMAT(20A4)
    300 READ(5,FOR)(A(I),I=1,3),J
        .
        .
        .
```

Statement 100 represents an ordinary read command in FORTRAN according to the format statement numbered 200. This format statement causes 80 columns of hollerith data to be read (4 columns each into 20 variables). Statement 300 causes 4 real numbers to be read into the vector A. However, instead of being read according to a numbered format statement contained in the program, these numbers are read according to the information contained in the variable FOR. FOR might contain:

(3F4.0,I1)

causing the coordinates of the point discussed above to be read as the vector A in format (3F4.0) and the reference number to be read as the integer J in format (I1). Figure 4 shows the cards required to enter the point and reference

number.



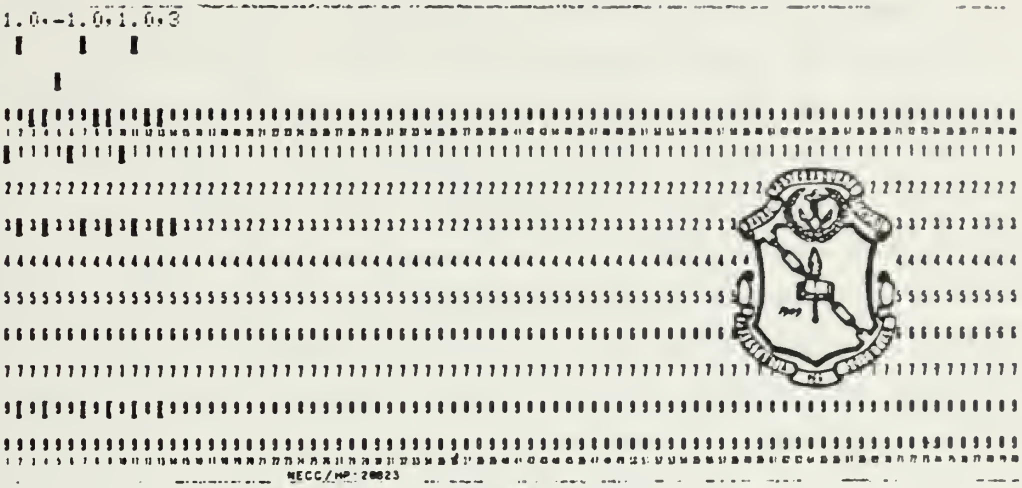
Sample Object Time Dimension Data

FIGURE 4

The first non-blank character on the data card containing the format must be an open parenthesis. The last non-blank character must be a close parenthesis. The characters in between must constitute a valid format for the data to be read or written.

Object time dimensioning allows the user to adjust formats to suit the data being input. However, if the user neglects to enter a format or enters one containing an error, the resulting problems can be frustrating. Free format relieves the user of the difficulties associated with using any type of predefined format. Figure 5 is an example of the data card required to enter the point and reference number discussed earlier. This example uses a comma (,) as

an end of field symbol and a blank ( ) as the end of record symbol.



Free Format Data

FIGURE 5

The first step of programming free-format input is to read the data card into an array called the card-image array. The data card must be read in format (80A1) so that the Jth element of the card-image array represents the character in column J of the data card. It is important to note that the value stored in the computer for a digit read as a hollerith character is different from the value stored for the same digit read as a number. The card-image array is then processed element by element to form the input variables. These input variables may be integers, real numbers or character strings, however, the variables must still be in the order expected by the program. If the expected input is a data card containing a step name and matrix argument list, a card containing real numbers will cause an error. Free format input allows the program and not the system error handling routine to determine appropriate corrective action.

The first step in processing an element of the card-image array is to identify the character stored in the element. How precise the identification must be depends on the application. The character being used as an end of field symbol and the character being used as an end of record must always be identified precisely. If, for instance, the expected data is a step name and matrix argument list, then the programmer might chose to identify all alphabetic characters as one type of symbol, all digits as a second type of symbol and all characters not otherwise identified as a third type of symbol. However, if the expected data contains a real number that may or may not have an exponent the choice of characters might be; 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, ., D, E, and others. The variable NSYM will be used to indicate the type of character.

The history of the processing is important in deciding what to do with a character. For instance, in the assembling of the real number 1.21, there are two types of character, digits and a decimal point. If the character encountered is a digit, the proper assembly action depends on whether the decimal point has occurred. If the decimal point has not occurred, the number being assembled must be multiplied by 10 and then the digit added to the number. On the other hand, if the decimal point has occurred, the digit must be divided by 10 then added to the number being assembled. The power of 10, N, is a counter representing the number of places to the right of the decimal this digit belongs. The varialbe IACT will be used to indicate the appropriate assembly action.

In addition to integral and fractional digits, a number may carry an algebraic sign and/or a sign in the exponent. The sign may occur at the beginning of the number or at the start of the exponent, however, a sign encountered elsewhere

will definetly be an error. It follows that for the assembly process of real numbers, the program performs one of the following operations: (a) sign identification, (b) integral digit assembly, (c) decimal point identification, (d) fractional digit assembly, and (e) exponent assembly. These processes are refered to as states. If the state is integral digit assembly and the character encountered is a sign, then obviously an error has occurred. On the other hand, if the character encountered is a decimal point, then there is no error but the state for the next operation needs to be changed to fractional digit assembly. A variable named ISTATE is introduced to indicate which one of the above operations the program last performed. Now, continuing with the assembly of a real number, the variable ISTATE may be assigned as follows:

ISTATE	Processing:
1	at the beginning of a Field.
2	after the beginning but before the decimal point.
3	after the decimal point.
4	after the exponent sign.
5	after an error has occurred in the field.

Again, for different applications, the programmer may define other states as needed.

The next step in processing a character from the card-image array is two-fold. The proper action to assemble the variable must be executed. This, of course, is given by the value of IACT. Also, if required, the state must be changed. A single integer can be used to represent both the next state and the assembly action required. For example, the information contained in ISTATE and IACT can be combined into a single variable, IPROC, by:

IPROC = IACT*100 + ISTATE

The information can be separated by:

IACT = IPROC/100

and

ISTATE = MOD(IPROC,100)

A matrix can be assembled that has the row index defined by ISTATE and the column index defined by NSYM. If an element of this matrix represents IPROC for the ISTATE and NSYM that define the element, then this matrix is an efficient method of programming the assembly task. The matrix so constructed is called the Symbol-State Matrix. Reference (4) contains futher discussion of Symbol-State matrices and their uses.

For illustration, consider the problem of reading a real linear array of length NEL. For simplicity, elements of the array may not have exponents. The state is identified as:

ISTATE	Processing:
1	at the beginning of the field.
2	before the decimal point.
3	after the decimal point.
4	after an error.

The steps that need to be programmed are:

a) Initialize the storage array, the symbol-state matrix, a hollerith translation table and the array pointers. A hollerith translation table is a vector that contains the hollerith character indicated by NSYM in the element with the index NSYM.

b) Read a card in A1 format into a temporary integer array.

c) Initialize the card column pointer called ICARD, the program state identifier (ISTATE), and the fractional digit counter called NCOUNT.

d) Identify the character occupying the ICARD column of the card-image array (NSYM) from the hollerith translation table.

e) Using NSYM and the current ISTATE as indices for entry in the Symbol-State matrix, called ISST, obtain IPROC (IPROC = ISST(ISTATE,NSYM). From IPROC, compute IACT and the new ISTATE.

f) Carry out the appropriate assembly action based on the value of IACT as follows:

1) Set a flag indicating the number is negative.

2) Assemble the digit into the integral part of the number (X = 10.0 * X + DIGIT).

3) Assemble the digit into the fractional part of the number (X = X + DIGIT/(10.0**NCOUNT)), and step the fractional digit counter (NCOUNT).

4) Print an error message. An action to compensate for the error may be included here.

5) Store the element in the storage array. If the array is filled, return; otherwise, reset NCOUNT and continue. If the symbol causing this action is EOR, reset ICARD and continue at step B.

Note that a plus sign or decimal point changes the state of the program but does not require an assembly action. After the assembly action above is complete, proceed to step 6.

6) Increment ICARD and test for the logical end of record. If ICARD is greater than the record length, reset ICARD, execute step f.5 and continue at step b. If ICARD is

less than or equal to the record length, continue at step d.

SYMBOL / STATE	COLUMN / ROW	DIGIT 1	+ 2	- 3	. 4	EOF 5	EOR 6	OTHERS 7
BOF/BOR	1	202	602	102	602	501	501	404
DIGIT BEFORE DECIMAL	2	202	404	404	603	501	501	404
DIGIT AFTER DECIMAL	3	303	404	404	404	501	501	404
ERROR	4	604	604	604	604	601	601	604

SYMBOL - STATE TABLE

TABLE 2

Table 2 is a Symbol-State matrix that the programmer might use to accomplish the variable assembly described above. The table presumes that the action to compensate for an error is to set the element equal to 0.0 (or some other value) and execute step 5. Action 7 is assumed to be the same as 5 except it allows for differentiating between an end of record symbol and an end of file symbol with a test

on the value of IACT. This table does not indicate an error if there is no decimal point. Instead, the decimal point is assumed after the last digit of the field. Additions to the Symbol-State Table or modifications to the logic are not difficult once the concept is understood.

Now consider the problem of assembling a character string, for instance, as a matrix name. If there are 4 bytes per integer word, then one element of the card-image array may contain:

'S'	bl	bl	bl

where "bl" represents the blank character. The byte containing the hollerith character S from the card-image array must be moved into the appropriate byte of the character string variable being assembled. This is not a trivial problem. ANSI FORTRAN does not support byte manipulation. Major computer manufacturers have added features to their FORTRAN which allow byte manipulation for string assembly. CDC allows for byte manipulation with MASK and SHIFT operations. In IBM FORTRAN the job can be done with a subroutine as follows:

```
      SUBROUTINE SHIFT(A,B,I,J)
      LOGICAL*1 A(1),B(1)
C-----THIS SUBROUTINE MOVES THE JTH BYTE OF
C     VARIABLE B INTO THE ITH BYTE OF
C     VARIABLE A.
      A(I) = B(J)
      RETURN
      END
```

The next update of the ANSI FORTRAN Standard tentatively includes variable word size specification with a "CHARACTER*n" declaration, where n is an integer. If the change materializes then the above subroutine will work for any machine by changing LOGICAL*1 to CHARACTER*1.

C. DATA MANAGEMENT

The comments on data management have been restricted to the management of data storage in core. Only one of several alternatives is discussed and the management system presented is not necessarily the most efficient method. It is advocated because it represents a straight forward approach and one does not have to be a computer scientist to implement the procedure.

The problem of in core data management can be reduced to the problem of storing, finding and deleting arrays in the main array mentioned previously. One way of organizing the main array is to store an array directory at the start of each array in the main array. The array directory needs to contain the following information:

a) The number of elements in the array.

b) The precision of the array (integer words/element).

c) A unique identification for the array.

d) The number of rows and/or the number of columns in the array.

D1	M1	D2	M2	D3	M3

Diagram of Array Storage

The process of reserving storage for an array should:

a) Ensure adequate space is available in the main array. Processing must be interrupted for resolution of the problem if sufficient room is not available.

b) Adjust NSIZE to compensate for the additional

array.

c) Create and store the array directory based on information from the calling program.

d) Check boundary alignment and adjust NEL for the previous array if necessary. It is possible to adjust NEL of the current array so that any subsequent array will be on a proper boundary no matter what the precision of the subsequent array. This is a simple solution but does make some available space unavailable.

e) Return the main array address for the first element of the array to the calling program.

Locating an array requires searching the array directories for a match with the identifier of the array. One method of doing this type of matching is:

```
IF(COMP(L(N),II)) GO TO nnn
```

where

N is the address, in the main array, of the first word of the array identifier being tested.

II contains the name to be matched.

nnn is the statement number that is branched to if a match is found.

COMP is a function subprogram defined as follows:

```
      FUNCTION COMP(I,J)
      DIMENSION I(3),J(3)
      LOGICAL COMP
      COMP = .FALSE.
      DO 10 K = 1,3
         IF(I(K) .NE. J(K)) RETURN
   10 CONTINUE
      COMP = .TRUE.
      RETURN
      END
```

This example assumes that three integer words are used to contain the array identifier. If the array is found, the main array address of the starting element and the directory information needs to be returned to the calling program. If the array is not found, processing should be interrupted for

resolution of the error.

The motivation for providing a method of deleting arrays is two-fold. If matrices are not stored with unique names, the locating system described above will only find the first array in storage whether or not it is the array needed. By using the delete operation under program (vice user) control, it is possible to ensure each array name is unique at the time it is created. The second reason for having a delete capability is that when an array has served its purpose, deletion under either program or user control allows the storage to be freed for other uses.

It is advantageous to leave all free storage postions at the end of the alocated storage. The simplest way to do this is to identify, in the main array, the address of the first position used by the array to be deleted (N1), and the first element of the next array in storage (N2), then:

```
          N3 = N2 - N1
          DO 10 I = N1,NSIZE
   10       L(I) = L(I+N3)
          NSIZE = NSIZE - N3
```

Note that if the directory is stored preceeding the array, then N1 would refer to the first element of the directory. In this case an alternate way of computing N3 is:

```
          N3 = NEL*IPR + NDIR
```

There is no reason to interrupt processing for error resolution when an array is not found to be deleted. It is helpful to print a message when an array is actually deleted especially if the DELETE operation is initiated by the program, for instance, to prevent duplicate array names.

The more complicated storage schemes for banded or skyline storage of arrays is also possible with this system. Banded and

Skyline storage schemes make more efficient use of storage by simply not storing many of the matrix elements

that are zero. The skyline storage algorithm is the most storage efficient of the two types. Reference (1) contains a substantive discussion of the skyline storage method, and examples of matrix manipulation subroutines for use with the skyline algorithm. Suffice it here to say that the skyline storage alogrithm requires the creation of two vectors. One is a vector of the elements from the matrix that are actually stored. The second vector contains the indices in the first vector of the diagonal elements of the matrix. For this case, it is convenient to use the same directory for both vectors. If there are NEL elements under the skyline, each element takes IPR integer words and there are NR rows in the matrix, then the storage required is

NDIR + NR + NEL * IPR

a typical block of the main array might look like

DIRECTORY	DIAGONAL ELEMENT INDICES	STORED ELEMENTS

The above presents a straightforward approach to data management in core. It is codeable in ANSI FORTRAN IV by an individual familiar with the basics of the language. It does not require a professional programmer for implementation.

D. LOOPING

The capability to repetitively execute a series of instructions (looping) is an important tool in problem solution. Frequently, for non-linear problems, there is no direct method of solution. An iterative scheme is normally employed to solve this type of problem. Brute force iteration is possible with any program by making multiple

runs of the program with suitable alteration to the input data for each run. This is very wasteful of both the user's time and computer time. On the other hand, if looping is possible, the user can program an iteration scheme using the commands available, then place the series of commands in a loop. A relatively simple method of providing a looping capability is to store the commands of the loop as a matrix in the main array. Then, with appropriate pointers, it is possible to execute a series of instructions repetitively from memory, just as if they were coming from the input device. Details of the method of looping used in the program CAL are contained in Chapter III.

A particularly attractive advantage of looping is the ability to iterate for selected eigenvalues of a matrix. This ability permits one to approximate the number of significant digits in the computer soluton of a problem as follows:

NSD = NDA - ALOG10(EMAX/EMIN)

where

NSD is the number of significant digits.

NDA is the number of digits available in the word size.

EMAX is the maximum eigenvalue of the matrix.

EMIN is the minimum eigenvalue.

ALOG10 is the ANSI standard function for the logarithm to the base 10.

This approximation holds for positive, definite, symmetric matrices. Computer results for problems where NSD approaches 0 should be viewed with extreme skepticism.

III. CAL ORGANIZATION

This chapter provides a general overview of the operation and organization of CAL. It is intended to provide sufficient information to permit users to program subroutines for use in CAL and to operate the CAL system sensibly.

CAL is a modern computer program written in the FORTRAN IV programming language. It was written as a teaching aid for illustrating the direct stiffness method of structural analysis. In addition to extensive linear algebra capabilities, the program provides several analytical alternatives for both static and dynamic analysis of elastic structures.

CAL execution is flexible, controlled by user selection of operations in a logical sequence (not unique) from the operations available in the program. The input data deck is a sequence of modules. The first card of each module contains the name of the requested operation, a list of matrix argument names, and integer parameters used in that operation. Comments may be placed on the card following a blank.

The operation card looks like this:

OPERATION,M1,M2,...,M9,N1,...N4 COMMENTS

M1 through M9 are matrix argument names. A matrix argument name consists of up to 8 alpha-numeric characters, the first being alphabetical. N1 throuch N4 are positive integers. An operation card may use from zero through nine matrix arguments and/or from zero through four integers. If required, the operation card is followed by a sequence of data cards containing numerical information used for the step. Some features of the output from CAL are also

controlled by user selectable operations. Details of the operations available and how to execute CAL at NPS are presented in Appendix B.

CAL is logically divided into four segments. The main segment contains the main program and service subroutines used commonally by all other segments. In subsequent descriptions, the main segment will be referred to as CAL. Among other things CAL contains the subroutines to deliver the next operation and the subroutines for data management. The three remaining segments will be referred to as GROUP 1, GROUP 2, and GROUP 3.

GROUP 1 contains subroutines which perform matrix manipulations, i.e., load, print, multiply, etc. GROUP 2 contains subroutines associated with formulation of the static problem and display of the computation results. For example, there are subroutines to:

a) input nodal geometry;

b) input boundary conditions;

c) input loading conditions;

d) form element stifness and mass matrices by several different methods;

e) combine element matrices into global matrices;

f) display results.

Finally, GROUP 3 contains subroutines associated with dynamic analysis. Two methods of evaluating the second order equations of motion, eigen analysis, and printer plotting routines are available.

Matrix storage in CAL is done dynamically as described in Chapter II. It is not possible to state with precision the maximum problem size. Experience shows that with the default parameters of the NPS version, static analysis problems are limited to approximately 48 degrees of freedom (DOF). Dynamic analysis problem size is controlled by both

the system DOF and the number of displacement vectors to be calculated. For a 25 DOF system, about 100 displacement vectors can be calculated.

All arrays and matrices in CAL are stored in a one dimensional matrix specified in blank common as L(XXXXX). XXXXX represents the total storage reserved for subscripted variables. Each array is preceeded by a directory which contains:

1) the number of elements in the array (NEL).

2) the number of integer words required to store an element (IPR),

3) the number of columns in the array,

4) the number of rows in the array,

5) the first four characters of the matrix name,

and 6) the last four characters of the matrix name.

An array uses (NEL*IPR + 6) integer words of storage. Larger problems can be solved by increasing the size of the L array in the main program. The variable MAX must be set to the current dimension of the L array. This data management system is described in Chapter II. CAL uses three subroutines for the data management sub-system. There is a subroutine called LIST which is used to reserve storage for new matrices as they are created. The calling arguments are, in order, NM, NR, NC, IPR, and IERR. NM is an array name assigned by the user. Array names must start with an alphabetic character and conatin 1 to 8 alpha-numeric characters.

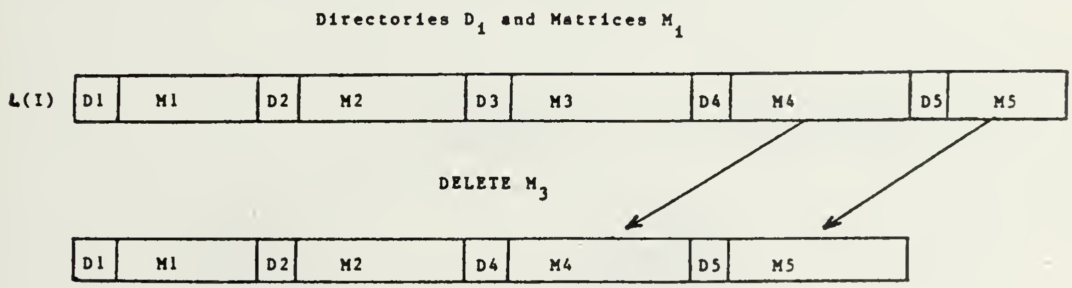
Since the name, NM, is used only in the directory, there is no reason to conform to FORTRAN name conventions. Thus, names beginning with the characters I through N may be assigned for real matrices with impunity. Call the stiffness matrix K, the mass matrix MASS and the nodal coordinate array NODES. All variables are treated as real,

double precision numbers unless clearly specified as integers in the command description.

NR and NC are respectively, the number of rows and the number of columns for the matrix to be stored. IPR is the number of integer words required to store an element. IERR is an error parameter. If IERR is not equal to 1, then an error has occurred during the list operation.

The subroutine LOCATE is used to locate arrays in the L array. The calling arguments are, in order, NM, NA, NR, NC, and IERR. NM, NR, NC, and IERR are used as described above. NA is the index in the L array for the first element of the array. In a subroutine call, L(NA) is used to pass the matrix NM to and from the called subroutine.

The remaining data management subroutine is DELETE. It has a single calling argument. NM is used as discussed previously. DELETE removes the matrix NM from the L array and frees the storage taken by NM at the end of the L array. Figure 7 illustrates this process.

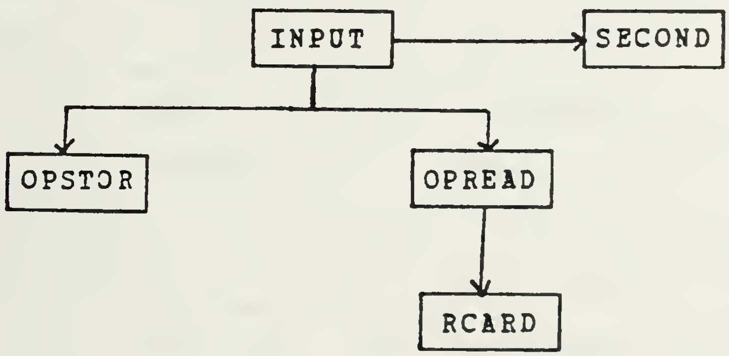


One Dimensional Array Storage

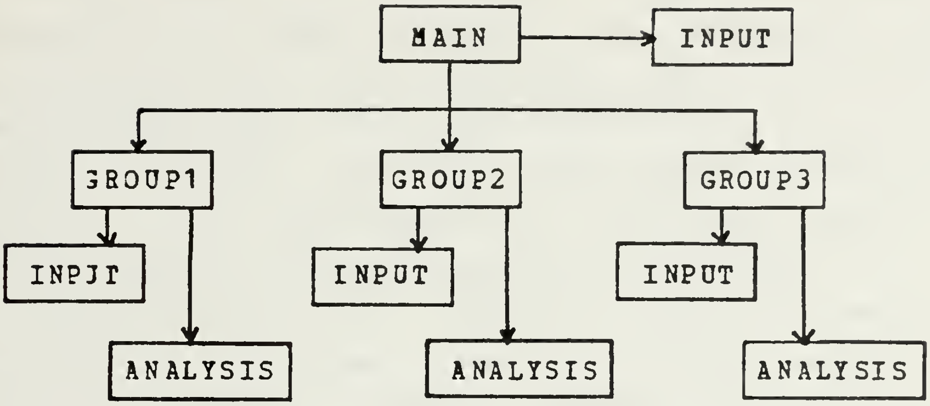
FIGURE 7

Now consider once more the function of the main program

segment CAL. In addition to the data management subroutines described above, the CAL segment contains four subroutines used to deliver the next operation to the calling program segment, and a timing subroutine. The subroutine INPUT determines whether the next operation is part of a loop or a new command from the input device. If the next operation is not part of a loop, INPUT calls the subroutine OPREAD. In turn, OPREAD calls the subroutine RCARD to read and interpret the free format information of the next command. Free format data input is discussed in Chapter II. If the command initiates looping, then OPREAD calls RCARD until all operations in the loop have been read. As operations are received by OPREAD, this subroutine stores them row-wise in the L array. After the last operation of the loop has been read, OPREAD calls LIST to prepare the directory. INPUT calls OPSTOR to provide each command in sequence from the loop matrix in the L array until the loop operations are complete. The subroutine SECOND is called by INPUT to determine the amount of time used in each computation step. A block diagram of the connections is:



A block diagram of the total system follows:



From this example, it is apparant that the program is organized so that only one group at a time needs to be resident in the core with CAL during execution. Thus, an overlay structure can be used which permits CAL to operate in a very small computer system.

Although CAL can operate in a small (64K) computer system, a mass storage device (magnetic tape or disk) must be available both to store program overlays and for use as a scratch pad during structural analysis operations. Element stiffness and mass matrices are written out on a mass storage device as they are created, then read back into the computer memory as needed for assembly into the global matrices. This method minimizes the amount of storage required in the core. Space is required to store the matrices only an element at a time, no matter how many elements are used.

The NPS version of CAL also contains commands which permit a user to stop execution before the end of a problem then resume at a later time. The operations also use a mass storage device to read or write the contents of the L array and certain other parameters.

One of the features of CAL which makes it so flexible is the availability of looping. Loops can be nested up to

five deep. Loops are initiated by the LOOP command and terminated by the NEXT command. For operations within a loop which require data, the data cards must appear in proper sequence, but after the last NEXT card of the loop. An example is shown with the loop command description in Appendix B. A skip command is available which allows selective skipping of operations within a loop.

A second feature of CAL which gives it really great flexibility is that user coded subroutines can be called under program control. GROUP 1 contains two operation names reserved for user coded subroutines. These names are USERA and USERB. A user coded subroutine might be used, for instance, to compute equivalent nodal forces for a distributed load.

Two subroutines are needed for a user operation. The actual numerical operation is coded normally in a subroutine with any name not already used in the CAL program. A buffer subroutine must then be written to interface the numerical computation with the CAL data management system. To illustrate the technique of interface, consider the familiar example of matrix multiplication (already available in CAL). Symbolically the problem is to compute C where C = A*B. The basic algorithm for matrix multiplication is

$$C(I,J) = A(I.K)*B(K,J)$$

A subroutine for the job might be:

```
      SUBROUTINE MULTIPLY(A,B,C,NRA,NCA,NCB)
      DIMENSION A(NRA,NCA),B(NCA,NCB),C(NRA,NCB)
      DO 20 I=1,NRA
      DO 20 J=1,NCA
      X = 0.0
      DO 10 K=1,NCA
   10 X = X + A(I,K)*B(K,J)
   20 C(I,J) = X
```

```
      RETURN
      END
```

The operation card might be selected as:

```
USERA,M1,M2,M3  COMMENTS
```

where M1, M2, and M3 are respectively the matrix argument names for the dummy arguments A, B, and C. The recommended steps to be programmed in the buffer subroutine named USERA are:

a) locate M1 and M2;

b) ensure the number of columns in A is equal to the number of rows in B;

c) call the DELETE subroutine to ensure that a matrix with the same name as M3 does not already exist in storage;

d) call LIST to reserve storage for M3;

e) test IERR to ensure that no errors have occurred in the above steps;

and f) call subroutine MULTIPLY to complete the operation;

At the minimum, steps a, d, and f must be programmed. Here is a possible subroutine to meet the requirement.

```
      SUBROUTINE USERA(IERR)
      COMMON MAX,NDP,L(1)
      COMMON /CARD/ INHOL(3,10),N(4)
      CALL LOCATE(INHOL(1,2),IA,NRA,NCA,IERR)
      CALL LOCATE(INHOL(1,3),IB,NRB,NCB,IERR)
      IF(NRA.NE.NCA) IERR= 2
      CALL DELETE(INHOL(1,4))
      CALL LIST(INHOL(1,4),NRA,NCB,NDP,IERR)
      CALL LOCATE(INHOL(1,4),IC,NRA,NCB,IERR)
      IF(IERR.NE.1) RETURN
      CALL MULTIPLY(L(IA),L(IB),L(IC),NRA,NCA,NCB)
      RETURN
```

```
      END
```

LOCATE, LIST, and DELETE are used as described previously. Blank common appears so that USERA has access to the L array and the parameter NDP. The variable NDP is initialized at 2, the number of integer words required to store a double precision variable. The named common CARD is initialized by RCARD or OPSTOR. The INHOL array contains the operation name and the matrix argument names. Each name is stored columnwise, i.e., column 1 has the operation name (USERA in this case), column 2 has the name of thematrix argument M1, etc. Row 1 elements have the first four characters, and row 2 elements contain the remaining four characters of the respective argument. Row 3 is not used in the NPS version of CAL. In the first call to subroutine LOCATE, INHOL(1,2) provides the name assigned by the user to matrix argument M1. On return from LOCATE, IA will be the index in the L array of the first element of M1. IB and IC respectively provide the same information for M2 and M3. Other variables are used as discussed previously.

Now, generalizing the procedure of the above example, the user must program two subroutines to define a user operation for CAL. One subroutine performs the operation, and the second acts as a buffer between CAL and the operation. Two names are available, USERA and USERB, for the buffer subroutine. Both have a single passing argument, the parameter IERR. The program assumes an error has occurred when IERR takes on a value other than 1. The buffer subroutine must:

a) locate matrix arguments designated for the operation,

b) reserve storage for arrays created by the operation,

c) pass the elements of the L array for the matrix

parameters used in the operation.
In addition, it is recommended that the buffer subroutine:

a) ensure that the matrix arguments to be used are compatible with the operation, i.e., square, symmetrical, etc.

b) call delete with names of matrices to be created prior to reserving storage. This will prevent matrices with duplicate names.

c) test IERR to ensure no error has occurred prior to calling the operation subroutine.

The operation subroutine can be coded with normal FORTRAN techniques.

In this chapter, the internal organization of CAL has been described, along with several important features of the program. Most of the non-numerical procedures described in the preceeding chapter are illustrated in CAL. This chapter has highlighted the flexibility of CAL and provided an example to assist users in coding their own operations to be executed by CAL.

IV. IMPLEMENTING CAL ON A DIFFERENT COMPUTER

In this chapter, the procedure to convert CAL for use on a different computer system is discussed. The NPS version of CAL has been organized to minimize the difficulty associated with implementing the program on a new computer. There remains, however, some things that are machine dependent.

The following information is required for the new system.

a) How many characters can be stored in one integer word?

b) How many integer words does it take to store a real variable? For structural analysis, a real variable should contain at least 12 significant figures.

c) What method can be used to manipulate bytes?

d) How is elapsed CPU time computed?

The answers to the above questions determine the complexity of the conversion task.

A. IBM FORTRAN STATEMENTS

There are a few statements in this version of CAL solely in the IBM FORTRAN Language. The statement "IMPLICIT REAL*8(A-H,O-Z)" makes all real variables double precesion. This statement appears in almost every subroutine. It should be replaced by an appropriate statement when using another computer. The calls to ERRSET suppress printout of error messages. These statements appear together in subroutine CAL1 and must be removed. They do not need to be replaced. The call to SETIME in the main program should be removed for use on non-IBM machines.

B. TIME COMPUTATION

There is no standard method of computing elapsed CPU time. The subroutine SECOND must be rewritten to compute elapsed CPU time in seconds for the new system. The library of subroutines for the computer system should have a suitable routine.

C. BYTE MANIPULATION

As previously mentioned, there is no standard way to shift bytes in a variable. The subroutine MOVEB will work if one byte logical variables are available. If this is not the case, than MOVEB must be rewritten using a byte manipulation technique suitable for the computer system.

D. OPERATION AND MATRIX NAMES

In this version, names are stored four characters to a variable since the IBM 360/67 computer uses four bytes per standard integer word. Names are stored column-wise in vectors of three elements. The parameter NH must be set to the number of elements required for storage of a name on the new system. LBYTE must be set to the number of characters per integer word. NH should be set such that NH*LBYTE at least 6. Both of the parameters are set in the BLOCK DATA subprogram. Data statements initialize operation names in the BLOCK DATA subroutine and in all three of the GROUP subroutines. All these data statements must be adjusted so that they initialize LBYTE characters per element. Format statements for write commands with output in format A4

should be changed. For the most part the output format wAd should be changed so that w = NH and d = LBYTE. Other uses should be apparent to the programmer.

E. BOUNDARY ALIGNMENT

There are three potential areas which may cause boundary alignment problems. NDP is set in the main program to the number of integer words needed to store a real variable. NDIR is the number of integer words used for a matrix directory. The first possible problem area is the relation between NDIR and NDP. If NDIR is not evenly divisable by NDP, add the remainder to the equation for NDIR in the BLOCK DATA Subroutine. The second possible problem area is in the Subroutine LIST. This subroutine contains a test to ensure the number of elements reserved for a single precision array is evenly divisible by NDP. If NDP is more than 2, the test is not valid. A better test would be

IF(NP.EQ.1) I(1) = NR * NC + MOD(NR*NC,NDP).

Finally, if NDP is greater than 2, then blank common must be adjusted so that there are NDP integer variables preceeding the L array in the main program and all subroutines where blank common appears.

V. CONCLUSIONS AND RECOMMENDATIONS

CAL provides a flexible tool for teaching modern sturctural analysis. The general matrix operations make it useful in the area of linear algebra as well as structural analysis. The use of CAL is highly encouraged.

A. RECOMMENDED MODIFICATIONS

A graphical display capability should be incorporated in CAL as soon as the equipment becomes available. It would be espically helpful to have a display of the element forces from the FORCE operation. Additionally, incorporation of heat transfer options into the program should be considered. The heat transfer capability would be a valuable tool for students in the department.

B. OTHER APPLICATIONS

The organization of CAL provides an excellent outline for development of other major analytical programs. The storage scheme can be simply modified to permit matrix storage using the skyline algorithm. Large problems can then be accomodated by incorporating existing algorithms for an out-of-core equation solver.

APPENDIX A - PROGRAM LISTING

```
C-----MAIN PROGRAM
C-----SET PROGRAM CAPICITY
      COMMON MTOT,NDP,L(6000)
      COMMON /ERROR/ NERR,MODE,NTIME
      MTOT = 6000
C-----NDP IS NUMBER OF COMPUTER WORDS USED BY A REAL VARIABLE.
      NDP = 2
C
C       CALL SETIME
C
C-----SETIME IS IBM 360 ROUTINE TO INITIALIZE CPU TIMER.
C
      CALL CAL1
      STOP
      END
C
C-----THE CAL PROGRAM IS A COMPUTER ANALYSIS LANGUAGE FOR THE STATIC
C-----AND DYNAMIC ANALYSIS OF STRUCTURES.  PROBLEM SIZE IS CONTROLLED
C-----BY THE DIMENSION IN BLANK COMMON AND THE VARIABLE MTOT.
C-----ALL PROBLEM ARRAYS AND THIER DIRECTORIES ARE STORED IN BLANK
C-----COMMON.  STRUCTURAL ANALYSIS OPERATIONS REQUIRE SCRATCH STORAGE.
C-----THE PROGRAM IS SET TO USE FILE FT01F001 FOR THIS.
C-----PROGRAMMED BY PROF EDWARD L. WILSON, CIVIL ENGINEERING DEPT,
C-----UNIVERSITY OF CALIFORNIA, BERKELEY, CA.
C
      SUBROUTINE CAL1
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /PSIZE/ NO,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR
      COMMON /NAMES/ LOOP(3),NEXT(3),LSKIP(3),LNAME(3),NSTART(3),
     1 LSTOP(3)
      COMMON /LOOPD/ LA,IN,INDEX(5),LSTART(5),IJKLM(5),NW
      COMMON /CARD/ INHOL(3,10),N(3),S1,S2
      COMMON /ERROR/ NERR,MODE,NTIME
      LOGICAL COMP
C
C-----SET INITIAL CONDITIONS
C
      CALL ERRSET(208,256,-1,1,1)
      CALL ERRSET(207,256,0,1,1)
      CALL ERRSET(209,256,0,1,1,216)
C
C-----ERRSET IS ERROR HANDLING ROUTINE FOR IBM 360 MACHINES.  THIS
C-----CALL SUPPRESSES PRINTOUT OF UNDERFLOW ERROR MESSAGES (IHC208) AND
C-----TRACEBACK OF SEVERAL OTHER ERRORS.
C
      NW=10*NH+4
  400 LA = 0
      IN = 0
      IERR = 1
      CALL DELETE(LNAME)
      CALL INPUT(IERR)
      IF(IERR .NE. 1) GO TO 500
      LAST=1
      GO TO 110
C-----  SEARCH GROUPS OF OPERATIONS FOR INPUT OPERATION
  100 IF(LAST.EQ.1) GO TO 300
  110 CALL GROUP1(LAST,IERR)
      IF(LAST.EQ.2) GO TO 300
      IF (IERR .NE. 1) GO TO 500
      CALL GROUP2(LAST,IERR)
      IF(LAST.EQ.3) GO TO 300
      IF (IERR .NE. 1) GO TO 500
      CALL GROUP3(LAST,IERR)
```

```
      IF (IERR .NE. 1) GO TO 500
      GO TO 100
  300 WRITE(NERR,3000)
  500 IF(MODE .NE. 1) GO TO 400
      NO = 0
  600 CALL RCARD(IERR)
      IF(COMP(INHOL(1,1),NSTART))GO TO 400
      IF(COMP(INHOL(1,1),LSTOP)) STOP
      GO TO 600
 3000 FORMAT (30HOOPERATION UNDEFINED OR BLANK    )
      END
      BLOCK DATA
      IMPLICIT REAL*8 (A-H,O-Z)
      LOGICAL INDIC
      COMMON /PSIZE/ NO,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR,INDIC
      COMMON /RECON/ IN(80),INDIGT(80),ITAB(5,4),NUM(10),LBYTE
      COMMON /NAMES/ LOOP(3),NEXT(3),LSKIP(3),LNAME(3),NSTART(3),
     1 LSTOP(3)
      COMMON /STR/ ST(9,16),XM(16),S(16,16),ND,NS,NRP,LM(16)
      COMMON /ERROR/ NERR,MODE,NTIME
      COMMON /PSAVE/ NUMNP,NEQ,NPROP,NUMEL,NTRUSS,NT,NSAVE
C-----NH IS NUMBER OF WORDS NEEDED TO CONTAIN 6 HOLLERITH CHARACTERS.
C-----NDIR = NH + 4*NSP
      DATA NO/0/,NARA/0/,NSIZE/1/,NSP/1/,NH/2/,NDIR/6/,INDIC /.TRUE./
C-----NREAD IS READ FILE DEVICE NUMBER.
C-----NWRITE IS WRITE FILE DEVICE NUMBER.
      DATA NREAD/5/,NWRITE/6/
      DATA NUM/2H1 ,2H2 ,2H3 ,2H4 ,2H5 ,2H6 ,2H7 ,2H8 ,2H9 ,2H0 /
      DATA ITAB/2*500,205,104,205,500,303,405,303,405,2*102,2*104,
     1 6*500/
C-----LBYTE IS NUMBER OF BYTES PER  INTEGER WORD.
      DATA LBYTE/4/
      DATA LOOP /4HLOOP,4H    ,4H    /
      DATA NEXT /4HNEXT,4H    ,4H    /
      DATA LSKIP /4HSKIP,4H    ,4H    /
      DATA LNAME/4HL1NA,4HME  ,4H    /
C-----NT IS FILE DEVICE NUMBER FOR SCRATCH TAPE.
C-----NSAVE IS FILE DEVICE NUMBER FOR SAVE FILE.
      DATA NT/1/,NSAVE/2/
      DATA NSTART/4HSTAR,4HT   ,4H    /
      DATA LSTOP /4HSTOP,4H    ,4H    /
C-----NERR IS FILE DEVICE NUMBER FOR ERROR MESSAGES.
C-----MODE SELECTS INTERACTIVE OR BATCH MODE OF OPERATION.
C     MODE=1 FOR BATCH PROCESS OR MODE .NE. 1 FOR INTERACTIVE PROCESS.
      DATA NERR/6/,MODE/2/
      DATA NTIME/0/
      END
      SUBROUTINE SECOND(T)
      IMPLICIT REAL*8 (A-H,O-Z)
```

```
      COMMON /ERROR/ NERR,MODE,NTIME
C-----ROUTINE TO COMPUTE ELAPSED CPU TIME IN SECONDS.
C-----THIS ROUTINE WILL HAVE TO BE CHANGED TO WORK ON OTHER MACHINES.
C-----STATEMENTS FOR CP/CMS AT NPS, MONTEREY, CA.
      DIMENSION ITIME(6)
      CALL IXCLOK(ITIME)
      T = DFLOAT(ITIME(6))/768.0D+2
C
C-----STANDARD IBM 360 TIMER PROGRAM.
C     CALL GETIME(IET)
C     T = DFLOAT(IET)*2.6D-5
C
      RETURN
      END
      SUBROUTINE MOVEB(A,I,B,J)
C
C-----THIS SUBROUTINE MOVES THE JTH BYTE OF VARIABLE B INTO THE
C-----ITH BYTE OF VARIABLE A.  IT MAY HAVE TO BE REWRITTEN TO
C-----WORK ON OTHER MACHINES.
C
      LOGICAL*1 A(1),B(1)
      A(I) = B(J)
      RETURN
      END
      SUBROUTINE INPUT(IERR)
      IMPLICIT REAL*8 (A-H,O-Z)
      LOGICAL INDIC
      COMMON MTOT,NDP,L(1)
      COMMON /LOOPD/ LA,IN,INDEX(5),LSTART(5),IJKLM(5),NW
      COMMON /PSIZE/ NO,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR,INDIC
      COMMON /ADIR/ NA,I(10)
      COMMON /NAMES/ LOOP(3),NEXT(3),LSKIP(3),LNAME(3)
      COMMON /ERROR/ NERR,MODE,NTIME
      IF(INDIC)CALL SECOND(TO)
      INDIC = .FALSE.
  100 CALL SECOND(T)
      TD=T-TO
      IF(NO.EQ.0 .AND. NTIME.EQ.0)WRITE(NWRITE,2000) TD
      TO=T
      IF(LA.EQ.0) CALL OPREAD(IERR)
      IF(LA.EQ.0 .OR. IERR .NE. 1) GO TO 175
      CALL FIND(LNAME,IERR)
      IF(IERR .NE. 1)RETURN
  150 CALL OPSTOR(L(NA),NW,IERR)
      IF(IERR .NE. 1) RETURN
      IF(LA.EQ.0) GO TO 100
  175 RETURN
 2000 FORMAT(1X,30(1H-),F10.3,8H SECONDS )
      END
```

```
      SUBROUTINE OPREAD(IERR)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON MTOT,NDP,L(1)
      COMMON /PSIZE/ NO,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR
      COMMON /CARD/ INHOL(3,10),NN(4),S1,S2
      COMMON /LOOPD/ LA,IN,INDEX(5),LSTART(5),IJKLM(5),NW
      COMMON /NAMES/ LOOP(3),NEXT(3),LSKIP(3),LNAME(3)
      LOGICAL COMP
C-----READ AND PRINT ALL INPUT OPERATION CARDS-----------------
      N=NSIZE+NDIR
      IN=0
  100 CALL RCARD(IERR)
      IF(IERR .NE. 1) RETURN
      IF(COMP(INHOL(1,1),LOOP)) IN=IN+1
      IF((IN.EQ.1).AND.(LA.EQ.0)) LA=1
      IF(LA.EQ.0) RETURN
C-----STORE ALL OPERATIONS WITHIN OUTER LOOP-----------
      DO 200 J=1,10
      DO 200 I=1,NH
      L(N)=INHOL(I,J)
 200  N=N+1
      DO 210 I=1,4
      L(N)=NN(I)
 210  N=N+1
      IF(COMP(INHOL(1,1),NEXT)) IN=IN-1
      IF(IN.EQ.0) GO TO 250
      LA=LA+1
      GO TO 100
C     SET NAME AND STORAGE FOR LOOP DATA
  250 CALL LIST(LNAME,NW,LA,NSP,IERR)
      IF(IERR .NE. 1) RETURN
  300 LA=1
      IN=0
      RETURN
      END
      SUBROUTINE RCARD(IERR)
C
C-----RCARD READS AND INTERPRETES AN OPERATION
C
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON MTOT,NDP,L(1)
      COMMON /PSIZE/ NO,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR
      COMMON /CARD/ INHOL(3,10),NN(4),S1,S2
      COMMON /RECON/ IN(80),INDIGT(80),ITAB(5,4),NUM(10),LBYTE
      DATA IBLANK/2H  /,IALPHA/2HA /,IZULU/2HZ /,ICOMMA/2H, /
C-----READ OP,M1,M2,M3,M4,N1,N2,N3,N4
   10 READ(NREAD,1000) IN
      IF(NO.EQ.0) WRITE(NWRITE,2000) IN
 1000 FORMAT(80A1)
```

```
 2000 FORMAT(3H **,80A1)
C-----SET INITIAL CONDITIONS
      DO 50 J=1,10
      DO 50 I = 1,NH
   50 INHOL(I,J) = IBLANK
      DO 60 J = 1,4
   60 NN(J) = 0
      IER = 0
      IEND = 80
      IROW = 1
      J = 1
      II = 1
      JNUM = 1
      IBYTE = 0
      JJ = 1
C-----IDENTIFY CHARACTER INDEX AND STORE IN INDIGT
      DO 90 K = 1,80
      IJ = IN(K)
      KTYPE = 400
      IF(IJ.EQ.IBLANK)GO TO 100
      DO 70 I = 1,10
      IF(IJ.NE.NUM(I))GO TO 70
      KTYPE  = 100 + I
      GO TO 90
   70 CONTINUE
      IF(IJ.NE.ICOMMA)GO TO 80
      KTYPE = 200
      GO TO 90
   80 IF(IJ.GE.IALPHA .AND. IJ.LE.IZULU)KTYPE = 300
   90 INDIGT(K) = KTYPE
      GO TO 120
  100 IEND = K - 1
C-----INTERPRET CHARACTER AND TRANSFER TO VARIABLE IN CARD COMMON
  120 ICOL = INDIGT(J)/100
      ISTAT = ITAB(IROW,ICOL)
C-----SEPARATE ACTION DIGIT FROM STATE DIGIT
      IACT = ISTAT/100
      IROW = ISTAT - 100*IACT
      GO TO (200,300,400,500,700),IACT
C-----HOLLERITH DATA
  200 IBYTE = IBYTE + 1
      IF(IBYTE.LE.LBYTE)GO TO 210
      IBYTE = 1
      II = II + 1
  210 CALL MOVEB(INHOL(II,JJ),IBYTE,IN(J),1)
      GO TO 600
C-----NUMERICAL DATA
  300 NNN = MOD(INDIGT(J),10)
      NN(JNUM)=10*NN(JNUM)+NNN
```

```
      GO TO 600
C-----COMMA AFTER HOLLERITH DATA
  400 JJ=JJ+1
      II = 1
      IBYTE = 0
      IF(JJ.GT.10 .AND. J.LT.IEND .AND. INDIGT(J+1).GT.200) GO TO 700
      GO TO 600
C-----COMMA AFTER NUMERICAL DATA
  500 JNUM = JNUM + 1
      IF(JNUM.GT.4 .AND. J.LT.IEND) GO TO 700
  600 J = J + 1
      IF(J.LE.IEND) GO TO 120
      GO TO 710
  700 WRITE(NWRITE,3000)
 3000 FORMAT(29H0  INPUT CARD ERROR, RE-ENTER)
      IERR = 2
  710 RETURN
      END
      SUBROUTINE OPSTOR(LOOPA,NW,IERR)
      IMPLICIT REAL*8 (A-H,O-Z)
      LOGICAL COMP
      COMMON MTOT,NDP,L(1)
      COMMON /PSIZE/ NO,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR
      COMMON /CARD/ INHOL(3,10),NN(4),S1,S2
      COMMON /LOOPD/ LA,IN,INDEX(5),LSTART(5),IJKLM(5)
      COMMON /ADIR/NA,III(10)
      COMMON /NAMES/ LOOP(3),NEXT(3),LSKIP(3),LNAME(3)
      DIMENSION LOOPA(NW,1)
      DATA IBLANK/4H    /
      IF(LA.NE.1) GO TO 50
      DO 40 I=1,5
   40 IJKLM(I)=0
      IF(NO.EQ.0) WRITE(NWRITE,2001)
C-----TRANSFER OPERATION FROM STORAGE TO CARD COMMON   ----------
   50 I=1
      DO 100 J=1,10
      DO 100 K=1,NH
      INHOL(K,J)=LOOPA(I,LA)
  100 I=I+1
      DO 110 J=1,4
      NN(J)=LOOPA(I,LA)
  110 I=I+1
      LA=LA+1
      IF(NO.EQ.0) WRITE(NWRITE,2000) ((INHOL(I,J),I=1,NH),J=1,10)
     1 ,(IJKLM(I),I=1,IN)
      IF(COMP(INHOL(1,1),LSKIP)) GO TO 200
      IF(COMP(INHOL(1,1),LOOP)) GO TO 300
      IF(COMP(INHOL(1,1),NEXT)) GO TO 400
      RETURN
```

```
C-----SKIP OPERATION- CHECK SIGN AND INCREMENT COUNTER------
  200 CALL FIND(INHOL(1,2),IERR)
      IF(IERR .NE. 1) RETURN
  250 CALL MOVE(L(NA),S1,NDP)
      IF(NO.EQ.0) WRITE(NWRITE,2002) (INHOL(J,2),J=1,NH),S1,NN(1)
      IF(S1.LT.0.0D0) LA=LA+NN(1)
      GO TO 50
C-----LOOP OPERATION- SET INDEX LIMIT FOR LOOP -----------
  300 IN=IN+1
      INDEX(IN)=NN(1)
      IJKLM(IN)=1
      LSTART(IN)=LA
      GO TO 50
C-----NEXT OPERATION- INCREMENT INDEX AND CHECK FOR LAST OPERATION---
  400 INDEX(IN)=INDEX(IN)-1
      IJKLM(IN)=IJKLM(IN)+1
      IF(NO.EQ.0)  WRITE(NWRITE,2003) IN
      IF(INHOL(1,2).EQ.IBLANK) GO TO 450
      CALL FIND(INHOL(1,2),IERR)
      IF(IERR .NE. 1) RETURN
  410 CALL MOVE(L(NA),S1,NDP)
      IF(NO.EQ.0) WRITE(NWRITE,2002) (INHOL(I,2),I=1,NH),S1
      IF(S1.LT.0.0D0)  INDEX(IN)=0
  450 IF(INDEX(IN).NE.0)  LA=LSTART(IN)
      IF(INDEX(IN).EQ.0)  IJKLM(IN)=0
      IF(INDEX(IN).EQ.0)  IN=IN-1
      IF(IN.EQ.0)  GO TO 500
      GO TO 50
C-----DELETE ARRAY OF OPERATIONS WITHIN LOOPS--------
  500 LA=0
      CALL DELETE(LNAME)
      RETURN
 2000 FORMAT(5H  *** 20A4,/,1H ,I5,3H=L1,I5,3H=L2,I5,3H=L3,I5,3H=L4,I5,3
     1 H=L5)
 2001 FORMAT(28H0START OF LOOPING OPERATIONS  )
 2002 FORMAT (1X,2A4,3H = E15.7,17H IF NEGATIVE SKIP   I5)
 2003 FORMAT(25H LAST OPERATION IN LOOP L ,I1/1H0)
      END
      SUBROUTINE ERCOM(IERR)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /LOOPD/ LA
      COMMON /ERROR/ NERR,MODE,NTIME
      WRITE(NERR,2000)
      IF(LA .GE. 1) WRITE(NERR,3000)
      IERR = 2
      RETURN
 2000 FORMAT  (24H0MATRICES NOT COMPATIBLE )
 3000 FORMAT  (39H0**WARNING  LOOP OPERATIONS CANCELLED**)
      END
```

```
      SUBROUTINE MOVE(II,JJ,N)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION II(3),JJ(3)
      DO 100 I=1,N
  100 JJ(I)=II(I)
      RETURN
      END
      FUNCTION COMP(I,J)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION I(3),J(3)
      LOGICAL COMP
      COMMON /PSIZE/ NO,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR
      COMP=.FALSE.
      DO 100 K=1,NH
      IF(I(K).NE.J(K)) RETURN
  100 CONTINUE
      COMP=.TRUE.
      RETURN
      END
      SUBROUTINE LIST(II,NR,NC,NP,IERR)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON MTOT,NDP,L(1)
      COMMON /PSIZE/ NO,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR
      COMMON /ADIR/ NA,I(10)
      COMMON /ERROR/ NERR,MODE,NTIME
      DIMENSION II(3),IBLANK(3)
      LOGICAL COMP
      DATA IBLANK/3*4H    /
      IF(COMP(II,IBLANK)) GO TO 150
      IF(NR.LT.1 .OR. NC.LT.1)GO TO 150
      IF(NO.EQ.0) WRITE(NWRITE,2001) NR,NC
      NEL = NR*NC
      I(1) = NR*NC
      IF(NP.EQ.1 .AND. MOD(NR*NC,NDP).EQ.1) I(1)=NR*NC+1
      I(2)=NP
      I(3)=NC
      I(4)=NR
      CALL MOVE(II,I(5),NH)
C-----CHECK CAPACITY OF L ARRAY
      NN=I(1)*I(2)+NDIR
      IF(NN.LT.MTOT-NSIZE) GO TO 50
      NX=NSIZE+NN
      WRITE(NERR,2000)MTOT,NX
      IERR = 2
  999 RETURN
C-----SET DIRECTORY AND RESERVE STORAGE FOR NEW ARRAY------
   50 DO 100 N=1,NDIR
      IN=NSIZE-1+N
  100 L(IN)=I(N)
```

```
      NA=NSIZE+NDIR
      NARA=NARA+1
      NSIZE=NSIZE+NN
      RETURN
  150 WRITE(NERR,2002)II
      IERR = 2
      RETURN
 2000 FORMAT (19H0STORAGE EXCEEDED-- I5,11H RESERVED-- I5,9H REQUIRED)
 2001 FORMAT(I5,6H ROWS ,I5,8H COLUMNS )
 2002 FORMAT(11H0  MATRIX $,3A4,25H$ ARGUMENT ERROR.  REPEAT )
      END
      SUBROUTINE FIND(II,IERR)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON MTOT,NDP,L(1)
      COMMON /PSIZE/ NO,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR
      COMMON /ADIR/ NA,I(10)
      COMMON /ERROR/ NERR,MODE,NTIME
      DIMENSION II(3)
      LOGICAL COMP
C-----LOCATE ARRAY AND ITS DIRECTORY-- NA=0 IF ARRAY II IS NOT FOUND
      N=1
      NA=0
  100 IF(N.EQ.NSIZE) GO TO 400
      IF(COMP(II,L(N+4))) GO TO 200
      N=N+L(N)*L(N+1)+NDIR
      GO TO 100
C-----COMPUTE ADDRESS AND TRANSFER DIRECTORY
  200 DO 300 K=1,NDIR
      M=N+K-1
  300 I(K)=L(M)
      NA=N+1
      RETURN
  400 WRITE(NERR,2000)  (II(J),J=1,NH)
      IERR = 2
  999 RETURN
 2000 FORMAT ( 7H0ARRAY$ 2A4,23H$NOT PREVIOUSLY DEFINED )
      END
      SUBROUTINE DELETE(II)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON MTOT,NDP,L(1)
      COMMON /PSIZE/ NO,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR
      COMMON /ADIR/ NA,I(10)
      DIMENSION II(3)
      LOGICAL COMP
C-----LOCATE ARRAY TO BE DELETED
      N=1
      NA=0
   10 IF(N.EQ.NSIZE) RETURN
      IF(COMP(II,L(N+4))) GO TO 20
```

```
      N=N+L(N)*L(N+1)+NDIR                                              ST004560
      GO TO 10                                                          ST004570
C-----COMPUTE ADDRESS AND TRANSFER DIRECTORY                            ST004580
   20 DO 30  K=1,NDIR                                                   ST004590
      M=N+K-1                                                           ST004600
   30 I(K)=L(M)                                                         ST004610
      NA=N+1                                                            ST004620
C-----DELETE ARRAY AND RELOCATE REMAINING ARRAYS                        ST004630
      NARA=NARA-1                                                       ST004640
      NN=NA+I(1)*I(2)                                                   ST004650
      NL=NSIZE-1                                                        ST004660
      NSH=NDIR+I(1)*I(2)                                                ST004670
      IF(NN.EQ.NSIZE) GO TO 200                                         ST004680
      DO 100 N=NN,NL                                                    ST004690
  100 L(N-NSH)=L(N)                                                     ST004700
  200 NSIZE=NSIZE-NSH                                                   ST004710
      IF(NO.EQ.0) WRITE(NWRITE,2000) (II(J),J=1,NH)                     ST004720
      RETURN                                                            ST004730
 2000 FORMAT (8H -ARRAY$ 2A4,8H$DELETED )                               ST004740
      END                                                               ST004750
      SUBROUTINE GROUP1(LAST,IERR)                                      ST004760
      IMPLICIT REAL*8 (A-H,O-Z)                                         ST004770
      COMMON MTOT,NDP,L(1)                                              ST004780
      COMMON /CARD/ INHOL(3,10),N1,N2,N3,N4,S1,S2                       ST004790
      COMMON /PSIZE/ NO,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR,INDIC       ST004800
      COMMON /ERROR/ NERR,MODE,NTIME                                    ST004810
      COMMON /PSAVE/ NUMNP,NEQ,NPROP,NUMEL,NTRUSS,NT,NSAVE              ST004820
      COMMON /STR/ ST(9,16),XM(16),S(16,16),ND,NS,NRP,LM(16)            ST004830
      LOGICAL COMP,INDIC                                                ST004840
      DIMENSION IOP(3,34)                                               ST004850
      DATA IOP(1,1), IOP(2,1), IOP(3,1) /4HSTAR,4HT   ,4H    /          ST004860
      DATA IOP(1,2), IOP(2,2), IOP(3,2) /4HLOAD,4H    ,4H    /          ST004870
      DATA IOP(1,3), IOP(2,3), IOP(3,3) /4HPRIN,4HT   ,4H    /          ST004880
      DATA IOP(1,4), IOP(2,4), IOP(3,4) /4HZERO,4H    ,4H    /          ST004890
      DATA IOP(1,5), IOP(2,5), IOP(3,5) /4HDELE,4HTE  ,4H    /          ST004900
      DATA IOP(1,6), IOP(2,6), IOP(3,6) /4HDUP ,4H    ,4H    /          ST004910
      DATA IOP(1,7), IOP(2,7), IOP(3,7) /4HADD ,4H    ,4H    /          ST004920
      DATA IOP(1,8), IOP(2,8), IOP(3,8) /4HSUB ,4H    ,4H    /          ST004930
      DATA IOP(1,9), IOP(2,9), IOP(3,9) /4HNO  ,4H    ,4H    /          ST004940
      DATA IOP(1,10),IOP(2,10),IOP(3,10) /4HYES ,4H    ,4H    /         ST004950
      DATA IOP(1,11),IOP(2,11),IOP(3,11) /4HSTOP,4H    ,4H    /         ST004960
      DATA IOP(1,12),IOP(2,12),IOP(3,12) /4HMULT,4H    ,4H    /         ST004970
      DATA IOP(1,13),IOP(2,13),IOP(3,13) /4HTRAN,4H    ,4H    /         ST004980
      DATA IOP(1,14),IOP(2,14),IOP(3,14) /4HINVE,4HL   ,4H    /         ST004990
      DATA IOP(1,15),IOP(2,15),IOP(3,15) /4HSQRE,4HL   ,4H    /         ST005000
      DATA IOP(1,16),IOP(2,16),IOP(3,16) /4HDLOG,4H    ,4H    /         ST005010
      DATA IOP(1,17),IOP(2,17),IOP(3,17) /4HSCAL,4HE   ,4H    /         ST005020
      DATA IOP(1,18),IOP(2,18),IOP(3,18) /4HSOLV,4HE   ,4H    /         ST005030
      DATA IOP(1,19),IOP(2,19),IOP(3,19) /4HUSER,4HA   ,4H    /         ST005040
```

```
      DATA IOP(1,20),IOP(2,20),IOP(3,20) /4HUSER,4HB   ,4H    /          ST005050
      DATA IOP(1,21),IOP(2,21),IOP(3,21) /4HMAX ,4H    ,4H    /          ST005060
      DATA IOP(1,22),IOP(2,22),IOP(3,22) /4HNORM,4H    ,4H    /          ST005070
      DATA IOP(1,23),IOP(2,23),IOP(3,23) /4HPROD,4H    ,4H    /          ST005080
      DATA IOP(1,24),IOP(2,24),IOP(3,24) /4HDUPS,4HM   ,4H    /          ST005090
      DATA IOP(1,25),IOP(2,25),IOP(3,25) /4HSTOS,4HM   ,4H    /          ST005100
      DATA IOP(1,26),IOP(2,26),IOP(3,26) /4HDUPD,4HG   ,4H    /          ST005110
      DATA IOP(1,27),IOP(2,27),IOP(3,27) /4HSTOD,4HG   ,4H    /          ST005120
      DATA IOP(1,28),IOP(2,28),IOP(3,28) /4HLABE,4HL   ,4H    /          ST005130
      DATA IOP(1,29),IOP(2,29),IOP(3,29)/4HREAD,4H    ,4H    /           ST005140
      DATA IOP(1,30),IOP(2,30),IOP(3,30) /4HWRIT,4HE   ,4H    /          ST005150
      DATA IOP(1,31),IOP(2,31),IOP(3,31) /4HTIME,4H    ,4H    /          ST005160
      DATA IOP(1,32),IOP(2,32),IOP(3,32) /4HSAVE,4H    ,4H    /          ST005170
      DATA IOP(1,33),IOP(2,33),IOP(3,33) /4HRESU,4HME  ,4H    /          ST005180
      DATA IOP(1,34),IOP(2,34),IOP(3,34) /4HLIST,4H    ,4H    /          ST005190
      NUMOP=34                                                           ST005200
      GO TO 175                                                          ST005210
C---- READ OPERATION FROM CARD OR STORAGE -----------------               ST005220
  100 IF (IERR .GT. 1) RETURN                                            ST005230
      CALL INPUT(IERR)                                                   ST005240
      IF(IERR .GT. 1) RETURN                                             ST005250
C-----INTERPRETE OPERATION ----------------------------------             ST005260
  175 DO 200 J=1,NUMOP                                                   ST005270
      N=J                                                                ST005280
      IF (COMP(INHOL(1,1),IOP(1,J))) GO TO 300                           ST005290
  200 CONTINUE                                                           ST005300
      RETURN                                                             ST005310
C-----EXECUTE APPROPRIATE OPERATION ---------------------                 ST005320
  300 LAST=1                                                             ST005330
      GO TO (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,   ST005340
     1 23,24,25,26,27,28,29,30,31,32,33,34),N                            ST005350
    1 NARG=0                                                             ST005360
      NSIZE=1                                                            ST005370
      NO=0                                                               ST005380
      WRITE(NWRITE,2000)                                                 ST005390
      GO TO 100                                                          ST005400
    2 CALL LOAD (IERR)                                                   ST005410
      GO TO 100                                                          ST005420
    3 IF(N1.GT.0) CALL LABEL(N1)                                         ST005430
      IF(N3.EQ.0) CALL PRINT(IERR)                                       ST005440
      GO TO 100                                                          ST005450
    4 CALL ZERO (IERR)                                                   ST005460
      GO TO 100                                                          ST005470
    5 CALL DELETE(INHOL(1,2))                                            ST005480
      GO TO 100                                                          ST005490
    6 CALL DUP  (IERR)                                                   ST005500
      GO TO 100                                                          ST005510
    7 CALL ADD(1.D0,IERR)                                                ST005520
      GO TO 100                                                          ST005530
```

```
 8 CALL ADD(-1.D0,IERR)                                   ST005540
   GO TO 100                                              ST005550
 9 NO=1                                                   ST005560
   GO TO 100                                              ST005570
10 NO=0                                                   ST005580
   GO TO 100                                              ST005590
11 IF(NRP.GT.1)END FILE NT                                ST005600
   STOP                                                   ST005610
12 CALL MUL  (IERR)                                       ST005620
   GO TO 100                                              ST005630
13 CALL TRAN(IERR)                                        ST005640
   GO TO 100                                              ST005650
14 CALL MOP(1,IERR)                                       ST005660
   GO TO 100                                              ST005670
15 CALL MOP(2,IERR)                                       ST005680
   GO TO 100                                              ST005690
16 CALL MOP(3,IERR)                                       ST005700
   GO TO 100                                              ST005710
17 CALL MOP(4,IERR)                                       ST005720
   GO TO 100                                              ST005730
18 CALL SYMSV(IERR)                                       ST005740
   GO TO 100                                              ST005750
19 CALL USERA(IERR)                                       ST005760
   GO TO 100                                              ST005770
20 CALL USERB(IERR)                                       ST005780
   GO TO 100                                              ST005790
21 CALL MOP(5,IERR)                                       ST005800
   GO TO 100                                              ST005810
22 CALL MOP(6,IERR)                                       ST005820
   GO TO 100                                              ST005830
23 CALL MOP(7,IERR)                                       ST005840
   GO TO 100                                              ST005850
24 CALL SMOP(1,IERR)                                      ST005860
   GO TO 100                                              ST005870
25 CALL SMOP(2,IERR)                                      ST005880
   GO TO 100                                              ST005890
26 CALL DGOP(1,IERR)                                      ST005900
   GO TO 100                                              ST005910
27 CALL DGOP(2,IERR)                                      ST005920
   GO TO 100                                              ST005930
28 IF(N1.GT.0) CALL LABEL(N1)                             ST005940
   GO TO 100                                              ST005950
29 IF(MODE .NE. 1) NREAD=N1                               ST005960
   IF(MODE .EQ. 1) WRITE(NERR,2001)                       ST005970
   GO TO 100                                              ST005980
30 IF(MODE .NE. 1) NWRITE=N1                              ST005990
   IF(MODE .EQ. 1) WRITE(NERR,2001)                       ST006000
   GO TO 100                                              ST006010
31 NTIME = 1 - NTIME                                      ST006020
```

```
      GO TO 100                                                         ST006030
   32 CALL SAVE (NARA,NSIZE,L,MTOT,1)                                   ST006040
      WRITE (NWRITE,2005) NARA,NSIZE,MTOT                               ST006050
      J=1                                                               ST006060
      GO TO 400                                                         ST006070
   33 CALL SAVE(NARA,NSIZE,L,MTOT,2)                                    ST006080
   34 J=1                                                               ST006090
      WRITE(NWRITE,2003)NARA,NSIZE,MTOT                                 ST006100
  400 IF(J.EQ.NSIZE)GO TO 100                                           ST006110
      K=J+4                                                             ST006120
      KJ=J+NH+3                                                         ST006130
      WRITE(NWRITE,2004) (L(KK),KK=K,KJ),L(J+3),L(J+2)                  ST006140
      J=J+L(J)*L(J+1)+NDIR                                              ST006150
      GO TO 400                                                         ST006160
 2000 FORMAT (1H1)                                                      ST006170
 2001 FORMAT (22H0 COMMAND NOT EXECUTED)                                ST006180
 2003 FORMAT(11H0 THERE ARE,I4,19H ARRAYS IN STORAGE.,I7,               ST006190
     1 36H STORAGE POSITIONS HAVE BEEN USED OF,I7,10H RESERVED.)        ST006200
 2004 FORMAT (8H ARRAY $,2A4,5H$ HAS,I6,9H ROWS AND,I6,9H COLUMNS.)     ST006210
 2005 FORMAT (1H0,I4,47H ARRAYS HAVE BEEN SAVED.  THESE ARRAYS OCCUPIED,ST006220
     1 I7,25H STORAGE POSITIONS OF THE,I7,20H POSITIONS RESERVED.)      ST006230
      END                                                               ST006240
      SUBROUTINE SAVB(I,J,K,L,M)                                        ST006250
      COMMON /PSAVE/ NUMNP,NEQ,NPROP,NUMEL,NTRUSS,NT,NSAVE              ST006260
      COMMON /STR/ ST(9,16),XM(16),S(16,16),ND,NS,NRP,LM(16)            ST006270
      DIMENSION K(L)                                                    ST006280
      REWIND NSAVE                                                      ST006290
      GO TO (1,2),M                                                     ST006300
    1 WRITE (NSAVE) I,J,NUMNP,NEQ,NPROP,NUMEL,NTRUSS,NRP,K              ST006310
      END FILE NSAVE                                                    ST006320
      RETURN                                                            ST006330
    2 READ (NSAVE) I,J,NUMNP,NEQ,NPROP,NUMEL,NTRUSS,NRP,K               ST006340
      RETURN                                                            ST006350
      END                                                               ST006360
      SUBROUTINE LOAD(IERR)                                             ST006370
      IMPLICIT REAL*8 (A-H,O-Z)                                         ST006380
      COMMON MTOT,NDP,L(1)                                              ST006390
      COMMON /PSIZE/ NO,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR             ST006400
      COMMON /CARD/ INHOL(3,10),N1,N2,N3,N4,S1,S2                       ST006410
      COMMON /ADIR/ NA,I(10)                                            ST006420
      CALL DELETE(INHOL(1,2))                                           ST006430
      CALL LIST(INHOL(1,2),N1,N2,NDP,IERR)                              ST006440
      IF(IERR .GT. 1) RETURN                                            ST006450
  100 CALL RLOAD(L(NA),N1,N2,N3)                                        ST006460
  999 RETURN                                                            ST006470
      END                                                               ST006480
      SUBROUTINE RLOAD(A,NR,NC,N3)                                      ST006490
      IMPLICIT REAL*8 (A-H,O-Z)                                         ST006500
      COMMON /PSIZE/ NO,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR             ST006510
```

```
      COMMON /TEMP/ FOR(40)
      DIMENSION A(NR,1)
      IF(N3.NE.0) GO TO 100
      READ(NREAD,1000) ((A(I,J),J=1,NC),I=1,NR)
      RETURN
  100 READ(NREAD,1001) FOR
      READ(NREAD,FOR) ((A(I,J),J=1,NC),I=1,NR)
      RETURN
 1000 FORMAT (8F10.0)
 1001 FORMAT(40A2)
      END
      SUBROUTINE PRINT(IERR)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON MTOT,NDP,L(1)
      COMMON /PSIZE/ NO,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR
      COMMON /CARD/ INHOL(3,10),N1,N2,N3,N4,S1,S2
      COMMON /ADIR/ NA,I(10)
      CALL FIND(INHOL(1,2),IERR)
      IF(IERR .GT. 1) RETURN
  100 NR=I(4)
      CALL RPRT(L(NA),NR,I(3),N2)
  999 RETURN
      END
      SUBROUTINE RPRT(A,NR,NC,N2)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /PSIZE/ NO,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR
      DIMENSION A(NR,1)
      IF(N2.LT.1 .OR. N2.GT.8) N2=8
      DO 100  I=1,NC,N2
      WRITE(NWRITE,2002)
      IH = MINO(I+N2-1,NC)
      WRITE(NWRITE,2000)  (K,K=I,IH)
      DO 100 J=1,NR
  100 WRITE(NWRITE,2001)  (J,(A(J,K),K=I,IH))
      RETURN
 2000 FORMAT (1H ,5X,8(I8,7X))
 2001 FORMAT (1H ,I5,8(1PD15.7))
 2002 FORMAT (1H0)
      END
      SUBROUTINE LABEL(N1)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /PSIZE/ NO,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR
      COMMON /TEMP/ II(40)
      DO 100 I=1,N1
      READ(NREAD,1000)II
      IF(NO.EQ.0) WRITE(NWRITE,1000)  II
  100 CONTINUE
      RETURN
 1000 FORMAT (40A2)
```

```
      END                                                               ST007010
      SUBROUTINE ZERO(IERR)                                             ST007020
      IMPLICIT REAL*8 (A-H,O-Z)                                         ST007030
      COMMON MTOT,NDP,L(1)                                              ST007040
      COMMON /PSIZE/ NO,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR             ST007050
      COMMON /CARD/ INHOL(3,10),N1,N2,N3,N4,S1,S2                       ST007060
      COMMON /ADIR/ NA,I(10)                                            ST007070
      S1=N3                                                             ST007080
      S2=N4                                                             ST007090
      CALL DELETE(INHOL(1,2))                                           ST007100
      CALL LIST(INHOL(1,2),N1,N2,NDP,IERR)                              ST007110
      IF(IERR .GT. 1) RETURN                                            ST007120
  100 CALL ZEROS(L(NA))                                                 ST007130
  999 RETURN                                                            ST007140
      END                                                               ST007150
      SUBROUTINE ZEROS(A)                                               ST007160
      IMPLICIT REAL*8 (A-H,O-Z)                                         ST007170
      COMMON /CARD/ INHOL(3,10),N1,N2,N3,N4,S1,S2                       ST007180
      DIMENSION A(1)                                                    ST007190
      NEL=N1*N2                                                         ST007200
      DO 100 II = 1,NEL                                                 ST007210
  100 A(II) = S2                                                        ST007220
      L=1                                                               ST007230
      NN = MIN0 (N1,N2)                                                 ST007240
      DO 200 JJ = 1,NN                                                  ST007250
      A(L) = S1                                                         ST007260
  200 L = L + N1 + 1                                                    ST007270
      RETURN                                                            ST007280
      END                                                               ST007290
      SUBROUTINE DUP(IERR)                                              ST007300
      IMPLICIT REAL*8 (A-H,O-Z)                                         ST007310
      COMMON MTOT,NDP,L(1)                                              ST007320
      COMMON /PSIZE/ NO,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR             ST007330
      COMMON /CARD/ INHOL(3,10),N1,N2,N3,N4,S1,S2                       ST007340
      COMMON /ADIR/ NA,I(10)                                            ST007350
      CALL DELETE(INHOL(1,3))                                           ST007360
      CALL FIND (INHOL(1,2),IERR)                                       ST007370
      IF(IERR .GT. 1) RETURN                                            ST007380
  100 NR=I(4)                                                           ST007390
      NC=I(3)                                                           ST007400
      NB=NA                                                             ST007410
      CALL LIST(INHOL(1,3),NR,NC,NDP,IERR)                              ST007420
      IF(IERR .GT. 1) RETURN                                            ST007430
      NEL = NR*NC                                                       ST007440
  200 CALL DUPL(L(NB),L(NA),NEL)                                        ST007450
  999 RETURN                                                            ST007460
      END                                                               ST007470
      SUBROUTINE DUPL(A,B,NEL)                                          ST007480
      IMPLICIT REAL*8 (A-H,O-Z)                                         ST007490
```

```
      DIMENSION A(1),B(1)
      DO 100 I=1,NEL
  100 B(I)=A(I)
      RETURN
      END
      SUBROUTINE DGOP(N,IERR)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON MTOT,NDP,L(1)
      COMMON /PSIZE/ NO,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR
      COMMON /CARD/ INHOL(3,10),N1,N2,N3,N4,S1,S2
      COMMON /ADIR/ NA,I(10)
C-----SUBROUTINE TO DUPLICATE OR STORE DIAGONAL-------------
      IF(N.EQ.1) CALL DELETE(INHOL(1,3))
      CALL FIND(INHOL(1,2),IERR)
      IF(IERR .GT. 1) RETURN
   10 N1=I(3)
      IF(I(3).NE.I(4)) CALL ERCOM(IERR)
      IF(IERR .GT. 1) RETURN
   20 N2=NA
      IF(N.NE.1) GO TO 100
      CALL LIST(INHOL(1,3),1,N1,NDP,IERR)
      IF(IERR .GT. 1) RETURN
  100 CALL FIND(INHOL(1,3),IERR)
      IF(IERR .GT. 1) RETURN
  110 N5=I(3)*I(4)
      IF(N5.NE.N1) CALL ERCOM(IERR)
      IF(IERR .GT. 1) RETURN
  200 N3=NA
      CALL DGOPS(L(N2),L(N3),N1,N)
  999 RETURN
      END
      SUBROUTINE DGOPS(A,B,M,N)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(1),B(1)
      DO 300 I=1,M
      IA = M*(I-1) + I
      GO TO (1,2),N
    1 B(I)=A(IA)
      GO TO 300
    2 A(IA)=B(I)
  300 CONTINUE
      RETURN
      END
      SUBROUTINE MOP(N,IERR)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON MTOT,NDP,L(1)
      COMMON /PSIZE/ NO,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR
      COMMON /CARD/ INHOL(3,10),N1,N2,N3,N4,S1,S2
      COMMON /ADIR/ NA,I(10)
```

```
      IF(N.GT.4)  CALL DELETE(INHOL(1,3))
      CALL FIND(INHOL(1,2),IERR)
      IF(IERR .GT. 1)  RETURN
  100 N3=NA
      NC=L(3)
      NR=L(4)
      IF(N.EQ.4)CALL FIND(INHOL(1,3),IERR)
      IF(N.EQ.5)  CALL LIST(INHOL(1,3),NR,1,NDP,IERR)
      IF(N.EQ.6)  CALL LIST(INHOL(1,3),1,NC,NDP,IERR)
      IF(N.EQ.7)  CALL LIST(INHOL(1,3),1,2 ,NDP,IERR)
      IF(IERR .GT. 1) RETURN
      CALL MOPS(L(N3),L(NA),NR,NC,N,N1)
  999 RETURN
      END
      SUBROUTINE MOPS(A,B,NR,NC,N,N1)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON MTOT,NDP,L(1)
      COMMON /PSIZE/ NO,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR
      DIMENSION A(1),B(1)
      NEL = NR*NC
      GO TO  (100,200,300,400,500,600,700),N
C-----REPLACE EACH ELEMENT WITH ITS INVERSE------
  100 DO 150 I=1,NEL
  150 A(I)=1.0D0/A(I)
      RETURN
C-----REPLACE EACH ELEMENT WITH ITS SQUARE ROOT---
  200 DO 250 I=1,NEL
  250 A(I)=DSQRT(A(I))
      RETURN
C-----REPLACE EACH ELEMENT WITH ITS LOG-----------
  300 DO 350 I=1,NEL
  350 A(I)=DLOG(A(I))
      RETURN
C-----REPLACE EACH ELEMENT WITH ITS SELF TIMES A SCALAR
  400 DO 450 I=1,NEL
  450 A(I)=A(I)*B(1)
      RETURN
C-----EVALUATE THE MAXIMUM OF EACH ROW------------
  500 DO 550 I=1,NR
      B(I)=0.0D0
      DO 540 J=1,NC
      IA = NR*(J-1) + I
      X=DABS(A(IA))
      IF(X.LT.B(I))  GO TO 540
      B(I)=X
      JMAX=J
  540 CONTINUE
      IF(NO.EQ.0)  WRITE(NWRITE,2000)  JMAX,B(I)
 2000 FORMAT  (I8,1PD16.7)
```

```
  550 CONTINUE
      RETURN
C-----EVALUATE COLUMN NORMS-----------------------
  600 DO 650 J=1,NC
      B(J)=0.0D0
      DO 640 I=1,NR
      IA = NR*(J-1) + I
      IF(N1.GT.0) GO TO 630
      B(J)=B(J)+DABS(A(IA))
      GO TO 640
  630 B(J)=B(J)+A(IA)*A(IA)
  640 CONTINUE
      IF(N1.GT.0) B(J)=DSQRT(B(J))
  650 CONTINUE
      RETURN
C------EVALUATE PRODUCT OF ALL ELEMENTS-----------
  700 B(1)=1.0D0
      B(2)=1.0D0
      DO 750 I=1,NR
      DO 740 J=1,NC
      IA = NR*(J-1) + I
      B(1)=B(1)*A(IA)
      IF(B(1).NE.0.D0) GO TO 710
      B(2) = 0.0D0
      RETURN
  710 IF(DABS(B(1)).LT.1.D0) GO TO 720
      B(1)=B(1)/10.D0
      B(2)=B(2)+1.0D0
      GO TO 710
  720 IF(DABS(B(1)).GT.0.1D0) GO TO 740
      B(1)=B(1)*10.D0
      B(2)=B(2)-1.0D0
      GO TO 720
  740 CONTINUE
  750 CONTINUE
      RETURN
      END
      SUBROUTINE ADD(S,IERR)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON MTOT,NDP,L(1)
      COMMON /PSIZE/ NO,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR
      COMMON /CARD/ INHOL(3,10),N1,N2,N3,N4,S1,S2
      COMMON /ADIR/ NA,I(10)
      CALL FIND(INHOL(1,3),IERR)
      IF(IERR .GT. 1) RETURN
      NR=I(3)
      NEL=I(3)*I(4)
      NB=NA
      CALL FIND(INHOL(1,2),IERR)
```

```
      IF(NR.NE.I(3)) CALL ERCOM(IERR)
      IF(IERR .GT. 1) RETURN
      CALL ADSB(L(NA),L(NB),NEL,S)
      RETURN
      END
      SUBROUTINE ADSB(A,B,NEL,S)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(1),B(1)
      DO 100 IA=1,NEL
  100 A(IA)=A(IA)+S*B(IA)
      RETURN
      END
      SUBROUTINE MUL(IERR)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON MTOT,NDP,L(1)
      COMMON /PSIZE/ NO,NABA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR
      COMMON /CARD/ INHOL(3,10),N1,N2,N3,N4,S1,S2
      COMMON /ADIR/ NA,I(10)
      CALL DELETE(INHOL(1,4))
      CALL FIND(INHOL(1,2),IERR)
      IF(IERR .GT. 1) RETURN
      NR=I(4)
      NT=I(3)
      N1=NA
      CALL FIND(INHOL(1,3),IERR)
      IF(NT.NE.I(4)) CALL ERCOM(IERR)
      IF(IERR .GT. 1) RETURN
      N2=NA
      NC=I(3)
      CALL LIST(INHOL(1,4),NR,NC,NDP,IERR)
      IF(IERR .GT. 1) RETURN
      CALL MULT(L(N1),L(N2),L(NA),NR,NT,NC)
      RETURN
      END
      SUBROUTINE MULT(A,B,C,NRA,NCA,NCB)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(1),B(1),C(1)
C-----DIMENSION A(NRA,NCA),B(NCA,NCB),C(NRA,NCB)
      DO 200 I=1,NRA
      DO 200 J=1,NCB
      X=0.0D0
      IC = (J-1)*NRA + I
      DO 100 K=1,NCA
      IA = (K-1)*NRA+I
      IB = (J-1)*NCA+K
  100 X=X+A(IA)*B(IB)
  200 C(IC)=X
      RETURN
      END
```

```
      SUBROUTINE TRAN(IERR)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON MTOT,NDP,L(1)
      COMMON /PSIZE/ NO,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR
      COMMON /CARD/ INHOL(3,10),N1,N2,N3,N4,S1,S2
      COMMON /ADIR/ NA,I(10)
      CALL DELETE(INHOL(1,3))
      CALL FIND(INHOL(1,2),IERR)
      IF(IERR .GT. 1) RETURN
      NR=I(4)
      NC=I(3)
      N1=NA
      CALL LIST(INHOL(1,3),NC,NR,NDP,IERR)
      IF(IERR .GT. 1) RETURN
      CALL TRANS(L(N1),L(NA),NR,NC)
      RETURN
      END
      SUBROUTINE TRANS(A,B,NR,NC)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(1),B(1)
      K = 1
      DO 100 I = 1,NR
      L = I
      DO 100 J = 1,NC
      B(K) = A(L)
      L = L + NR
  100 K = K + 1
      RETURN
      END
      SUBROUTINE SMOP(N,IERR)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON MTOT,NDP,L(1)
      COMMON /PSIZE/ NO,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR
      COMMON /CARD/ INHOL(3,10),N1,N2,N3,N4,S1,S2
      COMMON /ADIR/ NA,I(10)
C-----SUBROUTINE TO DUPLICATE OR STORE SUBMATRICES--------
      IF(N.NE.1) GO TO 100
      CALL DELETE(INHOL(1,3))
      CALL LIST(INHOL(1,3),N3,N4,NDP,IERR)
      IF(IERR .GT. 1) RETURN
      GO TO 200
  100 CALL FIND(INHOL(1,3),IERR)
      IF(IERR .GT. 1) RETURN
      N3=I(4)
      N4=I(3)
  200 NB=NA
      CALL FIND(INHOL(1,2),IERR)
      IF(IERR .GT. 1) RETURN
      NR=I(4)
```

```
      NC=L(3)
      IF(N1+N3-1.GT.NR)  CALL  ERCOM(IERR)
      IF(N2+N4-1.GT.NC)  CALL  ERCOM(IERR)
      IF(IERR .GT. 1)  RETURN
      CALL SMOPS(L(NA),L(NB),NR,NC,N1,N2,N3,N4,N)
      RETURN
      END
      SUBROUTINE  SMOPS(A,B,NR,NC,N1,N2,N3,N4,N)
      IMPLICIT REAL*8  (A-H,O-Z)
      DIMENSION A(NR,1),B(N3,1)
      II=N1
      DO 400  I=1,N3
      JJ=N2
      DO 300   J=1,N4
      GO TO  (1,2)  ,N
    1 B(I,J)=A(II,JJ)
      GO TO 300
    2 A(II,JJ)=B(I,J)
  300 JJ=JJ+1
  400 II=II+1
      RETURN
      END
      SUBROUTINE  SYMSV(IERR)
      IMPLICIT REAL*8  (A-H,O-Z)
      COMMON  MTOT,NDP,L(1)
      COMMON  /PSIZE/ NO,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR
      COMMON  /CARD/ INHOL(3,10),N1,N2,N3,N4,S1,S2
      COMMON  /ADIR/ NA,I(10)
      IF(NO.NE.0)  GO TO  100
      IF(N1.EQ.1)  WRITE(NWRITE,2000)
      IF  (N1.EQ.2)  WRITE(NWRITE,2001)
      IF(N1.EQ.3)  WRITE(NWRITE,2002)
  100 CALL  FIND(INHOL(1,2),IERR)
      IF(IERR .GT. 1)  RETURN
      NR=L(4)
      IF(N1.EQ.3)  GO  TO  300
      IF(NR.NE.I(3))  CALL ERCOM(IERR)
      IF(IERR  .GT.  1)  RETURN
      N3=NA
      IF(N1.EQ.1)  GO TO  200
      CALL  FIND(INHOL(1,3),IERR)
      IF(NR.NE.I(4))  CALL  ERCOM(IERR)
      IF(IERR  .GT.  1)  RETURN
  200 CALL  SYMSOL(L(N3),L(NA),NR,I(3),N1,N2)
      RETURN
  300 CALL  SYMIN(L(NA),NR)
      IF(N2.NE.0)WRITE(NWRITE,2003)
      RETURN
 2000 FORMAT  (20H TRIANGULARIZE ONLY   )
```

```
 2001 FORMAT (44H FORWARD REDUCTION AND BACKSUBSTITUTION ONLY )        STO10440
 2002 FORMAT (22H MATRIX INVERSION ONLY )                              STO10450
 2003 FORMAT(48H0   ***WARNING*** MATRIX REQUIRED TO BE SYMMETRIC)     STO10460
      END                                                              STO10470
      SUBROUTINE SYMSOL(A,B,NN,LL,M,N2)                                STO10480
      IMPLICIT REAL*8 (A-H,O-Z)                                        STO10490
      COMMON /PSIZE/ NO,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR            STO10500
C     SYMMETRIC EQUATION SOLVER- E.L. WILSON 1976                      STO10510
C     M=0 TRIANGULARIZE AND SOLVE                                      STO10520
C     M=1 TRIANGULARIZE ONLY                                           STO10530
C     M=2 FORWARD REDUCTION AND BACKSUBSTITUTION ONLY                  STO10540
C     IF N2 .NE. 0 MATRIX IS NON SYMMETRIC AND LU DECOMPOSITION IS USED.STO10550
      DIMENSION A(NN,1),B(NN,1)                                        STO10560
C     ESTIMATE THE ROUNDOFF VALUE OF ZERO                              STO10570
      ZERO = 0.0D0                                                     STO10580
      DO 10 I = 1,NN                                                   STO10590
   10 ZERO = ZERO + DABS(A(I,I))                                       STO10600
      ZERO = ZERO * 1.0D-16                                            STO10610
      IF(N2.NE.0) GO TO 1000                                           STO10620
C                                                                      STO10630
      IF(M.EQ.2) GO TO 500                                             STO10640
      DO 400 N=1,NN                                                    STO10650
      IF(N.EQ.NN) GO TO 500                                            STO10660
      D=A(N,N)                                                         STO10670
      IF(DABS(D).LT.ZERO) WRITE(NWRITE,2000) N                         STO10680
      N1=N+1                                                           STO10690
      DO 300 J=N1,NN                                                   STO10700
      IF(A(N,J).EQ.0.0D0) GO TO 300                                    STO10710
      A(N,J)=A(N,J)/D                                                  STO10720
      DO 200 I=J,NN                                                    STO10730
      A(I,J)=A(I,J)-A(I,N)*A(N,J)                                      STO10740
  200 A(J,I)=A(I,J)                                                    STO10750
  300 CONTINUE                                                         STO10760
  400 CONTINUE                                                         STO10770
C     FORWARD REDUCTION AND BACKSUBSTITUTION                           STO10780
  500 IF(M.EQ.1) RETURN                                                STO10790
      DO 700 N=1,NN                                                    STO10800
      DO 600 L=1,LL                                                    STO10810
  600 B(N,L)=B(N,L)/A(N,N)                                             STO10820
      IF(N.EQ.NN) GO TO 800                                            STO10830
      N1=N+1                                                           STO10840
      DO 700 L=1,LL                                                    STO10850
      DO 700 I=N1,NN                                                   STO10860
  700 B(I,L)=B(I,L)-A(I,N)*B(N,L)                                      STO10870
C                                                                      STO10880
  800 N1=N                                                             STO10890
      N=N-1                                                            STO10900
      IF(N.EQ.0) RETURN                                                STO10910
      DO 900 L=1,LL                                                    STO10920
```

```
      DO 900  J=N1,NN
  900 B(N,L)=B(N,L)-A(N,J)*B(J,L)
      GO TO 800
C     NON-SYMMETRIC MATRIX LU DECOMPOSITION
 1000 NM1=NN-1
      IF(M.EQ.2)GO TO 1500
      DO 1100  K=1,NM1
      KP1=K+1
      AKK=A(K,K)
      IF(DABS(AKK).LT.ZERO) WRITE(NWRITE,2000) K
      DO 1100  I=KP1,NN
      G=-A(I,K)/AKK
      A(I,K)  = G
      DO 1100  J=KP1,NN
 1100 A(I,J)=A(I,J)+G*A(K,J)
C     FORWARD REDUCTION AND BACK SUBSTITUTION
      IF(M.EQ.1)  RETURN
 1500 NP1 = NN + 1
      DO 1400 L = 1,LL
      DO 1200 K = 1,NM1
      KP1 = K + 1
      BK = B(K,L)
      DO 1200 I = KP1,NN
 1200 B(I,L)= B(I,L)  +  A(I,K)*BK
      B(NN,L)=B(NN,L)/A(NN,NN)
      DO 1400 K =  2,NN
      I = NP1-K
      J1 = I + 1
      BI = B(I,L)
      DO 1300 J = J1,NN
 1300 BI = BI - A(I,J)*B(J,L)
      B(I,L)=BI
 1400 B(I,L)=B(I,L)/A(I,I)
      RETURN
C
 2000 FORMAT (39H0***ZERO DIAGONAL TERM EQUATION NUMBER   I4)
      END
      SUBROUTINE SYMIN(A,M)
      IMPLICIT REAL*8  (A-H,O-Z)
C-----INVERSION OF POSITIVE DEFINITE SYMMETRIC MATRIX-------
      DIMENSION A(M,1)
C     EVALUATION OF NEGATIVE INVERSE
      DO 400  N=1,M
      D=A(N,N)
      DO 300  I=1,M
      AN=A(I,N)/D
      IF(I.EQ.N)  GO TO 200
      DO 100  J=I,M
      A(I,J)=A(I,J)-AN*A(N,J)
```

```
  100 A(J,I)=A(I,J)
  200 A(I,N)=AN
  300 A(N,I)=AN
  400 A(N,N)=-1.0D0/D
C     CHANGE SIGN OF INVERSE
      DO 500 I=1,M
      DO 500 J=1,M
  500 A(I,J)=-A(I,J)
C
      RETURN
      END
      SUBROUTINE USERA(IERR)
      IMPLICIT REAL*8 (A-H,O-Z)
      WRITE(NWRITE,1000)
 1000 FORMAT(37H0 SUBROUTINE USERA IS NOT YET WRITTEN)
      RETURN
      END
      SUBROUTINE USERB (IERR)
      IMPLICIT REAL*8 (A-H,O-Z)
      WRITE(NWRITE,1000)
 1000 FORMAT(37H0 SUBROUTINE USERB IS NOT YET WRITTEN)
      RETURN
      END
      SUBROUTINE GROUP2(LAST,IERR)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON MTOT,NDP,L(1)
      COMMON /PSIZE/ NO,NABA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR
      COMMON /CARD/ INHOL(3,10),N1,N2,N3,N4,S1,S2
      COMMON /TEMP/ SSS(100)
      LOGICAL COMP
      DIMENSION IOP(3,14)
      DATA IOP(1,1),IOP(2,1),IOP(3,1)  /4HNODE,4HS   ,4H    /
      DATA IOP(1,2),IOP(2,2),IOP(3,2)  /4HBOUN,4HD   ,4H    /
      DATA IOP(1,3),IOP(2,3),IOP(3,3)  /4HTRUS,4HS   ,4H    /
      DATA IOP(1,4),IOP(2,4),IOP(3,4)  /4HBEAM,4H    ,4H    /
      DATA IOP(1,5),IOP(2,5),IOP(3,5)  /4HADDS,4HP   ,4H    /
      DATA IOP(1,6),IOP(2,6),IOP(3,6)  /4HLOAD,4HS   ,4H    /
      DATA IOP(1,7),IOP(2,7),IOP(3,7)  /4HFORC,4HE   ,4H    /
      DATA IOP(1,8),IOP(2,8),IOP(3,8)  /4HDISP,4HL   ,4H    /
      DATA IOP(1,9),IOP(2,9),IOP(3,9)  /4HLOAD,4HI   ,4H    /
      DATA IOP(1,10),IOP(2,10),IOP(3,10)  /4HPLAN,4HE   ,4H    /
      DATA IOP(1,11),IOP(2,11),IOP(3,11)  /4HSLOP,4HE   ,4H    /
      DATA IOP(1,12),IOP(2,12),IOP(3,12)  /4HADDK,4H    ,4H    /
      DATA IOP(1,13),IOP(2,13),IOP(3,13)  /4HEMFR,4HC   ,4H    /
      DATA IOP(1,14),IOP(2,14),IOP(3,14)  /4HFRAM,4HE   ,4H    /
      NUMOP=14
      GO TO 175
C---- READ OPERATION FROM CARD OR STORAGE  -----------------
  100 CALL INPUT(IERR)
```

```
      IF(IERR .GT. 1) RETURN                                            STO11910
C-----INTERPRETE OPERATION ---------------------------------             STO11920
  175 DO 200 J=1,NUMOP                                                  STO11930
      N=J                                                               STO11940
      IF (COMP(INHOL(1,1),IOP(1,J))) GO TO 300                          STO11950
  200 CONTINUE                                                          STO11960
      RETURN                                                            STO11970
C-----EXECUTE APPROPRIATE OPERATION ----------------------               STO11980
  300 LAST=2                                                            STO11990
      CALL STRUC(N,IERR)                                                STO12000
      IF(IERR .GT. 1) RETURN                                            STO12010
      GO TO 100                                                         STO12020
      END                                                               STO12030
      SUBROUTINE STRUC(NOP,IERR)                                        STO12040
      IMPLICIT REAL*8 (A-H,O-Z)                                         STO12050
      COMMON MTOT,NDP,L(1)                                              STO12060
      COMMON /PSIZE/ NO,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR             STO12070
      COMMON /CARD/ INHOL(3,10),N1,N2,N3,N4,S1,S2                       STO12080
      COMMON /STR/ ST(9,16),XM(16),S(16,16),ND,NS,NRP,LM(16)            STO12090
      COMMON /ADIR/ NA,I(10)                                            STO12100
      COMMON /PSAVE/ NUMNP,NEQ,NPROP,NUMEL,NTRUSS                       STO12110
      EQUIVALENCE (M1,INHOL(1,2)),(M2,INHOL(1,3)),(M3,INHOL(1,4))       STO12120
      EQUIVALENCE (M4,INHOL(1,5)),(M5,INHOL(1,6)),(M6,INHOL(1,7))       STO12130
      DATA IBLANK/4H    /                                               STO12140
      GO TO (100,200,300,300,500,600,700,800,900,300,910,920,930,940),  STO12150
     1  NOP                                                             STO12160
C-----READ NODE COORDINATES----------------------                       STO12170
  100 CALL DELETE(M1)                                                   STO12180
      NUMNP=N1                                                          STO12190
      CALL LIST(M1,N1,3,NDP,IERR)                                       STO12200
      IF(IERR .GT. 1) RETURN                                            STO12210
      CALL NODES(L(NA),N1)                                              STO12220
      RETURN                                                            STO12230
C-----SPECIFICATION OF DISPLACEMENT BOUNDARY CONDITIONS                  STO12240
  200 CALL DELETE(M1)                                                   STO12250
      CALL LIST(M1,NUMNP,6,NSP,IERR)                                    STO12260
      IF(IERR .GT. 1) RETURN                                            STO12270
      CALL BOUND(L(NA),NUMNP,NEQ)                                       STO12280
      CALL PTAPE(1,1)                                                   STO12290
      NUMEL=0                                                           STO12300
      RETURN                                                            STO12310
C-----CALCULATION OF TRUSS AND BEAM STIFFNESS MATRICES------             STO12320
  300 CALL FIND(M2,IERR)                                                STO12330
      IF(IERR .GT. 1) RETURN                                            STO12340
      NC=N1                                                             STO12350
      NE=N2                                                             STO12360
      N1=NA                                                             STO12370
      CALL FIND(M3,IERR)                                                STO12380
      IF(IERR .GT. 1) RETURN                                            STO12390
```

```
      N2=NA                                                             STO12400
      CALL FIND(M4,IERR)                                                STO12410
      IF(IERR .GT. 1) RETURN                                            STO12420
      NPROP=I(4)                                                        STO12430
      N3=N1+NUMNP*NDP                                                   STO12440
      N4=N1+2*NUMNP*NDP                                                 STO12450
      IF(NOP.EQ.10) GO TO 370                                           STO12460
      IF(NOP.EQ.4) GO TO 350                                            STO12470
C-----CALCULATION OF TRUSS ELEMENT MATRICES-----NOP=3                   STO12480
      ND=6                                                              STO12490
      NS=1                                                              STO12500
      CALL TRUSS(L(N1),L(N3),L(N4),L(N2),L(NA),NUMNP,NTRUSS,NPROP)      STO12510
      GO TO 400                                                         STO12520
C-----CALCULATION OF BEAM ELEMENT STIFFNESS MATRICES-- NOP=4             STO12530
  350 ND=12                                                             STO12540
      NS=3                                                              STO12550
      CALL BEAM(L(N1),L(N3),L(N4),L(N2),L(NA),NUMNP,NTRUSS,NPROP)       STO12560
      GO TO 400                                                         STO12570
C-----CALCULATION OF 3 TO 8 NODE FINITE ELEMENT---                      STO12580
  370 ND=16                                                             STO12590
      NS=9                                                              STO12600
      CALL PLANE(L(N1),L(N3),L(N4),L(N2),L(NA),NUMNP,NTRUSS,NPROP,NC,NE) STO12610
  400 CALL DELETE(M1)                                                   STO12620
      CALL LIST(M1,1,5,NSP,IERR)                                        STO12630
      IF(IERR .GT. 1) RETURN                                            STO12640
      L(NA)=NTRUSS                                                      STO12650
      L(NA+1)=ND                                                        STO12660
      L(NA+2)=NS                                                        STO12670
      L(NA+4)=NUMEL+1                                                   STO12680
      NUMEL=NUMEL+NTRUSS                                                STO12690
      RETURN                                                            STO12700
C------FORMATION OF TOTAL STIFFNESS AND MASS MATRICES                   STO12710
  500 CALL DELETE(M1)                                                   STO12720
      IF(M2.NE.IBLANK) CALL DELETE(M2)                                  STO12730
      CALL PTAPE(1,1)                                                   STO12740
      IF(M2.EQ.IBLANK) GO TO 550                                        STO12750
      CALL LIST(M2,NEQ,1,NDP,IERR)                                      STO12760
      IF(IERR .GT. 1) RETURN                                            STO12770
      N1=NA                                                             STO12780
  550 CALL LIST(M1,NEQ,NEQ,NDP,IERR)                                    STO12790
      IF(IERR .GT. 1) RETURN                                            STO12800
      CALL ADDSF(L(NA),L(N1),NEQ,NUMEL,M2)                              STO12810
      RETURN                                                            STO12820
C-----FORMATION OF LOAD MATRIX-------------------                       STO12830
  600 CALL DELETE(M1)                                                   STO12840
      CALL FIND(M2,IERR)                                                STO12850
      IF(IERR .GT. 1) RETURN                                            STO12860
      N2=NA                                                             STO12870
      CALL LIST(M1,NEQ,N1,NDP,IERR)                                     STO12880
```

```
      IF(IERR .GT. 1) RETURN
      CALL LOADS(L(N2),L(NA),NEQ,N1,NUMNP)
      RETURN
C-----EVALUATION OF MEMBER FORCES
  700 CALL FIND(M1,IERR)
      IF(IERR .GT. 1) RETURN
      NUME=L(NA)
      ND=L(NA+1)
      NSS=0
      NS=L(NA+2)
      NR=L(NA+4)
      CALL PTAPE(1,NR)
      CALL FIND(M2,IERR)
      IF(IERR .GT. 1) RETURN
      N1=NA
      NL=I(3)
      IF(M3.EQ.IBLANK) GO TO 1050
      NSS=NS*NUME
      CALL LIST(M3,NSS,NL,NDP,IERR)
      IF(IERR .GT. 1) RETURN
 1050 CALL FORCE(L(N1),L(NA),NEQ,NL,NUME,NSS)
      RETURN
C-----PRINT OF NODE DISPLACEMENTS-----------------
  800 IF(IO.NE.0) RETURN
      CALL FIND(M1,IERR)
      IF(IERR .GT. 1) RETURN
      N2=NA
      NL=I(3)
      CALL FIND(M2,IERR)
      IF(IERR .GT. 1) RETURN
      CALL DISPL(L(N2),L(NA),NEQ,NL,NUMNP)
      RETURN
C----- LOAD INTERGER ARRAY-----------------------
  900 CALL DELETE(M1)
      CALL LIST(M1,N1,N2,NSP,IERR)
      IF(IERR .GT. 1) RETURN
      CALL LOADI(L(NA),N1,N2,N3,N4)
      RETURN
C-----FORM 4 X 4 BEAM STIFFNESS MATRIX---------------
  910 CALL DELETE(M1)
      CALL LIST(M1,4,4,NDP,IERR)
      IF(IERR .GT. 1) RETURN
      CALL SLOPE(L(NA))
      RETURN
C-----ADD ELEMENT STIFFNESS TO TOTAL STIFFNESS----------
  920 CALL LOCATE(MI,NK,NR,NEQ,IERR)
      IF(NR.NE.NEQ) CALL ERCOM(IERR)
      IF(IERR.GT.1) RETURN
      CALL LOCATE(M2,NE,NR,NC,IERR)
```

```
      IF(NR.NE.NC)CALL ERCOM(IERR)                                      STO13380
      IF(IERR .GT. 1) RETURN                                            STO13390
      CALL LOCATE(M3,NL,ND,NC,IERR)                                     STO13400
      IF(NR.NE.ND)CALL ERCOM(IERR)                                      STO13410
      IF(IERR .GT. 1) RETURN                                            STO13420
      NL=NL+NR*(N1-1)                                                   STO13430
      CALL ADDK(L(NK),L(NE),L(NL),NR,NEQ)                               STO13440
      RETURN                                                            STO13450
C-----CALCULATE MEMBER FORCES---------------------------               STO13460
  930 CALL DELETE(M4)                                                   STO13470
      CALL LOCATE(M1,NE,NR,NC,IERR)                                     STO13480
      CALL LOCATE(M2,ND,NEQ,NL,IERR)                                    STO13490
      CALL LOCATE(M3,N2,N3,N4,IERR)                                     STO13500
      N2=N3+N2*(N1-1)                                                   STO13510
      CALL LIST(M4,NR,NL,NDP,IERR)                                      STO13520
      IF(IERR .GT. 1) RETURN                                            STO13530
      CALL MEMFRC(L(NE),L(ND),L(N2),L(NA),NR,NC,NEQ,NL)                 STO13540
      RETURN                                                            STO13550
C-----CALCULATION OF 2D FRAME MATRICES- 6 DOF --------                 STO13560
  940 CALL DELETE(M1)                                                   STO13570
      CALL DELETE (M2)                                                  STO13580
      CALL LIST(M1,6,6,NDP,IERR)                                        STO13590
      NK=NA                                                             STO13600
      CALL LIST(M2,3,6,NDP,IERR)                                        STO13610
      IF(IERR .GT. 1)RETURN                                             STO13620
      CALL FRAME(L(NK),L(NA))                                           STO13630
      RETURN                                                            STO13640
      END                                                               STO13650
      SUBROUTINE LOCATE(M1,NT,NR,NC,IERR)                               STO13660
      COMMON /ADIR/ NA,I(10)                                            STO13670
      DIMENSION M1(1)                                                   STO13680
      CALL FIND(M1,IERR)                                                STO13690
      NT=NA                                                             STO13700
      NC=I(3)                                                           STO13710
      NR=I(4)                                                           STO13720
      RETURN                                                            STO13730
      END                                                               STO13740
      SUBROUTINE ADDSF(A,B,NEQ,NUMEL,M2)                                STO13750
      IMPLICIT REAL*8 (A-H,O-Z)                                         STO13760
      COMMON /STR/ ST(9,16),XM(16),S(16,16),ND,NS,NRP,LM(16)            STO13770
      DIMENSION A(NEQ,1),B(1)                                           STO13780
      DATA IBLANK/4H    /                                               STO13790
C                                                                       STO13800
      DO 100 I=1,NEQ                                                    STO13810
      IF(M2.NE.IBLANK) B(I)=0.0D0                                       STO13820
      DO 100 J=1,NEQ                                                    STO13830
  100 A(I,J)=0.0D0                                                      STO13840
      DO 300 N=1,NUMEL                                                  STO13850
      CALL PTAPE(3,NR)                                                  STO13860
```

```
      DO 200 I=1,ND
      II=LM(I)
      IF(II.LE.0) GO TO 200
      IF(M2.NE.IBLANK) B(II)=B(II)+XM(I)
      DO 150 J=1,ND
      JJ=LM(J)
      IF(JJ.LE.0) GO TO 150
      A(II,JJ)=A(II,JJ)+S(I,J)
  150 CONTINUE
  200 CONTINUE
  300 CONTINUE
      RETURN
      END
      SUBROUTINE ADDK(A,S,LM,NR,NEQ)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NEQ,1),S(NR,1),LM(NR)
C-----ADDITION OF MEMBER STIFFNESS TO TOTAL STIFFNESS-----
      DO 400 I=1,NR
      II=LM(I)
      IF(II.EQ.0)GO TO 400
      DO 300 J=1,NR
      JJ=LM(J)
      IF(JJ.EQ.0)GO TO 300
      A(II,JJ)=A(II,JJ)+S(I,J)
  300 CONTINUE
  400 CONTINUE
      RETURN
      END
      SUBROUTINE SLOPE(S)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /PSIZE/ NO,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR,INDIC
      DIMENSION S(4,1)
      READ(NREAD,1000) XI,E,XL
      IF(NO.EQ.0) WRITE(NWRITE,2000) XI,E,XL
      EIL=E*XI/XL
      S(1,1)=4.0D0*EIL
      S(1,2)=2.0D0*EIL
      S(1,3)=-6.0D0*EIL/XL
      S(1,4)=-S(1,3)
      S(2,2)=S(1,1)
      S(2,3)=S(1,3)
      S(2,4)=S(1,4)
      S(3,3)=12.0D0*EIL/(XL*XL)
      S(3,4)=-S(3,3)
      S(4,4)=S(3,3)
      DO 100 J=1,3
      JP1 = J + 1
      DO 100 I = JP1,4
  100 S(I,J)=S(J,I)
```

```
      RETJRN                                                            STO14360
 1000 FORMAT(3F10.0)                                                    STO14370
 2000 FORMAT(4H I= ,1PD15.6,5H  E= ,1PD15.6,5H  L= ,1PD15.6)            STO14380
      END                                                               STO14390
      SUB3OUTINE FRAME(S,A)                                             STO14400
      IMPLICIT REAL*8 (A-H,O-Z)                                         STO14410
      COMMON /PSIZE/ NO,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR,INDIC       STO14420
      DIMENSION S(6,1),A(3,1)                                           STO14430
C-----------------------------------------                              STO14440
      READ(NREAD,1000)AR,E,XI,X1,Y1,X2,Y2                               STO14450
      IF(NO.EQ.0) WRITE(NWRITE,2000) AR,E,XI,X1,Y1,X2,Y2                STO14460
      DX=X2-X1                                                          STO14470
      DY = Y2-Y1                                                        STO14480
      XL = DSQRT(DX*DX+DY*DY)                                           STO14490
      SIN=DY/XL                                                         STO14500
      COS=DX/XL                                                         STO14510
C -----------------------------------------                             STO14520
      A(1,1)= SIN/XL                                                    STO14530
      A(1,2)=-COS/XL                                                    STO14540
      A(1,3)=1.0D0                                                      STO14550
      A(1,4)=-A(1,1)                                                    STO14560
      A(1,5)=-A(1,2)                                                    STO14570
      A(1,6)=0.0D0                                                      STO14580
      A(2,1)= SIN/XL                                                    STO14590
      A(2,2)=-COS/XL                                                    STO14600
      A(2,3)=0.0D0                                                      STO14610
      A(2,4)=A(1,4)                                                     STO14620
      A(2,5)=A(1,5)                                                     STO14630
      A(2,6)=1.0D0                                                      STO14640
      A(3,1)=-COS                                                       STO14650
      A(3,2)=-SIN                                                       STO14660
      A(3,3)=0.0D0                                                      STO14670
      A(3,4)=COS                                                        STO14680
      A(3,5)=SIN                                                        STO14690
      A(3,6)=0.0D0                                                      STO14700
C-----------------------------------------                              STO14710
      S12=2.0D0*E*XI/XL                                                 STO14720
      S11=S12+S12                                                       STO14730
      S33=AR*E/XL                                                       STO14740
C-----------------------------------------                              STO14750
      DO 300 I=1,6                                                      STO14760
      T1=S11*A(1,I)+S12*A(2,I)                                          STO14770
      T2=S12*A(1,I)+S11*A(2,I)                                          STO14780
      T3=S33*A(3,I)                                                     STO14790
      DO 200 J=1,6                                                      STO14800
      S(J,I)=A(1,J)*T1+A(2,J)*T2+A(3,J)*T3                              STO14810
  200 S(I,J)=S(J,I)                                                     STO14820
      A(1,I)=T1                                                         STO14830
      A(2,I)=T2                                                         STO14840
```

```
      A(3,I)=T3
  300 CONTINUE
      RETURN
 1000 FORMAT(7F10.0)
 2000 FORMAT(14X,1HA,14X,1HE,14X,1HI,13X,2HX1,13X,2HY1,
     1 13X,2HX2,13X,2HY2,/(7F15.4))
      END
      SUBROUTINE MEMFRC(S,U,LM,F,NR,NC,NEQ,NL)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION S(NR,NC),U(NEQ,NL),LM(NC),F(NR,NL)
C-----EVALUATION OF MEMBER FORCES------------
      DO 100 I=1,NR
      DO 100 L=1,NL
      SUM=0.0D0
      DO 50 K = 1,NC
      KK=LM(K)
      IF(KK.EQ.0) GO TO 50
      SUM=SUM +S(I,K)*U(KK,L)
   50 CONTINUE
      F(I,L)=SUM
  100 CONTINUE
      RETURN
      END
      SUBROUTINE TRUSS(X,Y,Z,ID,EE,NUMNP,NTRUSS,NPROP)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /PSIZE/ NO,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR
      DIMENSION X(1),Y(1),Z(1),ID(NUMNP,6),EE(NPROP,3)
      COMMON /STR/ ST(9,16),XM(16),S(16,16),ND,NS,NRP,LM(16)
C------EVALUATION OF TRUSS ELEMENT MATRICES-------
      NTRUSS=0
      IF(NO.EQ.0) WRITE(NWRITE,2000)
  100 READ(NREAD,1001) M,I,J,NP,NPT
      IF(M.EQ.0) RETURN
      AREA=EE(NP,1)
      E=EE(NP,2)
      DEN=EE(NP,3)
      IF(NO.EQ.0) WRITE(NWRITE,2001) M,I,J,AREA,E,DEN
      DX=X(I)-X(J)
      DY=Y(I)-Y(J)
      DZ=Z(I)-Z(J)
      XL2=DX*DX+DY*DY+DZ*DZ
      XL=DSQRT(XL2)
      DEN=DEN*XL/2.D0
      XX=E*AREA/XL
      ST(1,1)=DX/XL
      ST(1,2)=DY/XL
      ST(1,3)=DZ/XL
      ST(1,4)=-ST(1,1)
      ST(1,5)=-ST(1,2)
```

```
      ST(1,6)=-ST(1,3)                                                   STO15340
C                                                                        STO15350
      DO 300 L=1,6                                                       STO15360
      YY=ST(1,L)*XX                                                      STO15370
      DO 250 K=L,6                                                       STO15380
      S(K,L)=ST(1,K)*YY                                                  STO15390
  250 S(L,K)=S(K,L)                                                      STO15400
      ST(1,L)=YY                                                         STO15410
  300 XM(L)=DEN                                                          STO15420
      DO 400 L=1,3                                                       STO15430
      LM(L)=ID(I,L)                                                      STO15440
  400 LM(L+3)=ID(J,L)                                                    STO15450
      CALL PTAPE(2,NR)                                                   STO15460
      NTRUSS=NTRUSS+1                                                    STO15470
      GO TO 100                                                          STO15480
C                                                                        STO15490
 1000 FORMAT(5I5)                                                        STO15500
 2000 FORMAT( 7HONUMBER 6X 1HI 6X 1HJ 11X 4HAREA 14X 1HE 12X 3HDEN )     STO15510
 2001 FORMAT (3I7,F15.3,2(1PD15.6))                                      STO15520
      END                                                                STO15530
      SUBROUTINE BEAM(X,Y,Z,ID,P,NUMNP,NUME,NP)                          STO15540
      IMPLICIT REAL*8 (A-H,O-Z)                                          STO15550
      DIMENSION X(NUMNP),Y(NUMNP),Z(NUMNP),ID(NUMNP,6),P(NP,7)           STO15560
      COMMON /PSIZE/ NO,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR              STO15570
      COMMON /STR/ ST(9,16),XM(16),S(16,16),ND,NS,NRP,LM(16)             STO15580
      COMMON /TEMP/ A(6,12),T(3,3),V(4,4)                                STO15590
C-----READ AND PRINT BEAM DATA--------------------                       STO15600
      NUME=0                                                             STO15610
      IF(NO.EQ.0) WRITE(NWRITE,2000)                                     STO15620
  100 READ(NREAD,1000) N,I,J,K,NN                                        STO15630
      IF(N.EQ.0) RETURN                                                  STO15640
      IF(NO.EQ.0) WRITE(NWRITE,2001) N,I,J,K,NN,(P(NN,L),L=1,7)          STO15650
      NUME=NUME+1                                                        STO15660
C-----FORM 3 X 3 TRANSFORMATION MATRX ------------                       STO15670
      CALL VECTOR(V(1,1),X(I),Y(I),Z(I),X(J),Y(J),Z(J))                  STO15680
      XL=V(4,1)                                                          STO15690
      CALL VECTOR(V(1,4),X(I),Y(I),Z(I),X(K),Y(K),Z(K))                  STO15700
      CALL CROSS(V(1,1),V(1,4),V(1,3))                                   STO15710
      CALL CROSS(V(1,3),V(1,1),V(1,2))                                   STO15720
      DO 200 K=1,3                                                       STO15730
      DO 200 L=1,3                                                       STO15740
  200 T(K,L)=V(L,K)                                                      STO15750
C-----FORM DESTINATION VECTOR --------------------                       STO15760
      DO 300 L=1,6                                                       STO15770
      LM(L)=ID(I,L)                                                      STO15780
  300 LM(L+6)=ID(J,L)                                                    STO15790
C-----FORM STIFFNESS MATRIX IN LOCAL SYSTEM-------                       STO15800
      DO 400 I=1,6                                                       STO15810
      DO 400 J=1,6                                                       STO15820
```

```
  400 S(I,J)=0.0D0                                               ST015830
      S(1,1)=P(NN,1)*P(NN,5)/XL                                  ST015840
      S(4,4)=P(NN,2)*P(NN,6)/XL                                  ST015850
      Q=4.D0*P(NN,5)/XL                                          ST015860
      S(5,5)=Q*P(NN,3)                                           ST015870
      S(6,6)=Q*P(NN,4)                                           ST015880
      Q=3.D0/(XL*XL)                                             ST015890
      S(3,3)=Q*S(5,5)                                            ST015900
      S(2,2)=Q*S(6,6)                                            ST015910
      Q=1.5D0/XL                                                 ST015920
      S(2,6)=-Q*S(6,6)                                           ST015930
      S(6,2)=S(2,6)                                              ST015940
      S(3,5)=Q*S(5,5)                                            ST015950
      S(5,3)=S(3,5)                                              ST015960
C-----FORM LOCAL-GLOBAL TRANSFORMATION-----------                ST015970
      DO 500 I=1,6                                               ST015980
      DO 500 J=1,12                                              ST015990
  500 A(I,J)=0.0D0                                               ST016000
      DO 500 J=1,3                                               ST016010
      A(2,J+3)=-XL*T(3,J)                                        ST016020
      A(3,J+3)= XL*T(2,J)                                        ST016030
      DO 500 I=1,3                                               ST016040
      A(I,J)=-T(I,J)                                             ST016050
      A(I+3,J+3)=-T(I,J)                                         ST016060
      A(I,J+6)=T(I,J)                                            ST016070
  500 A(I+3,J+9)=T(I,J)                                          ST016080
C----- FORM GLOBAL STIFFNESS MATRIX                               ST016090
      DO 300 I=1,6                                               ST016100
      DO 300 J=1,12                                              ST016110
      ST(I,J)=0.0D0                                              ST016120
      DO 800 K=1,6                                               ST016130
  800 ST(I,J)=ST(I,J)+S(I,K)*A(K,J)                              ST016140
      DO 900 I=1,12                                              ST016150
      ST(7,I)= ST(3,I)*XL-ST(5,I)                                ST016160
      ST(3,I)=-ST(2,I)*XL-ST(6,I)                                ST016170
      DO 900 J=I,12                                              ST016180
      Q=0.0D0                                                    ST016190
      DO 850 K=1,6                                               ST016200
  850 Q=Q+A(K,I)*ST(K,J)                                         ST016210
      S(I,J)=Q                                                   ST016220
  900 S(J,I)=Q                                                   ST016230
C-----CALCULATE MASS MATRIX ---------------------                 ST016240
      XMASS=P(NN,7)*XL/2.D0                                      ST016250
      V(1,1)=XMASS*P(NN,2)/P(NN,1)                               ST016260
      V(2,1)=XMASS*XL*XL/12.D0                                   ST016270
      V(3,1)=V(2,1)                                              ST016280
      DO 950 I=1,3                                               ST016290
      XM(I)=XMASS                                                ST016300
      XM(I+6)=XMASS                                              ST016310
```

```
      XM(I+3)=0.0D0                                                     STO16320
      DO 950  J=1,3                                                     STO16330
  950 XM(I+3)=XM(I+3)+T(J,I)*V(J,1)                                     STO16340
      XM(I+3)=DABS(XM(I+3))                                             STO16350
  960 XM(I+9)=XM(I+3)                                                   STO16360
C-----STORE ELEMENT MATRICES ON LOW SPEED STORAGE--                     STO16370
      CALL PTAPE(2,NR)                                                  STO16380
      GO TO 100                                                         STO16390
 1000 FORMAT(5I5)                                                       STO16400
 2000 FORMAT (25H0 EL.     I     J     K    NP 9X,1HA,7X,3HJ11,7X,3HI22,7X, STO16410
     1 3HI33,9X,1HE,9X,1HG,9X,1HM )                                     STO16420
 2001 FORMAT (5I5,1F10.2,5F10.0,F10.4)                                  STO16430
      END                                                               STO16440
      SUBROUTINE VECTOR(V,XI,YI,ZI,XJ,YJ,ZJ)                            STO16450
      IMPLICIT REAL*8 (A-H,O-Z)                                         STO16460
      DIMENSION V(4)                                                    STO16470
      X=XJ-XI                                                           STO16480
      Y=YJ-YI                                                           STO16490
      Z=ZJ-ZI                                                           STO16500
      V(4)=DSQRT(X*X+Y*Y+Z*Z)                                           STO16510
      V(3)=Z/V(4)                                                       STO16520
      V(2)=Y/V(4)                                                       STO16530
      V(1)=X/V(4)                                                       STO16540
      RETURN                                                            STO16550
      END                                                               STO16560
      SUBROUTINE CROSS(A,B,C)                                           STO16570
      IMPLICIT REAL*8 (A-H,O-Z)                                         STO16580
      DIMENSION A(4),B(4),C(4)                                          STO16590
      X=A(2)*B(3)-A(3)*B(2)                                             STO16600
      Y=A(3)*B(1)-A(1)*B(3)                                             STO16610
      Z=A(1)*B(2)-A(2)*B(1)                                             STO16620
      C(4)=DSQRT(X*X+Y*Y+Z*Z)                                           STO16630
      C(3)=Z/C(4)                                                       STO16640
      C(2)=Y/C(4)                                                       STO16650
      C(1)=X/C(4)                                                       STO16660
      RETURN                                                            STO16670
      END                                                               STO16680
      SUBROUTINE PLANE(X,Y,Z,ID,P,NUMNP,NUME,NP,INR,INS)                STO16690
      IMPLICIT REAL*8 (A-H,O-Z)                                         STO16700
      DIMENSION X(NUMNP),Y(NUMNP),Z(NUMNP),ID(NUMNP,6),P(NP,4)          STO16710
      COMMON /PSIZE/ NO,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR             STO16720
      COMMON /STR/ ST(9,16),XM(16),S(16,16),ND,NS,NRP,LM(16)            STO16730
      COMMON /TEMP/ H(3,8),YZ(2,8),B(3,16),NODE(8),DJ,NN                STO16740
C---- POINTS AND WEIGHTS FOR 1 TO 3 POINT INTEGRATION-----             STO16750
      DIMENSION RSI(3,3),WGT(3,3),RS(2,3)                               STO16760
      DATA RSI/ 0.D0, 0.D0, 0.D0, -.577350269189626D0, 0.577350269189626STO16770
     1D0,  0.0D0,  -0.774596669241483D0, 0.0D0, 0.774596669241483 D0/  STO16780
      DATA WGT/ 2.0D0, 0.0D0, 0.0D0, 1.0D0, 1.0D0, 0.0D0, 0.555555555555STO16790
     1556D0,         0.888888888888889D0, 0.555555555555556 D0/         STO16800
```

```
      IF(INR.LT.1)  INR=2                                               STO16810
      IF(INR.GT.3)  INR=3                                               STO16820
      IF(INS.LT.1)  INS=2                                               STO16830
      IF(INS.GT.3)  INS=3                                               STO16840
      IF(NO.EQ.0)  WRITE(NWRITE,2002)INR,INS                            STO16850
 2002 FORMAT (22H NUMERICAL INTEGRATION 2I3,22H IN R AND S DIRECTIONS ) STO16860
C-----READ AND PRINT ELEMENT DATA-----------------                      STO16870
      NUME=0                                                            STO16880
      IF(NO.EQ.0)  WRITE(NWRITE,2000)                                   STO16890
  100 READ(NREAD,1000)  N,NODE,NN,RS                                    STO16900
      IF(N.EQ.0)  RETURN                                                STO16910
      IF (NO.EQ.0)  WRITE(NWRITE,2001) N,NODE,(P(NN,L),L=1,4),RS        STO16920
      THK=P(NN,3)                                                       STO16930
      IF(THK.EQ.0.0D0)  THK=1.0D0                                       STO16940
      DEN=P(NN,4)                                                       STO16950
      VOL=0.D0                                                          STO16960
C-----ZERO STIFFNESS ------------------------------                      STO16970
      DO 200  I=1,16                                                    STO16980
      XM(I)=0.0D0                                                       STO16990
      DO 200  J=1,16                                                    STO17000
  200 S(I,J)=0.0D0                                                      STO17010
C-----FORM STRESS-STRAIN MATRIX--------------------                      STO17020
      C11=P(NN,1)/(1.D0-P(NN,2)*P(NN,2))                                STO17030
      C12=C11*P(NN,2)                                                   STO17040
      C33=C11*(1.D0+P(NN,2))/2.D0                                       STO17050
C-----SELECT NODE COORDINATES AND DESTINATION VECTOR                     STO17060
      J=0                                                               STO17070
      DO 250  I=1,8                                                     STO17080
      II=NODE(I)                                                        STO17090
      IF(II)  250,250,240                                               STO17100
  240 J=J+1                                                             STO17110
      YZ(1,J)=Y(II)                                                     STO17120
      YZ(2,J)=Z(II)                                                     STO17130
      LM(2*J-1)=ID(II,2)                                                STO17140
      LM(2*J)=ID(II,3)                                                  STO17150
  250 CONTINUE                                                          STO17160
      NN=J                                                              STO17170
      ND=2*NN                                                           STO17180
C-----FORM BT*C*B FOR EACH INTEGRATION POINT------                       STO17190
      DO 400  IR=1,INR                                                  STO17200
      DO 400  IS=1,INS                                                  STO17210
      CALL FORMB(RSI(IR,INR),RSI(IS,INS))                               STO17220
      DV=WGT(IR,INR)*WGT(IS,INS)*DJ*THK                                 STO17230
      VOL=VOL+DV                                                        STO17240
      DO 300  J=1,NN                                                    STO17250
  300 XM(J)=XM(J)+H(1,J)*H(1,J)*DV                                      STO17260
      DO 400  J=1,ND                                                    STO17270
      CB1=C11*B(1,J)+C12*B(2,J)                                         STO17280
      CB2=C12*B(1,J)+C11*B(2,J)                                         STO17290
```

```
      CB3=:33*B(3,J)
      DO 400  I=J,ND
      SUM=B(1,I)*CB1+B(2,I)*CB2+B(3,I)*CB3
      S(I,J)=S(I,J)+DV*SUM
  400 S(J,I)=S(I,J)
C-----PORM STRESS DISPLACEMENT TRANSFORMATION MATRIX
      DO 500  I=1,3
      CALL FORMB(RS(1,I),RS(2,I))
      JJ=3*(I-1)+1
      DO 500  J=1,ND
      ST(JJ  ,J)=C11*B(1,J)+C12*B(2,J)
      ST(JJ+1,J)=C12*B(1,J)+C11*B(2,J)
  500 ST(JJ+2,J)=C33*B(3,J)
C-----FORM DIAGONAL MASS MATRIX-------------------
      FAC=0.0D0
      DO 550  J=1,NN
  550 FAC=FAC+XM(I)
      FAC=VOL*DEN/FAC
      DO 580  J=1,NN
  580 YZ(1,J)=XM(J)*FAC
      DO 500  J=1,NN
      XM(2*J)=YZ(1,J)
  600 XM(2*J-1)=YZ(1,J)
C-----WRITE ELEMENT MATRICES ON TAPE-------------
      CALL PTAPE(2,NR)
      NUME=NUME+1
      GO TO 100
C-----------------------------------------------------
 1000 FORMAT (10I5,6F5.0)
 2000 FORMAT (116HONODE    N1    N2    N3    N4    N5    N6    N7    N8
     1E    NO         T           M    R1    S1    R2    S2    R3    S3 )
 2001 FORMAT(9I5,4(1PD10.3),6F5.2)
      END
      SUBROUTINE FORMB(R,S)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /TEMP/ H(3,8),YZ(2,8),B(3,16),NODE(8),DJ,NN
C------FORM INTERPOLATION FUNCTIONS AND DERIVATIVES---
      CALL FORMH(R,S)
C------FORM B MATRIX -----------------------------
      DO 500  L=1,NN
      K=2*L-1
      B(1,K)=H(2,L)
      B(1,K+1)=0.0D0
      B(2,K)=0.0D0
      B(2,K+1)=H(3,L)
      B(3,K)=H(3,L)
  600 B(3,K+1)=H(2,L)
      RETJRN
      END
```

```
      SUBROUTINE FORMH(R,S)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /PSIZE/ NJ,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR
      COMMON /TEMP/ H(3,8),YZ(2,8),B(3,16),NODE(8),DJ,NN
      DIMENSION N(2,8),D(2,2)
      DATA N/1,1,-1,1,-1,-1,1,-1,0,1,-1,0,0,-1,1,0/
C------FORM INTERPOLATION FUNCTIONS AND DERIVATIVES---
      DO 200 I=1,8
      CALL GD(R,N(1,I),GR,GRD)
      CALL GD(S,N(2,I),GS,GSD)
      H(1,I) = GR *GS
      H(2,I) = GRD*GS
  200 H(3,I) = GR *GSD
C-----MODIFY FOR NODES 5 TO 8 ----------------------------
      M=4
      DO 300 I=5,8
      IF(NODE(I)) 300,300,250
  250 M=M+1
      DO 290 K=1,3
      X=H(K,I)/2.D0
      H(K,M)=H(K,I)
      II=I-4
      H(K,II)=H(K,II)-X
      II=II+1
      IF(II.GT.4) II=1
  290 H(K,II)=H(K,II)-X
  300 CONTINUE
C------CALCULATE JACOBIAN MATRIX------------------
      DO 400 I=1,2
      DO 400 J=1,2
      SUM=0.0D0
      DO 350 K=1,NN
  350 SUM=SUM+H(I+1,K)*YZ(J,K)
  400 D(I,J)=SUM
C-----INVERT JACOBIAN MATRIX --------------------
      DJ=D(1,1)*D(2,2)-D(1,2)*D(2,1)
      IF(DJ.LE.0.0D0) WRITE(NWRITE,2000)DJ
      SUM=D(1,1)/DJ
      D(1,1)=D(2,2)/DJ
      D(1,2)=-D(1,2)/DJ
      D(2,1)=-D(2,1)/DJ
      D(2,2)=SUM
C-----CALCULATE H,Y AND H,Z ---------------------
      DO 500 K=1,NN
      HR=H(2,K)
      HS=H(3,K)
      H(2,K)=D(1,1)*HR+D(1,2)*HS
  500 H(3,K)=D(2,1)*HR+D(2,2)*HS
      RETURN
```

```
2000 FORMAT (10H JACOBIAN=  1PD15.6)                                STO18280
     END                                                              STO18290
     SUBROUTINE GD(B,IB,G,D)                                          STO18300
     IMPLICIT REAL*8 (A-H,O-Z)                                        STO18310
     IF(IB) 100,200,300                                               STO18320
 100 G=(1-B)/2.D0                                                     STO18330
     D=-.5D0                                                          STO18340
     RETURN                                                           STO18350
 200 G=1.D0-B*B                                                       STO18360
     D=-2.D0*B                                                        STO18370
     RETURN                                                           STO18380
 300 G=(1.D0+B)/2.D0                                                  STO18390
     D=.5D0                                                           STO18400
     RETURN                                                           STO18410
     END                                                              STO18420
     SUBROUTINE PTAPE(NN,NR)                                          STO18430
     IMPLICIT REAL*8 (A-H,O-Z)                                        STO18440
     COMMON /STR/ ST(9,16),XM(16),S(16,16),ND,NS,NRP,LM(16)           STO18450
     COMMON /PSAVE/ NUMNP,NEQ,NPROP,NUMEL,NTRUSS,NT,NSAVE             STO18460
     GO TO (50,600,700),NN                                            STO18470
C-----POSITION TAPE------------------------------                     STO18480
  50 IF(NR.NE.1) GO TO 100                                            STO18490
     REWIND NT                                                        STO18500
     NRP=1                                                            STO18510
 100 IF(NRP.EQ.NR) RETURN                                             STO18520
     NX=NR-NRP                                                        STO18530
     IF(NX.LT.0) GO TO 300                                            STO18540
     DO 200 N=1,NX                                                    STO18550
 200 READ (NT)                                                        STO18560
     GO TO 500                                                        STO18570
 300 NX=-NX                                                           STO18580
     DO 400 N=1,NX                                                    STO18590
 400 BACKSPACE NT                                                     STO18600
 500 NRP=NR                                                           STO18610
     RETURN                                                           STO18620
C-----WRITE  RECORD ON LOW SPEED STORAGE---------                     STO18630
 600 WRITE (NT) ST,XM,S,ND,NS,LM                                      STO18640
     NRP=NRP+1                                                        STO18650
     RETURN                                                           STO18660
 700 READ  (NT) ST,XM,S,ND,NS,LM                                      STO18670
     NRP=NRP+1                                                        STO18680
     RETURN                                                           STO18690
     END                                                              STO18700
     SUBROUTINE FORCE(D,P,NEQ,NL,NUME,NSS)                            STO18710
     IMPLICIT REAL*8 (A-H,O-Z)                                        STO18720
     DIMENSION D(NEQ,NL),P(NSS,NL)                                    STO18730
     COMMON /PSIZE/ NO,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR            STO18740
     COMMON /STR/ ST(9,16),XM(16),S(16,16),ND,NS,NRP,LM(16)           STO18750
     DO 600 N=1,NUME                                                  STO18760
```

```
      IF(NO.EQ.0)  WRITE(NWRITE,2000)  N,(I,I=1,NS)                     STO18770
      CALL PTAPE(3,NB)                                                  STO18780
      DO 500 L=1,NL                                                     STO18790
      DO 200 I=1,NS                                                     STO18800
      XM(I)=0.0D0                                                       STO18810
      DO 200 J=1,ND                                                     STO18820
      JJ=LM(J)                                                          STO18830
      IF(JJ.LE.0) GO TO 200                                             STO18840
      XM(I)=XM(I)+ST(I,J)*D(JJ,L)                                       STO18850
  200 CONTINUE                                                          STO18860
      IF(NO.EQ.0)  WRITE(NWRITE,2001)  L,(XM(I),I=1,NS)                 STO18870
      IF(NSS.EQ.0)  GO TO 500                                           STO18880
      M=NS*(N-1)+1                                                      STO18890
      DO 300 I=1,NS                                                     STO18900
      F(M,L)=XM(I)                                                      STO18910
  300 M=M+1                                                             STO18920
  500 CONTINUE                                                          STO18930
  600 CONTINUE                                                          STO18940
      RETURN                                                            STO18950
 2000 FORMAT  (12H0FOR ELEMENT I4,/,11H LOAD/FORCE,9(I7,6X))            STO18960
 2001 FORMAT  (1H ,I3,7X,9(1PD13.5))                                    STO18970
      END                                                               STO18980
      SUBROUTINE LOADI(L,NR,NC,N3,N4)                                   STO18990
      IMPLICIT REAL*8  (A-H,O-Z)                                        STO19000
      COMMON /TEMP/ FOR(40)                                             STO19010
      DIMENSION L(NR,NC)                                                STO19020
      COMMON /PSIZE/ NO                                                 STO19030
C----- SUBPROGRAM TO LOAD AND PRINT INTERGER ARRAY----                  STO19040
      IF(N3.NE.0) GO TO 100                                             STO19050
      READ(NREAD,1000)  ((L(I,J),J=1,NC),I=1,NR)                        STO19060
      GO TO 200                                                         STO19070
  100 READ(NREAD,1001)FOR                                               STO19080
      READ(NREAD,FOR) ((L(I,J),J=1,NC),I=1,NR)                          STO19090
  200 IF(NO.NE.0)  RETURN                                               STO19100
      IF(N4.LT.1 .OR. N4.GT.20)N4=20                                    STO19110
      DO 300 I=1,NC,N4                                                  STO19120
      IH = MIN0(I+N4-1,NC)                                              STO19130
      WRITE(NWRITE,2000)  (K,K=I,IH)                                    STO19140
      DO 300 J=1,NR                                                     STO19150
  300 WRITE(NWRITE,2001)(J,(L(J,K),K=I,IH))                             STO19160
      RETURN                                                            STO19170
 1000 FORMAT(16I5)                                                      STO19180
 1001 FORMAT  (40A2)                                                    STO19190
 2000 FORMAT  (5X,20I5)                                                 STO19200
 2001 FORMAT  (21I5)                                                    STO19210
      END                                                               STO19220
      SUBROUTINE DISPL(U,ID,NEQ,NL,NUMNP)                               STO19230
      IMPLICIT REAL*8  (A-H,O-Z)                                        STO19240
      DIMENSION U(NEQ,NL),ID(NUMNP,6)                                   STO19250
```

```
      COMMON /PSIZE/ NO,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR
      COMMON /TEMP/UU(6)
C
      DO 200 N=1,NUMNP
      WRITE(NWRITE,2000)
      DO 200 L=1,NL
      DO 100 I=1,6
      UU(I)=0.0D0
      II=ID(N,I)
      IF(II.LE.0) GO TO 100
      UU(I)=U(II,L)
  100 CONTINUE
  200 WRITE(NWRITE,2001)N,L,UU
      RETURN
 2000 FORMAT(12H  NODE  LOAD 13X 2HUX 13X 2HUY 13X 2HUZ 13X 2HTX
     1 13X 2HTY 13X 2HTZ )
 2001 FORMAT (2I6,6(1PD15.6))
      END
      SUBROUTINE LOADS(ID,R,NEQ,NL,NUMNP)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION R(NEQ,NL),ID(NUMNP,6)
      COMMON /TEMP/ RR(6)
      COMMON /PSIZE/ NO,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR
      DO 100 N=1,NEQ
      DO 100 L=1,NL
  100 R(N,L)=0.0D0
      IF(NO.EQ.0) WRITE(NWRITE,2000)
  200 READ(NREAD,1000) N,L,RR
      IF(N.EQ.0) RETURN
      IF(NO.EQ.0) WRITE(NWRITE,2001) N,L,RR
      DO 300 I=1,6
      II=ID(N,I)
      IF(II.LE.0) GO TO 300
      R(II,L)=RR(I)
  300 CONTINUE
      GO TO 200
 1000 FORMAT (2I5,6F10.0)
 2000 FORMAT (12H  NODE  LOAD  13X 2HFX 13X 2HFY 13X 2HFZ 13X 2HMX
     1 13X 2HMY 13X 2HMZ )
 2001 FORMAT (2I6,6(1PD15.6))
      END
      SUBROUTINE NODES(XYZ,NUMNP)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /PSIZE/ NO,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR
      DIMENSION XYZ(NUMNP,3)
C-----READ AND PRINT OF NODAL POINT COORDINATES---
      IF(NO.EQ.0) WRITE(NWRITE,2001)
  100 READ(NREAD,1000) N,X,Y,Z
      IF(N.EQ.0) RETURN
```

```
      XYZ(N,1)=X
      XYZ(N,2)=Y
      XYZ(N,3)=Z
      IF(ND.EQ.0) WRITE(NWRITE,2000)  N,XYZ(N,1),XYZ(N,2),XYZ(N,3)
      GO TO 100
 1000 FORMAT (I5,5X,3F10.0)
 2001 FORMAT (8H0NODE NO   14X,1HX,14X,1HY,14X,1HZ)
 2000 FORMAT(I8,3F15.3)
      END
      SUBROUTINE BOUND(ID,NUMNP,NEQ)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION ID(NUMNP,6),II(6)
      COMMON /PSIZE/ ND,NARA,NSIZE,NSP,NREAD,NWRITE
C-----ZERO ID ARRAY-------------------------------
      DO 100 N=1,NUMNP
      DO 100 I=1,6
  100 ID(N,I)=0
C-----SPECIFICATION OF UNKNOWN DISPLACEMENTS
  200 READ(NREAD,1000)NL,NH1,(II(I),I=1,6),INC
      IF(NL.EQ.0) GO TO 300
      IF(INC.EQ.0) INC=1
      IF(ND.EQ.0) WRITE(NWRITE,1000) NL,NH1,(II(I),I=1,6),INC
      DO 250 J=NL,NH1,INC
      DO 250 I=1,6
  250 ID(J,I)=II(I)
      GO TO 200
C-----EVALUATION OF EQUATION NUMBERS--------------
  300 NEQ=0
      DO 400 N=1,NUMNP
      DO 350 I=1,6
      IF(ID(N,I).EQ.0) GO TO 350
      NEQ=NEQ+1
      ID(N,I)=NEQ
  350 CONTINUE
  400 CONTINUE
      IF(ND.EQ.0) WRITE(NWRITE,2000) (N,(ID(N,I),I=1,6),N=1,NUMNP)
      RETURN
 1000 FORMAT (9I5)
 2000 FORMAT(42H0EQUATION NUMBERS FOR NODAL DISPLACEMENTS  /
     1 35H NODE     X     Y     Z    XX    YY    ZZ /(7I5))
      END
      SUBROUTINE GROUP3(LAST,IERR)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON MTOT,NDP,L(1)
      COMMON /PSIZE/ ND,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR
      COMMON /CARD/ INHOL(3,10),N1,N2,N3,N4,S1,S2
      LOGICAL COMP
      DIMENSION IOP(3,10)
      DATA IOP(1,1),IOP(2,1),IOP(3,1) /4HFUNG,4H    ,4H    /
```

```
      DATA IOP(1,2),IOP(2,2),IOP(3,2) /4HSTEP,4H    ,4H    /         ST020240
      DATA IOP(1,3),IOP(2,3),IOP(3,3) /4HEIGE,4HN   ,4H    /         ST020250
      DATA IOP(1,4),IOP(2,4),IOP(3,4) /4HDYNA,4HM   ,4H    /         ST020260
      DATA IOP(1,5),IOP(2,5),IOP(3,5) /4HPLOT,4H    ,4H    /         ST020270
      NUMOP=5                                                        ST020280
      GO TO 175                                                      ST020290
C---- READ OPERATION FROM CARD OR STORAGE ----------------           ST020300
  100 CALL INPUT(IERR)                                               ST020310
      IF(IERR .GT. 1) RETURN                                         ST020320
C-----INTERPRETE OPERATION ----------------------                    ST020330
  175 DO 200 J=1,NUMOP                                               ST020340
      N=J                                                            ST020350
      IF (COMP(INHOL(1,1),IOP(1,J))) GO TO 300                       ST020360
  200 CONTINUE                                                       ST020370
      RETURN                                                         ST020380
C-----EXECUTE APPROPRIATE OPERATION --------------                   ST020390
  300 LAST=3                                                         ST020400
      GO TO (1,2,3,4,5),N                                            ST020410
    1 CALL FUNG(IERR)                                                ST020420
      IF(IERR .GT. 1) RETURN                                         ST020430
      GO TO 100                                                      ST020440
    2 CALL STEP(IERR)                                                ST020450
      IF(IERR .GT. 1) RETURN                                         ST020460
      GO TO 100                                                      ST020470
    3 CALL EIGEN(IERR)                                               ST020480
      IF(IERR .GT. 1) RETURN                                         ST020490
      GO TO 100                                                      ST020500
    4 CALL DYNAM(IERR)                                               ST020510
      IF(IERR .GT. 1) RETURN                                         ST020520
      GO TO 100                                                      ST020530
    5 CALL PLOT(IERR)                                                ST020540
      IF(IERR .GT. 1) RETURN                                         ST020550
      GO TO 100                                                      ST020560
      END                                                            ST020570
      SUBROUTINE FUNG(IERR)                                          ST020580
      IMPLICIT REAL*8 (A-H,O-Z)                                      ST020590
      COMMON MTOT,NDP,L(1)                                           ST020600
      COMMON /PSIZE/ NO,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR          ST020610
      COMMON /CARD/ INHOL(3,10),N1,N2,N3,N4,S1,S2                    ST020620
      COMMON /ADIR/ NA,I(10)                                         ST020630
      CALL DELETE(INHOL(1,3))                                        ST020640
      IF(N1.LE.0) CALL ERCOM(IERR)                                   ST020650
      IF(IERR .GT. 1) RETURN                                         ST020660
      IF(N2.NE.0) N2=1                                               ST020670
      N2=N2+1                                                        ST020680
      CALL LIST(INHOL(1,3),N2,N1,NDP,IERR)                           ST020690
      N3=NA                                                          ST020700
      CALL FIND(INHOL(1,4),IERR)                                     ST020710
      IF(IERR .GT. 1) RETURN                                         ST020720
```

```
      N4=NA
      CALL FIND(INHOL(1,2),IERR)
      IF(IERR .GT. 1) RETURN
      IF(I(4).NE.2) CALL ERCOM(IERR)
      IF(IERR .GT. 1) RETURN
      CALL FUNGS(L(NA),L(N3),I(3),N1,N2,L(N4))
      RETURN
      END
      SUBROUTINE FUNGS(G,GG,NC,N1,N2,DT)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION G(2,NC),GG(N2,N1)
      T=G(1,1)
      J=1
      NCC=NC-1
      DO 200 I=1,NCC
      S=(G(2,I+1)-G(2,I))/(G(1,I+1)-G(1,I))
  100 GG(1,J)=T
      GG(N2,J)=G(2,I)+S*(T-G(1,I))
      IF(J.EQ.N1) RETURN
      J=J+1
      T=T+DT
      IF(T.LT.G(1,I+1)) GO TO 100
  200 CONTINUE
      CALL ERCOM(IERR)
      RETURN
      END
      SUBROUTINE STEP(IERR)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON MTOT,NDP,L(1)
      COMMON /PSIZE/ NO,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR
      COMMON /CARD/ INHOL(3,10),N1,N2,N3,N4,S1,S2
      COMMON /ADIR/ NA,I(10)
      CALL DELETE(INHOL(1,6))
      CALL FIND(INHOL(1,2),IERR)
      IF(I(3).NE.I(4)) CALL ERCOM(IERR)
      IF(IERR .GT. 1) RETURN
      NK=NA
      N=I(3)
      CALL FIND(INHOL(1,3),IERR)
      IF(I(3).NE.I(4)) CALL ERCOM(IERR)
      IF(IERR .GT. 1) RETURN
      NM=NA
      CALL FIND(INHOL(1,4),IERR)
      IF(I(3).NE.I(4)) CALL ERCOM(IERR)
      IF(IERR .GT. 1) RETURN
      NC=NA
      CALL FIND(INHOL(1,5),IERR)
      IF(IERR .GT. 1) RETURN
      NI=NA
```

```
      CALL LIST(INHOL(1,6),N,N2,NDP,IERR)                                ST021220
      IF(IERR .GT. 1) RETURN                                             ST021230
      NU=NA                                                              ST021240
      READ(NREAD,1000) DEL,ALF,THE                                       ST021250
      IF(NO.EQ.0) WRITE(NWRITE,2000) DEL,ALF,THE                         ST021260
      CALL FIND(INHOL(1,7),IERR)                                         ST021270
      IF(IERR .GT. 1) RETURN                                             ST021280
      ND=NA                                                              ST021290
      CALL FIND(INHOL(1,8),IERR)                                         ST021300
      IF(IERR .GT. 1) RETURN                                             ST021310
      NF=NA                                                              ST021320
      CALL FIND(INHOL(1,9),IERR)                                         ST021330
      IF(IERR .GT. 1) RETURN                                             ST021340
      CALL STEPS(L(NK),L(NM),L(NC),L(NI),L(ND),L(NF),L(NU),N,N1,N2,L(NA) ST021350
     1,DEL,ALF,THE)                                                      ST021360
      RETURN                                                             ST021370
 1000 FORMAT(3F10.0)                                                     ST021380
 2000 FORMAT(2X,4HDEL=1PD15.6,3X4HALF=1PD15.6,3X,4HTHE=1PD15.6)          ST021390
      END                                                                ST021400
      SUBROUTINE STEPS(S,XM,C,UI,D,F,U,N,N1,N2,DTT,DEL,ALF,THE)          ST021410
      IMPLICIT REAL*8 (A-H,O-Z)                                          ST021420
      DIMENSION S(N,N),XM(N,N),C(N,N),UI(N,3),U(N,N2),D(1),F(1)          ST021430
C---- COMPUTE INTEGRATION CONSTANTS--------------------                  ST021440
      IF(THE.EQ.0.0D0) THE=1.0D0                                         ST021450
      DT=THE*DTT                                                         ST021460
      A0=1.D0/(ALF*DT*DT)                                                ST021470
      A1=DEL/(ALF*DT)                                                    ST021480
      A2=1.D0/(ALF*DT)                                                   ST021490
      A3=0.5D0/ALF-1.D0                                                  ST021500
      A4=DEL/ALF -1.D0                                                   ST021510
      A5=0.5D0*DT*(DEL/ALF-2.D0)                                         ST021520
      A6=DTT*(1.D0-DEL)                                                  ST021530
      A7=DTT*DEL                                                         ST021540
      A8=(.5D0-ALF)*DTT*DTT                                              ST021550
      A9=ALF*DTT*DTT                                                     ST021560
C---- FORM AND TRIANGULARIZE EFFECTIVE STIFFNESS MATRIX--------           ST021570
      DO 100 I=1,N                                                       ST021580
      DO 100 J=1,N                                                       ST021590
  100 S(I,J)=S(I,J)+A0*XM(I,J)+A1*C(I,J)                                 ST021600
      CALL SYMSOL(S,UI,N,1,1,0)                                          ST021610
C---- FOR EACH TIME STEP--------------------------------                 ST021620
      K=1                                                                ST021630
      DO 400 I=1,N2                                                      ST021640
      DO 400 J=1,N1                                                      ST021650
C      1. CALCULATE EFFECTIVE LOAD AT TIME T+DT                          ST021660
      K=K+1                                                              ST021670
      FF=F(K-1)+THE*(F(K)-F(K-1))                                        ST021680
      DO 200 L=1,N                                                       ST021690
  200 U(L,I)=D(L)*FF                                                     ST021700
```

```
      DO 250 L=1,N                                                      STO21710
      X=A0*UI(L,1)+A2*UI(L,2)+A3*UI(L,3)                                STO21720
      Y=A1*UI(L,1)+A4*UI(L,2)+A5*UI(L,3)                                STO21730
      DO 250 M=1,N                                                      STO21740
  250 U(M,I)=U(M,I)+XM(M,L)*X+C(M,L)*Y                                  STO21750
C       2. SOLVE FOR DISPLACEMENT AT T+DT                               STO21760
      CALL SYMSOL(S,U(1,I),N,1,2,0)                                     STO21770
C       3. CALCULATE ACCELERATIONS AND VELOCITIES AT TIME T+DT          STO21780
      DO 300 L=1,N                                                      STO21790
      A=A0*(U(L,I)-UI(L,1))-A2*UI(L,2)-A3*UI(L,3)                       STO21800
      DA=(A-UI(L,3))/THE                                                STO21810
      A=UI(L,3)+DA                                                      STO21820
      V=UI(L,2)+A6*UI(L,3)+A7*A                                         STO21830
      U(L,I)=UI(L,1)+DTT*UI(L,2)+A8*UI(L,3)+A9*A                        STO21840
      UI(L,3)=A                                                         STO21850
      UI(L,2)=V                                                         STO21860
  300 UI(L,1)=U(L,I)                                                    STO21870
  400 CONTINUE                                                          STO21880
C                                                                       STO21890
      RETURN                                                            STO21900
      END                                                               STO21910
      SUBROUTINE DYNAM(IERR)                                            STO21920
      IMPLICIT REAL*8 (A-H,O-Z)                                         STO21930
      COMMON MTOT,NDP,L(1)                                              STO21940
      COMMON /PSIZE/ NO,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR             STO21950
      COMMON /CARD/ INHOL(3,10),N1,N2,N3,N4,S1,S2                       STO21960
      COMMON /ADIR/ NA,I(10)                                            STO21970
C-----SUBROUTINE TO EVALUATE MODAL DYNAMIC RESPONSE--------              STO21980
      CALL DELETE(INHOL(1,6))                                           STO21990
      CALL FIND(INHOL(1,2),IERR)                                        STO22000
      IF(IERR .GT. 1) RETURN                                            STO22010
      N2=NA                                                             STO22020
      N=I(3)*I(4)                                                       STO22030
      CALL FIND(INHOL(1,3),IERR)                                        STO22040
      IF(IERR .GT. 1) RETURN                                            STO22050
      N3=NA                                                             STO22060
      N5=I(3)*I(4)                                                      STO22070
      IF(N.NE.N5) CALL ERCOM(IERR)                                      STO22080
      IF(IERR .GT. 1) RETURN                                            STO22090
      CALL FIND(INHOL(1,4),IERR)                                        STO22100
      IF(IERR .GT. 1) RETURN                                            STO22110
      N4=NA                                                             STO22120
      N5=I(3)*I(4)                                                      STO22130
      IF(N.NE.N5) CALL ERCOM(IERR)                                      STO22140
      IF(IERR .GT. 1) RETURN                                            STO22150
      CALL FIND(INHOL(1,5),IERR)                                        STO22160
      IF(IERR .GT. 1) RETURN                                            STO22170
      N5=NA                                                             STO22180
      CALL FIND(INHOL(1,7),IERR)                                        STO22190
```

```
      IF(IERR .GT. 1) RETURN
      N6=NA
      CALL LIST(INHOL(1,6),N,N1,NDP,IERR)
      IF(IERR .GT. 1) RETURN
      CALL DYNAMS(L(N2),L(N3),L(N4),L(N5),L(NA),N1,N,L(N6))
      RETURN
      END
      SUBROUTINE DYNAMS(FQ,DAM,P,PA,X,NTIME,NM,DDT)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION FQ(NM),DAM(NM),P(NM),X(NM,NTIME),PA(2,1)
C     EVALUATION OF MODAL RESPONSE ... USING EXPLICIT INTEGRATION
      DO 700 M=1,NM
      W=FQ(M)
      DAMP=DAM(M)
      WW=W*W
      ZW=DAMP*W
      TZW=2.D0*ZW
      WD=W*DSQRT(1.0D0-DAMP*DAMP)
      FB=TZW/(WW*WW)
      FA=ZW/WW
      FV=ZW*WW
      FVD=WW*(2.D0*DAMP*DAMP-1.0D0)
      FBB=(2.D0*DAMP*DAMP-1.0D0)/WW
      L=0
      II=1
      VO=0.D0
      VDO=0.D0
      TO=PA(1,1)
C
   10 DT=DDT
   50 B=(PA(2,II+1)-PA(2,II))/(PA(1,II+1)-PA(1,II))
      B=B*P(M)
      A=P(M)*PA(2,II)+B*(TO-PA(1,II))
      TT=TO+DT
      IF(PA(1,II+1).GT.TT) GO TO 100
      DELT=PA(1,II+1)-TO
      GO TO 200
  100 DELT=DT
C
  200 EX=DEXP(-ZW*DELT)
      FT=WD*DELT
      CS=DCOS(FT)
      SN=DSIN(FT)
      VT=(VDO+ZW*VO-FA*A+FBB*B)*SN/WD
      VT=VT+(VO-A/WW+FB*B)*CS
      VT=VT*EX+A/WW-FB*B+B*DELT/WW
      VDT=(A-WW*VO-ZW*(VDO+B/WW))*SN/WD
      VDT=EX*((VDO-B/WW)*CS+VDT)+B/WW
      VDDT=(B+FV*VO+FVD*VDO-ZW*A)/WD
```

```
      VDDT=EX*((A-WW*VO-TZW*VDO)*CS+VDDT*SN)                    STO22690
      VO=VT                                                     STO22700
      VDO=VDT                                                   STO22710
C                                                               STO22720
      IF(PA(1,II+1).GT.TT)  GO TO 500                           STO22730
      DT=DT-DELT                                                STO22740
      II=II+1                                                   STO22750
      TO=PA(1,II)                                               STO22760
      IF(DT.EQ.0.DO)  GO TO 600                                 STO22770
      GO TO 50                                                  STO22780
  500 TO=TO+DT                                                  STO22790
  600 L=L+1                                                     STO22800
      X(M,L)=VT                                                 STO22810
      IF(L.LT.NTIME)  GO TO 10                                  STO22820
  700 CONTINUE                                                  STO22830
C                                                               STO22840
      RETURN                                                    STO22850
      END                                                       STO22860
      SUBROUTINE EIGEN(IERR)                                    STO22870
      IMPLICIT REAL*8  (A-H,O-Z)                                STO22880
      COMMON  MTOT,NDP,L(1)                                     STO22890
      COMMON /PSIZE/ NO,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR     STO22900
      COMMON /CARD/ INHOL(3,10),N1,N2,N3,N4,S1,S2               STO22910
      COMMON /ADIR/ NA,I(10)                                    STO22920
      CALL DELETE(INHOL(1,3))                                   STO22930
      CALL FIND(INHOL(1,2),IERR)                                STO22940
      IF(I(3).NE.I(4))  CALL  ERCOM(IERR)                       STO22950
      IF(IERR .GT. 1)  RETURN                                   STO22960
      N2=I(4)                                                   STO22970
      N3=NA                                                     STO22980
      CALL FIND(INHOL(1,4),IERR)                                STO22990
      IF(I(1).NE.N2) CALL ERCOM(IERR)                           STO23000
      IF(IERR .GT. 1) RETURN                                    STO23010
      N4=NA                                                     STO23020
      CALL LIST(INHOL(1,3),N2,N2,NDP,IERR)                      STO23030
      IF(IERR .GT. 1) RETURN                                    STO23040
      CALL EIGENS(L(N3),L(NA),L(N4),N2,N1,N3)                   STO23050
      IF(NO.EQ.0) WRITE(NWRITE,2000)  N1,N3                     STO23060
      RETURN                                                    STO23070
 2000 FORMAT(I3,27H FIGURE TOLERANCE SPECIFIED  / I6,           STO23080
     1 20H ROTATIONS PERFORMED )                                STO23090
      END                                                       STO23100
      SUBROUTINE EIGENS(H,U,E,N,NS,NR)                          STO23110
      IMPLICIT REAL*8  (A-H,O-Z)                                STO23120
      COMMON /PSIZE/ NO,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR     STO23130
      DIMENSION H(N,1),U(N,1),E(N)                              STO23140
C                                                               STO23150
      IF(NS.EQ.0)  NS=4                                         STO23160
      TEST=1.0D0/10.D0**(2*NS)                                  STO23170
```

```
      NN=N-1                                                            STO23180
      NR=0                                                              STO23190
      NRLM=5*N*N                                                        STO23200
      TOLER=0.1D0                                                       STO23210
      ZERO = 0.0D0                                                      STO23220
C-----NORMALIZE TO UNIT MATRIX--------------------                      STO23230
      DO 10 I=1,N                                                       STO23240
   10 E(I)=1.0D0/DSQRT(E(I))                                            STO23250
      DO 30 I=1,N                                                       STO23260
      DO 20 J=1,N                                                       STO23270
      H(I,J)=E(I)*H(I,J)*E(J)                                           STO23280
   20 U(I,J)=0.0D0                                                      STO23290
      ZERO = ZERO + DABS(H(I,I))                                        STO23300
   30 U(I,I)=1.0D0                                                      STO23310
      ZERO = ZERO*1.0D-16                                               STO23320
C---- REDUCE MATRIX ----------------------------                         STO23330
   50 XMAX=0.0D0                                                        STO23340
      DO 700 II=1,NN                                                    STO23350
      JL=II+1                                                           STO23360
      DO 600 JJ=JL,N                                                    STO23370
C-----CHECK IF ROTATION IS REQUIRED   ------------                      STO23380
      HII=H(II,II)                                                      STO23390
      HIJ=H(II,JJ)                                                      STO23400
      HJJ=H(JJ,JJ)                                                      STO23410
      D=DABS(HII*HJJ)                                                   STO23420
      H2=HIJ*HIJ                                                        STO23430
      IF(H2.GT.XMAX*D) XMAX=H2/D                                        STO23440
      IF(H2.LT.TOLER*D) GO TO 600                                       STO23450
C-----COMPUTE TAN, SIN AND COS ------------------                        STO23460
      NR=NR+1                                                           STO23470
      IF(DABS(HIJ) - ZERO)300,300,320                                   STO23480
  300 CS = 0.0D0                                                        STO23490
      SN = 1.0D0                                                        STO23500
      GO TO 350                                                         STO23510
  320 HT=.5D0*(HII-HJJ)/HIJ                                             STO23520
      TN=-HT-DSIGN(DSQRT(HT*HT+1.0D0),HT)                               STO23530
      CS =1.0D0/DSQRT(1.D0+TN *TN)                                      STO23540
      SN=CS*TN                                                          STO23550
  350 C2=CS*CS                                                          STO23560
      S2=SN*SN                                                          STO23570
C-----REDUCE II,JJ ELEMENT TO ZERO --------------                        STO23580
      HT=2.D0*HIJ*CS*SN                                                 STO23590
      H(II,JJ)=0.0D0                                                    STO23600
      H(II,II)=HII*C2+HT+HJJ*S2                                         STO23610
      H(JJ,JJ)=HII*S2-HT+HJJ*C2                                         STO23620
      DO 530 I=1,N                                                      STO23630
      IF(I-II) 370,530,420                                              STO23640
  370 HT=H(I,II)                                                        STO23650
      H(I,II)=CS *HT+SN *H(I,JJ)                                        STO23660
```

```
      H(I,JJ)=-SN *HT+CS *H(I,JJ)
      GO TO 530
  420 IF(I-JJ) 430,530,480
  430 HT=H(II,I)
      H(II,I)=CS *HT+SN *H(I,JJ)
      H(I,JJ)=-SN *HT+CS *H(I,JJ)
      GO TO 530
  480 HT=H(II,I)
      H(II,I)=CS *HT+SN *H(JJ,I)
      H(JJ,I)=-SN *HT+CS *H(JJ,I)
  530 CONTINUE
C-----OPERATE ON EIGENVECTORS --------------------
  540 DO 550 I=1,N
      HT=U(I,II)
      U(I,II)=CS *HT+SN *U(I,JJ)
  550 U(I,JJ)=-SN *HT+CS *U(I,JJ)
  600 CONTINUE
  700 CONTINUE
C-----TEST FOR END OF ITERATION AND SET NEW TOLERANCE
      IF(NRLM.LT.NR) GO TO 1000
      IF(XMAX.LT.TEST) GO TO 710
      TOLER=0.10D0*XMAX
      GO TO 50
C-----NORMALIZE AND ORDER EIGENVECTORS -----------
  710 DO 800 I=1,N
      DO 750 J=1,N
  750 U(I,J)=U(I,J)*E(I)
  800 E(I)=H(I,I)
      DO 900 I=1,NN
C-----ORDER EIGENVALUES AND EIGENVECTORS ---------
      JL=I+1
      HT=E(I)
      IM=I
      DO 850 J=JL,N
      IF(HT.LT.E(J)) GO TO 850
      HT=E(J)
      IM=J
  850 CONTINUE
      E(IM)=E(I)
      E(I)=HT
      DO 900 J=1,N
      HT=U(J,I)
      U(J,I)=U(J,IM)
  900 U(J,IM)=HT
      RETURN
C
 1000 WRITE(NWRITE,2000)
      RETURN
 2000 FORMAT(41H0ITERATION TERMINATED WITHOUT CONVERGENCE )
```

```
      END                                                               STO24160
      SUBROUTINE PLOT(IERR)                                             STO24170
      IMPLICIT REAL*8 (A-H,O-Z)                                         STO24180
      COMMON MTOT,NDP,L(1)                                              STO24190
      COMMON /PSIZE/ NO,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR             STO24200
      COMMON /CARD/ INHOL(3,10),N1,N2,N3,N4,S1,S2                       STO24210
      COMMON /ADIR/ NA,I(10)                                            STO24220
      DIMENSION IPLOT(3)                                                STO24230
      DATA IPLOT /4HPLOT,4HNM  ,4H    /                                 STO24240
      NT=121+N1+N1                                                      STO24250
      CALL LIST(IPLOT,1,NT,NSP,IERR)                                    STO24260
      IF(IERR .GT. 1) RETURN                                            STO24270
      NP=NA                                                             STO24280
      NK=NA+121                                                         STO24290
      NN=NK+N1                                                          STO24300
      CALL FIND(INHOL(1,2),IERR)                                        STO24310
      IF(IERR .GT. 1) RETURN                                            STO24320
      CALL PLOTS(L(NA),L(NP),L(NK),L(NN),I(4),I(3),N1)                  STO24330
      CALL DELETE(IPLOT)                                                STO24340
      RETURN                                                            STO24350
      END                                                               STO24360
      SUBROUTINE PLOTS(A,K,KODE,NROW,NR,NC,N1)                          STO24370
      IMPLICIT REAL*8 (A-H,O-Z)                                         STO24380
      COMMON /PSIZE/ NO,NARA,NSIZE,NSP,NREAD,NWRITE,NH,NDIR             STO24390
      DIMENSION A(NR,NC),K(121),KODE(N1),NROW(N1)                       STO24400
C---- SUBROUTINE TO PRINTER PLOT N1 ROWS OF ARRAY A                     STO24410
      DATA INDIA /2HI /,IBLANK /2H  /,IZERO /2H0 /                      STO24420
      READ(NREAD,1000) (KODE(I),NROW(I),I=1,N1)                         STO24430
      IF(NO.NE.0) RETURN                                                STO24440
      WRITE(NWRITE,2000) (KODE(I),NROW(I),I=1,N1)                       STO24450
C----- LOCATE LARGEST AND SMALLEST ELEMENTS-------                      STO24460
      II=NROW(1)                                                        STO24470
      AL=A(II,1)                                                        STO24480
      AS=AL                                                             STO24490
      DO 100 I=1,N1                                                     STO24500
      II=NROW(I)                                                        STO24510
      DO 100 J=1,NC                                                     STO24520
      AX=A(II,J)                                                        STO24530
      IF(AX.GT.AL) AL=AX                                                STO24540
      IF(AX.LT.AS) AS=AX                                                STO24550
  100 CONTINUE                                                          STO24560
      AX=(AL-AS)/120.D0                                                 STO24570
      WRITE(NWRITE,2001)AX,AS,AL                                        STO24580
      KO=-AS/AX +1.D0                                                   STO24590
      DO 500 I=1,NC                                                     STO24600
C---- FILL LINE BUFFER WITH BLANKS AND CODES---                         STO24610
      DO 300 J=1,121                                                    STO24620
  300 K(J)=IBLANK                                                       STO24630
      K(1)=INDIA                                                        STO24640
```

```
      K(121)=INDIA                                                       STO24650
      IF(K0.GT.0) K(K0)=IZERO                                            STO24660
      DO 400 J=1,N1                                                      STO24670
      JJ=NROW(J)                                                         STO24680
      II=(A(JJ,I)-AS)/AX+1.D0                                            STO24690
  400 K(II)=KODE(J)                                                      STO24700
  500 WRITE(NWRITE,2003)K                                                STO24710
      WRITE(NWRITE,2004)                                                 STO24720
      RETURN                                                             STO24730
 1000 FORMAT (1A1,I4)                                                    STO24740
 2000 FORMAT (11H0SYMBOL ROW /(5X,1A1,I5))                               STO24750
 2001 FORMAT (11H ONE SPACE= 1PD15.6/9H0MINIMUM= 1PD15.6,74X 9HMAXIMUM= STO24760
     1 1PD15.6/  122(1H0))                                               STO24770
 2003 FORMAT(1H ,121A1)                                                  STO24780
 2004 FORMAT(122(1H0))                                                   STO24790
      END                                                                STO24800
```

APPENDIX B - USER'S MANUAL

This appendix provides details on use of CAL with the IBM 360/67 computer at NPS. Section I provides details on the command structure. Section II is a summary of commands available. Section III provides the job control language for executing the program in both the batch and interactive modes at NPS. Section IV contains detailed specifications for each available command. Finally, section V gives direction for solving larger problems with CAL. The majority of this appendix was originally published as reference (3). The author wishes to express appreciation to Professor Wilson for permission to use this material.

I. FORM AND RESTRICTION OF THE LANGUAGE

CAL is an interpretive language which is designed to manipulate arrays and matrices and to perform several standard structural analysis operations. A CAL program run involves the reading of the input deck once and executing the commands designated by the operation cards as they are encountered. Looping operations allow a sequence of commands to be executed more than once.

The input deck is composed of operation cards and data cards. The data cards directly follow each operation card which requires data (see LOOP operation for exception to this). The operation card contains the name of the operation to be executed, names of arrays associated with the operation and integer constants. Examples of the general form of this card are

```
OP,M1,M2,M3,M4,M5,N1,N2,N3,N4  COMMENTS
OP,M1,N1,N2
OP,N1
OP
```

in which OP is the name of the operation to be executed, Mi is the name of an array and Ni is an integer. The names of OP or Mi are one to eight alphabetic or numeric characters to be selected by the user. The first character of a name must be alphabetic. The sequence of terms OP,Mi and Ni must be separated by commas. Characters following a blank will be printed as comments in the output from the program run.

If an operation attempts to load or generate an array which previously existed the program will delete the array before the execution of the operation. A new array need not be the same size of the old array which had the same name.

II. SUMMARY OF COMMANDS

A. GENERAL COMMANDS

* indicates a significant change or addition in CAL-NPS

START	- Initialize for the next problem	
STOP	- Normal termination	
NO	- Temporary suppression of output	
YES	- Restores output	
LABEL	- Print comments	
READ	- Change logical device for input	*
WRITE	- Change logical device for output	*
TIME	- Suppress time printout	*
SAVE	- Interrupt a problem	*
RESUME	- Continue an interrupted problem	*
LIST	- List arrays and storage used	*

B. GENERAL MATRIX COMMANDS

LOAD	- Load user defined matrix	
ZERO	- Create null or unit matrix	
PRINT	- Matrix print operation	*
DUP	- Matrix duplication	
ADD	- Matrix addition	
SUB	- Matrix subtraction	
MULT	- Matrix multiplication	
TRAN	- Matrix transpose	
SCALE	- Multiply a matrix by a scalar	
SOLVE	- Solution of linear equations	*
DUPSM	- Form sub-matrix from large matrix	
STOSM	- Store sub-matrix in large matrix	
DUPDG	- Form row matrix from diagonal	

STODG - Store row on diagonal

MAX - Evaluate row maximums

NORM - Evaluate matrix norms

INVEL - Inverte each term in matrix

SQREL - Square root of each term in matrix

LOG - Natural log of each term in matrix

PROD - Evaluate product of all terms in matrix

DELETE - Delete matrix from storage

C. STATIC ANALYSIS OPERATIONS

NODES - Input joint geometry

BOUND - Specify boundary conditions

BEAM - Form 3-D beam stiffness matrix

TRUSS - Form 3-D truss stiffness matrix

PLANE - Form 3 to 8 node plane stiffness matrix

SLOPE - Form stiffness matrix from slope/deflection eq.

FRAME - Form 2-D frame stiffness matrix

LOADI - Load integer array (connectivity matrix) *

LOADS - Form load vector

ADDSF - Form global stiffness and mass matrices

ADDK - Add element matrix to global matrix

MEMFRC - Calculate element forces from joint displacements

DISPL - Print joint displacements

FORCE - Evaluate and print member forces

D. DYNAMIC ANALYSIS OPERATIONS

FUNG - Generate equal interval time function

STEP - Integrate dynamic equilibrium equations

EIGEN - Evaluate mode shapes and frequencies

DYNAM - Evaluate uncoupled equations of motion by mode superposition method

PLOT - Line printer plot of joint time history

E. LOOP OPERATIONS

LOOP - Start of loop

NEXT - End of loop

SKIP - Conditional skip of operations within loop

F. NAMES AVAILABLE FOR USER SUBROUTINES

USERA

USERB

III. JCL FOR EXECUTION

The job control language (JCL) necessary to initiate program execution for the OS/MVT (batch) system is:

```
//(standard green job card)
//  EXEC  CAL
//SYSIN DD *


CAL  COMMANDS AND DATA CARDS


/*
```

For the CP/CMS (time-sharing) system the procedure is:

```
use standard login procedure, then
CP LINK 0099P 191 199 PASS= ESAN
LOGIN 199 A,P
CAL


CAL  COMMANDS AND DATA CARDS
```

The following information is provided for the user who wishes to use his own subroutine or solve larger problems then allowed with the default core allocation. The load module library for the OS/MVT operating system is in partitioned data set F0099.CAL on DISK02. The load module is assigned member name MODULE. The file used as a scratch

pad is FT01F001. File FT02F001 is used as the scratch file for the SAVE and RESUME operations. The FT01F001 file is currently pointed at the SYSDA disk and FT02F001 is pointed at dummy. If for any reason a user desires to use the SAVE operation with the OS/MVT system, both FT01F001 and FT02F001 must be redifined. See Computer Center TN No. 0141-05, USER LIBRARIES AND SOURCE CODE EDITING UNDER OS for procedures. File FT01F001 requires 3404 bytes of storage per element. File FT02F001 requires a total of 4 times the dimension of the L array plus 8 bytes and should have RECFM=VBS.

For CP/CMS, the execution routine is

```
&TYPEOUT OFF
CP SET LINELN 130
GLOBAL T SYSLIB SSPLIB
FILEDEF 01  DSK-P4 FILE  FT01F001 RECFM  VS LRECL  3408
BLKSIZE 3412
FILEDEF 02  DSK-P4 FILE  FT02F001 RECFM  VS LRECL  7290
BLKSIZE 7294
FILEDEF 04 DSK
FILEDEF 08 PRT RECFM FBA LRECL 133 (PERM)
LOAD STORE USER CALH (CLEAR NOMAP XEQ)
```

FT01F001 is the scratch pad for the structural analysis. FT02F001 is created by the SAVE operation. Both FT01F001 and FT02F001 must be available for the RESUME operation.

Temporary additional storage can be obtained, if needed, by typing GETTEMP while in CMS. The GETTEMP execution routine is:

```
&TYPEOUT OFF
VSET RDYMSG OFF
CP SET LINELN 130
&BEGSTACK
VSET BLIP +
```

```
CP DEFINE T2314 192 4
FORMAT T ALL (NOTYPE)
RELEASE 192 T
LOGIN 192 P
LOGIN 191 B,P
&ENDSTACK
```

IV. CAL COMMAND SPECIFICATIONS

A. GENERAL MATRIX OPERATIONS

CAL has most of the standard matrix operations plus some special array operations which are useful in structural analysis. The following is a list of approximately 25 operations which are used for control and general matrix manipulation.

\+ indicates the formation of a new matrix. A matrix previously defined with the same name will be deleted.

\- indicates modification of an existing matrix.

START

This operation eliminates all arrays which were perviously loaded or generated.

STOP

This operation causes normal termination of a CAL program.

NO
YES

These operations are used to selectively suppress output from CAL. The NO operations suppresses all printing, except diagnostics, until the operation YES is encountered. Therefore, in subsequent runs of the same CAL program, output which was previously correct need not be reprinted if these cards are inserted in the data deck.

LABEL,N1

This operation will read and print N1 comment cards which follow this operation card. Column 1 of each card will be interpreted as a standard carriage control symbol (i.e. 0 for double space and 1 for skip to the top of the next page).

READ,N1

THIS OPERATION IS VALID ONLY WITH THE CP/CMS TIME SHARING SYSTEM.
This operation permits the selection of the offline printer or the terminal as the input file device. Default is the terminal. N1 = 4 will read subsequent commands from FILE FT04F001 on the users p-disk. N1 = 5 will restore the terminal as the input file device. All disk files prepared for use with this command should end with either STOP or READ,4. This command will not be executed on the OS/MVT (batch) system.

WRITE,N1

THIS OPERATION IS VALID ONLY WITH THE CP/CMS TIME SHARING SYSTEM¬
This operation permits the selection of the offline printer or the terminal as the output file device. Default is the terminal and all error messages will be printed at the terminal regardless of the output file device selected. N1 = 3 selects the offline printer. N1 = 6 restores the terminal as the output file device. This command will not be executed on the OS/MVT (batch) system.

TIME

This operation permits the time printout to be suppressed without loss of other output. A second TIME will restore the time printout unless the print output is suppressed with the NO command.

SAVE

The SAVE command creates FILE FT02F001 containing all arrays in storage at the time of issuance. Note that saved arrays will contain any modification since creation. For instance, if a matrix has been reduced by the SOLVE operation, the reduced form of the matrix will be stowed in FILE FT02F001. Note the requirements discussed previously for use of this command with the OS system.

RESUME

The RESUME operation reads FILE FT02F001 into memory. Any arrays currently in storage will be destroyed. FILE FT02F001 must have previously been created on a mass storage device using the SAVE operation. Note that FILE FT01F001 must also be accessible if an interrupted structural analysis problem is being resumed.

LIST

The LIST operation prints the directory information for arrays in storage and the amount of storage used.

```
   +
LOAD,M1,N1,N2,N3
```

This operation will load an array of real numbers named M1 which has N1 rows and N2 columns. The terms of the array are punched in row-wise sequence on data cards following this operation. If N3 is zero or blank, the cards are punched in a format of (8F10.0). If N3 is nonzero, an additional card containing the format of the data cards must follow this operation and preceed the data cards. If the data is to be 4 numbers per card in field widths of 15, the additional cards would contain the following information: (4F15.0).

```
   +
ZERO,M1,N1,N2,N3,N4
```

A real matrix named M1 is created with N1 rows and N2 columns. The terms in this matrix will have the following values.

```
          M1(I,I) = N3    I = 1,...N1
          M1(I,J) = N4    J = 1,...N2
```

Therefore, this operation can be used to form null or unit matrices.

```
PRINT,M1 or PRINT,M1,N1 or PRINT,M1,N1,N2 or PRINT,M1,,N2
```

This operation will print the real array named M1 in a matrix format of up to 8 columns per line. If N1 is greated than zero the operation will read and print N1 comment cards wich follow the operation card. N2 is optional but note that an extra comma must be used in place of N1 if no printed title is desired. The matrix M1 will be printed in partitioned form with N2 columns per partition. Lines will have N1*15+5 characters. N2 defaults to 8, printing 125 characters per line. The user is cautioned not to overcome the capacity of the printing device in use.

```
   +
DUP,M1,M2
```

This operation will form an array named M2 which is identical to the array named M1.

```
   -
ADD,M1,M2
```

This operation will replace matrix M1 with the sum of the matrices M1 and M2.

```
   -
SUB,M1,M2
```

This operation will replace matrix M1 with matrix M1 less matrix M2.

MULT,M1,M2,M3 (M3 marked +)

This operation generates a new matrix M3 which is the product of matrices M1 and M2, or M3 = M1*M2.

TRAN,M1,M2 (M2 marked +)

This operation generates a new matrix M2 which is the transpose of matrix M1.

SCALE,M1,M2 (M1 marked -)

This operation replaces each term in the matrix named M1 with the term multiplied by the term M2(1,1) of the matrix named M2.

SOLVE,M1,M2,N1,N2 or SOLVE,M1,M2,,N2 or SOLVE,M1,N1,N2 or SOLVE,M1,M2,N1 or SOLVE,M1,M2 or SOLVE,M1

If N1 = 0, this operation solves the matrix equation AX=B. M1 is the name of the A matrix and M2 is the name of the B matrix. Matrix A is trangularized and the results X, are stored in M2.
If N1=1 Matrix A is triangularized only.
N1=2 For a given B matrix and the A matrix previously triangularized, the B matrix is replaced by the results X.
N1=3 Matrix A is replaced by its inverse FOR SYMMETRIX MATRICES ONLY.
N2=0 or blank, matrix A is symmetric. If N2 is nonzero the matrix A is not symmetric.
For symmetric matrices, matrix A is factored into the LDL form. The diagonal D matrix is stored on the diagonal of A. The parameter N2 permits the direct solution of non-symmetric systems of equations. If N2 is not equal to 0, an LU decomposition of matrix A will be performed. No direct replacement of M1 by its inverse is availabe for the non-symmetric case. Instead, use the ZERO operation to create an identity matrix M2 of the same order as M1. The command SOLVE,M1,M2,,N2 will then replace the matrix M2 with the inverse of the matrix A.

DUPSM,M1,M2,N1,N2,N3,N4 (M2 marked +)

This operation forms a new submatrix named M2 with N3 rows and N4 columns from terms within the matrix named M1. The first term of matrix M2, M2(1,1), will be from row N1 and column N2 of matrix M1, or M1(N1,N2).

$\bar{\text{STOSM}}$,M1,M2,N1,N2

STOSM,M1,M2,N1,N2

This operation stores a submatrix named M2 within the matrix named M1. The first term of the submatrix M2 will be stored at row N1 and column N2 of matrix M1. The terms within the area of M1 in which M2 is stored will be destroyed.

DUPDG,M1,M2

This operation forms a new row matrix named M2 from the diagonal terms of matrix M1.

STODG,M1,M2

This operation stores a row or column matrix named M2 at the diagonal locations of matrix M1.

MAX,M1,M2

This operation forms a column matrix named M2 in which each row contains the maximum absolute value of the corresponding row in matrix M1. The maximum and its column number is printed for each row.

NORM,M1,M2,N1

If N1 = 0, a row matrix named M2 is formed in which each column contains the sum of the absolute values of the corresponding column of matrix M1. If N1 /= 0, a row matrix named M2 is formed in which each column contains the square root of the sum of the squares of the values of the corresponding columns of matrix M1.

INVEL,M1

This operation replaces each term in the matrix named M1 with it's inverse.

SQREL,M1

This operation replaces each term in the matrix named M1 with the square root of the term.

LOG,M1

This operation replaces each term in the matrix named M1 with the natural log of the term.

PROD,M1,$\overset{+}{M}2$

This operation forms a 1 x 2 array named M2 which contains the product of all terms in the matrix M1. The product, X, is stored as two numbers of the form

X = P*10**E

in which M2(1) = P and M2(2) = E, the exponent.

DELETE,$\overset{-}{M}1$

This operation will cause the elimination from storage of the array named M1.

B. STATIC ANALYSIS OPERATIONS

The purpose of this series of operations is to form the total stiffness and diagonal (lumped) mass matrices for systems of two or three-dimensional elements. For three-edimensional analysis, there are beam and truss elements available. For two-dimensional analysis, there is a frame element, a slope/deflection element for beams, and a 3 to 8 node isoparametric finite element available.

After the creation of an array containing the coordinates of the joints of the system, the specification of displacement boundary conditions, the tabulation of material and section properties, the mass and stiffness matrices are formed for each structural member and placed in sequence on low speed storage along with the global equation numbers which are associated with their stiffness terms. In addition, the member force-displacement transformation matrices are formed and stored on a separate low speed storage file along with the appropriate displacement numbers.

The NODES operation is used to specify or generate the geometry of the system. The operation BOUND specifies which joint displacements exists and assigns internal equation numbers to these displacements. Therefore, each joint may have from zero to six displacement degrees of freedom. Tables of material and section properties for the various members are loaded and printed as standard arrays of information.

A special operation, ADDSF, is used for the direct addition of element stiffnesses to form the total stiffness and diagonal mass matrix of the system. The ADDK operation may be used to add individual elements into the total system

matrices. The LOADS operation specifies the concentrated joint loads for all load conditions. After the direct solution for joint displacements due to static or dynamic loads, the member forces can be evaluated useng the FORCE operation. Individual member forces can be evaluated using the MEMFRC operation. The DISPL operation is used to print the displacements in joint number order.

NODES,M1,N1

The cards following this operation provide information for the creation of a N1 by 3 array which will contain the coordinates for all the joints in the system. Where

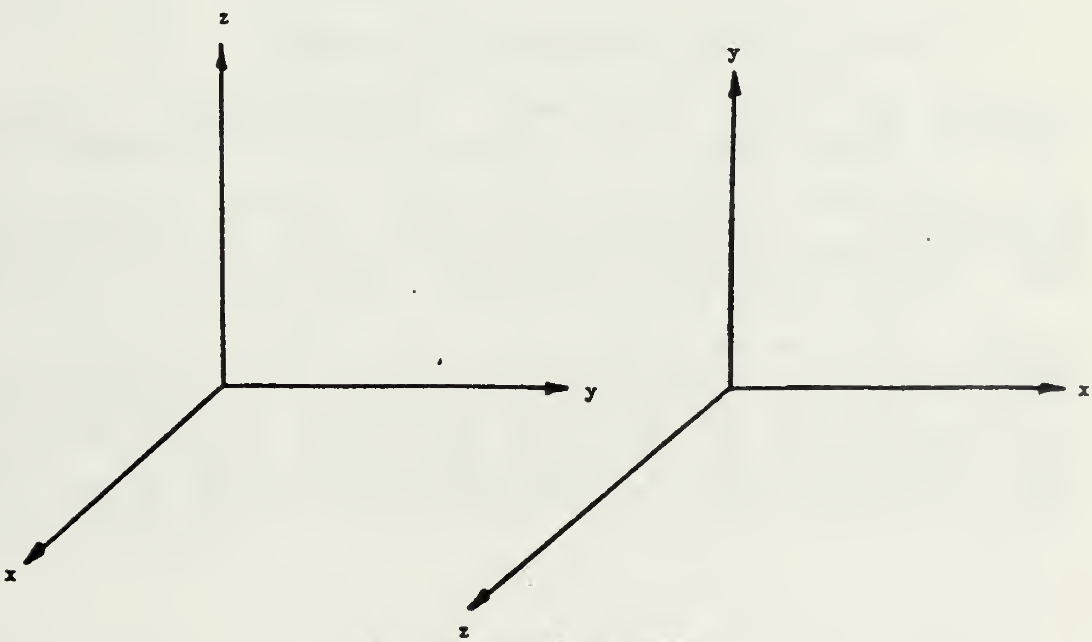
M1 = Name of new coordinate array to be loaded.

N1 = Number of joints (or nodes) in the system.

The following sequence of cards punched in a (2I5,3F10.0) format must follow this operation.

Columns	Contain
1 - 5	Node number selected by user
6 - 10	blank
11 - 20	X-coordinate
21 - 30	Y-coordinate
31 - 40	Z-coordinate

Nodes cards may be supplied in any order; however, all nodes must be defined. If nodes are defined more than once the last definintion will be used. This sequence of data must be terminated with a blank card.



POSITIVE COORDINATE SYSTEMS

BOUND,$\overset{+}{M}1$

This operation specifies the displacements which are nonzero for the structural system of joints specified by the NODES operation. Where

M1 = Name of boundary condition cade array to be generated.

This operation is followed by a series of cards containing the following information punched in a (8I5) format.

Columns	Contents
1 - 5	Node number for the first node in a series of nodes with identical displacement specification.
6 - 10	Node number for the last node in the series.
11 - 15	X-translation.
16 - 20	Y-translation
21 - 25	Z-translation
26 - 30	X-rotation
31 - 35	Y-rotation
36 - 40	Z-rotation
41 - 45	Node number increment used to generate Conditions for additional nodes.

A translition or rotation equals: (a) zero for zero or undifined displacements, or (b) one for nonzero displacements to be evaluated by other operations.

If a node boundary condition is not specified, all displacements at that node are assumed zero. Cards may be supplied in any order. If node boundary conditions are specified more than once, the last difinition is used. This sequence of data must be terminated by a blank card.

The selection by the user of which nodes have nonzero displacements requires an understanding of the direct stiffness procedure. Displacements degrees of freedom which have no stiffness associated with the displacement must be considered to be undifined since it is not possible to develop an equilibrium equation for that direction. The total number of nonzero displacements specified will be the size of the total stiffness matrix to be defined by the ADDSF operation.

BEAM,M1,M2,M3,M4

This operation calculates the element stiffness, mass and force-displacement transformation matrices for 3-D beam members. These arrays are stored in sequence on low speed storage to be used by other operations where:

M1 is the name of the beam element group
M2 is the name of the coordinate array
M3 is the name of the boundary condition array
M4 is the name of the arra which contain beam properties and has been loaded by the standard matrix LOAD operation

One card for each beam in this group of beam elements must follow this operation. The beam cards are punched in (5I5) format, where

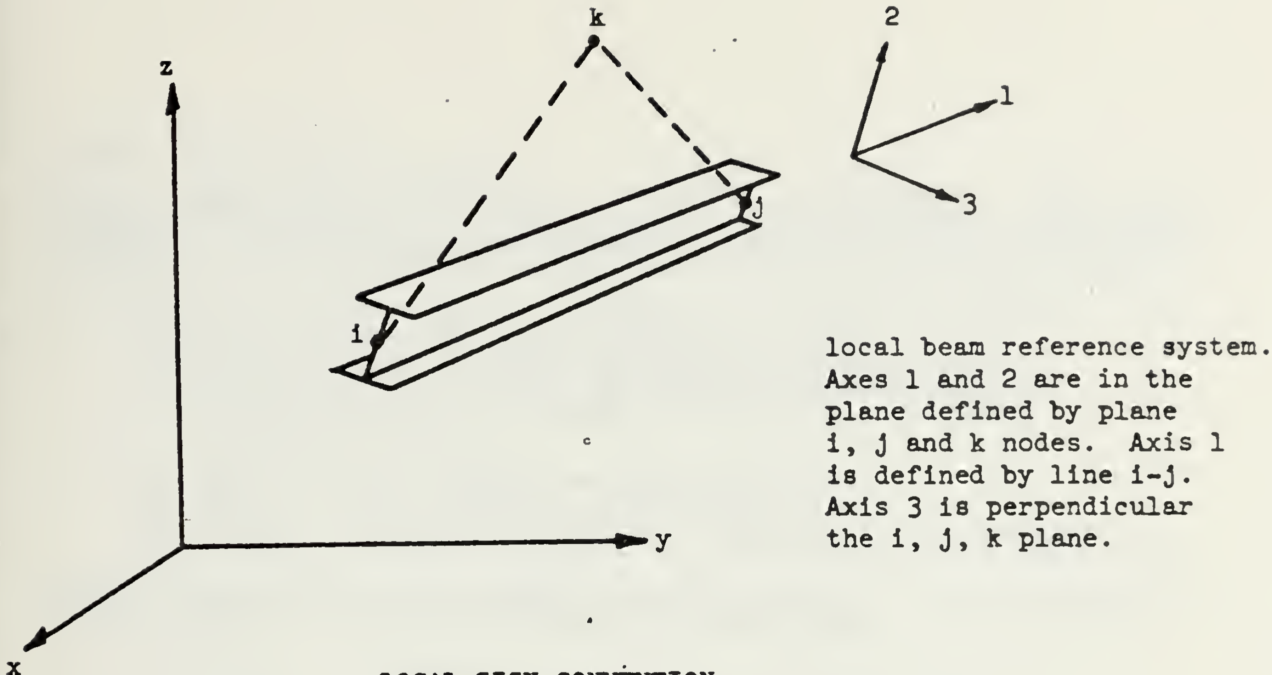
Columns	Contain
1 - 5	Beam identification number
6 - 10	Node number I
11 - 15	Node number J
16 - 20	Node number K
21 - 25	Beam property number NP

This sequence of cards must be terminated with a blank card.

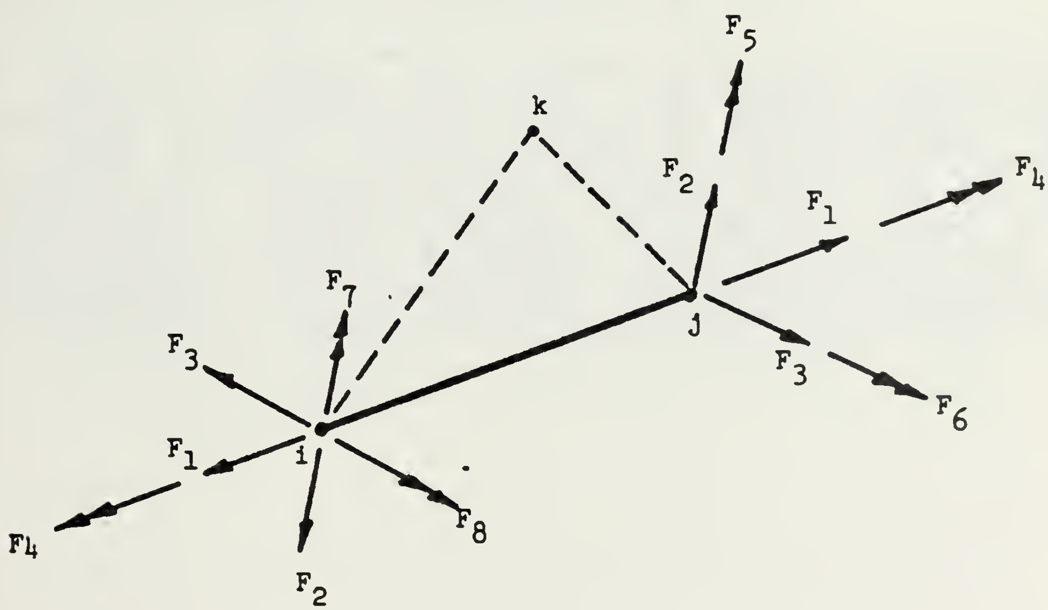
The material and geometric properties for each element are give in the M4 array in the following order:

M4 (NP,1) = Axial area of member, A
M4 (NP,2) = Torsional Moment of Inertia, J
M4 (NP,3) = Moment of Inertia about axis 2, I
M4 (NP,4) = Moment of Inertia about axis 3, I
M4 (NP,5) = Modulus of Elasticity, E
M4 (NP,6) = Shear Modulus, G
M4 (NP,7) = Mass per unit length of beam

where NP is the specific material property number specified in columns 21 - 25 of the beam card. The local sign convention is given in the following figure.



LOCAL SIGN CONVENTION



DEFINITION OF POSITIVE BEAM FORCES

+
TRUSS,M1,M2,M3,M4

This operation forms the element stiffness, mass and force-displacement transformation matrices for 3-D truss members. The arrays are stored on low speed storage in sequence and will be used by other structural operations.

M1 is the name of this groups of truss members
M2 is the name of the coordinate array
M3 is the name of the boundary condition array
M4 is an NP by 3 array of section properties in which NP is the number of different section properties and

M4 (NP,1) = the cross-section area, A
M4 (NP,2) = the Modulus of Elasticity, E
M4 (NP,3) = the mass per unit length of the member
This matrix can be loaded by the matrix load operation.

This operation is followed by one card per truss member in (4I5) format with the following information:

Columns	Content
1 - 5	Truss member identification number
6 - 10	Joint number I
11 - 15	Joint number J
16 - 20	Section property number, NP

This operation must be terminated by a blank card.

+ +
LOADI,M1,N1,N2,N3,N4 or LOADI,M1,N1,N2,,N4 or
LOADI,M1,N1,N2,N3 or LOADI,M1,N1,N2

This operation will load an integer array named M1 which has N1 rows and N2 columns. The terms of the array are punched in row-wise sequence on data cards which follow this operation. If N3 is zero or blank, the data must be punched in (16I5) format. If N3 is nonzero, an additional card containing the format of the data cards must follow this operation and preceed the data cards. N4 is optional but note that an extra comma must be used in place of N3 if the user does not supply a format. The matrix M1 will be printed in partitioned form with N4 columns per partition. Lines have (N4 + 1)*5 characters. N4 devaults to 20 causing 125 characters to print per line. The user is cautioned not to overcome the capacity of the printing device in use.

PLANE,M1,M2,M3,M4,N1,N2

This operation calculates the element stiffness, mass and stress-displacement transformation matrices for 3 to 8 node isoparametric elements. (Y-Z plane only). These arrays are stored in sequence as a group on low speed storage to be used later by other operations (i.e., ADDSF and FORCE). The arguments are defined as

M1 is the user defined name of the element group
M2 is the name of the joint coordinate array
M3 is the name of the boundary condition array
M4 is the name of the array which contains the material properties of the elements (one row per different material) where

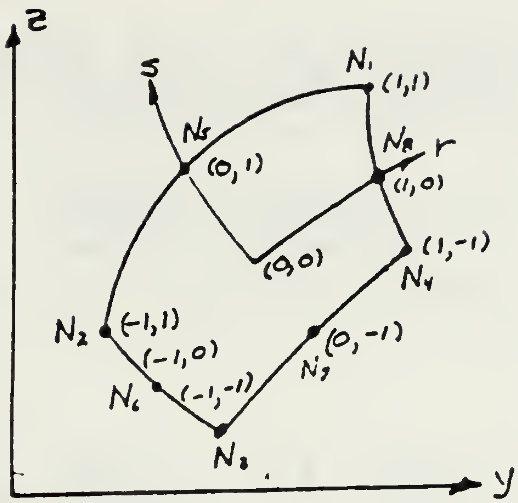
M4 (NP,1) = Modulus of Elasticity, E
M4 (NP,2) = Poissons Ratio, v
M4 (NP,3) = Thickness of the element
M4 (NP,4) = Mass density of the element
NP is the material identification number

N1 and N2 are the number of integration points in the r and s directions respectively.

One card for each 3 to 8 node element in the group must follow the operation card. The card is punched in a (10I5,6F5.0) format and contains the following information:

Columns	Contain
1 - 5	element identification number
6 - 10	Node number N1
11 - 15	Node number N2
16 - 20	Node number N3
21 - 25	Node number N4
26 - 30	Node number N5
31 - 35	Node number N6
36 - 40	Node number N7
41 - 45	Node number N8
46 - 50	Material identification number, NP
51 - 55	Natural coordinate of stress output r1
56 - 60	Natural coordinate of stress output s1
61 - 65	Natural coordinate of stress output r2
66 - 70	Natural coordinate of stress output s2
71 - 75	Natural coordinate of stress output r3
76 - 80	Natural coordinate of stress output s3

N4 through N8 are optional. The midside nodes, if present, must be within center half of side. The local numbering system for the element is shown in the following figure.



ISOPARAMETRIC ELEMENT

Stresses will be printed by the FORCE operation at the three points defined in columns 51 through 80. The forces are defined as follows:

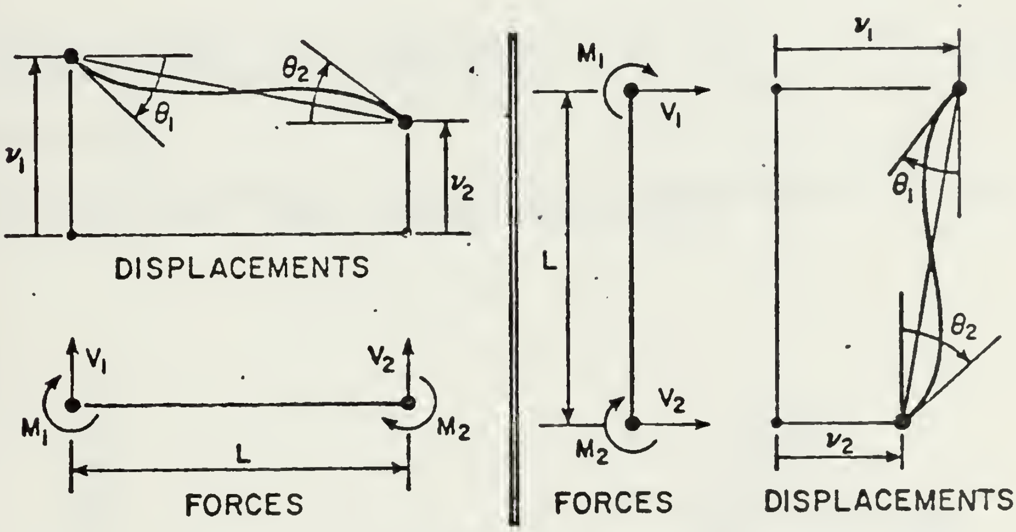
$$\begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \\ F_7 \\ F_8 \\ F_9 \end{bmatrix} = \begin{bmatrix} \tau_{xx}^{(1)} \\ \tau_{yy}^{(1)} \\ \tau_{xy}^{(1)} \\ \tau_{xx}^{(2)} \\ \tau_{yy}^{(2)} \\ \tau_{xy}^{(2)} \\ \tau_{xx}^{(3)} \\ \tau_{yy}^{(3)} \\ \tau_{xy}^{(3)} \end{bmatrix}$$

SLOPE,$\overset{+}{M}1$

This operation forms a 4 x 4 stiffness matrix, M1 for a beam or column member from the classical slope-deflection equations. The properties of the member are defined on one card immediately following the operation. This second card is punched in a (3F10.0) format and contains the following information:

Columns	Contain
1 - 10	Moment of Inertia, I
11 - 20	Modulus of Elasticity, E
21 - 30	Length of Member, L

The sign convention is defined as follows:



The member forces are defined in terms of joint displacements by the following slope deflection equations.

$$M_1 = \frac{EI}{L}\left[4\theta_1 + 2\theta_2 - \frac{6}{L}\left(v_1 - v_2\right)\right]$$

$$M_2 = \frac{EI}{L}\left[2\theta_1 + 4\theta_2 - \frac{6}{L}\left(v_1 - v_2\right)\right]$$

$$V_1 = -V_2 = \frac{M_1 + M_2}{L}$$

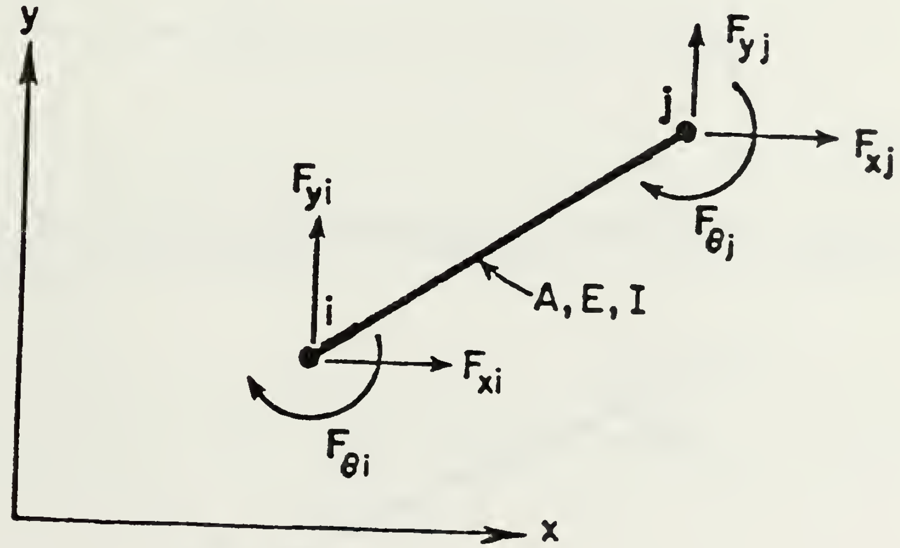
or in matrix form

$$\begin{bmatrix} M_1 \\ M_2 \\ V_1 \\ V_2 \end{bmatrix} = \frac{EI}{L} \begin{vmatrix} 4 & 2 & -\frac{6}{L} & \frac{6}{L} \\ 2 & 4 & -\frac{6}{L} & \frac{6}{L} \\ -\frac{6}{L} & -\frac{6}{L} & \frac{12}{L^2} & -\frac{12}{L^2} \\ \frac{6}{L} & \frac{6}{L} & -\frac{12}{L^2} & \frac{12}{L^2} \end{vmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \nu_1 \\ \nu_2 \end{bmatrix}$$

or symbolically F = KΔ Where K is the 4 x 4 stiffness matrix defined by the name M1.

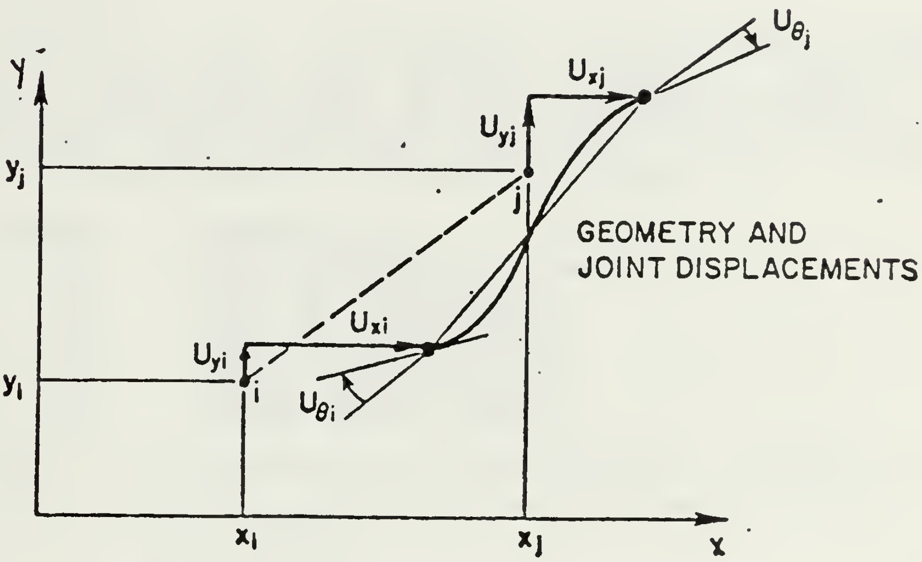
FRAME,$\overset{+}{M}$1,$\overset{+}{M}$2

This operation forms the 6 x 6 stiffness matrix M1 for the two-dimensional frame member shown below.

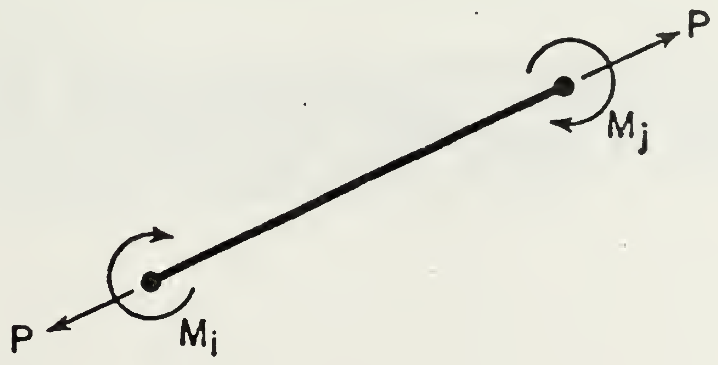


The properties of the member are defined on one card immediately following the FRAME operation card. This second card is punched in a (8F10.0) format and contains the following information:

Columns	Contain
1 - 10	Axial area, A
11 - 20	Modulus of Elasticity, E
21 - 30	Moment of Inertia, I
31 - 40	Xi
41 - 50	Yi
51 - 60	Xj
61 - 70	Yj



M2 is a 3 x 6 force-displacement transformation matrix which is based on the positive difinition of the element forces shown below.



These forces can be calculated from the following matrix equation, with the MEMFRC operation.

$$\begin{bmatrix} M_i \\ M_j \\ P \end{bmatrix} = \underline{M2} \begin{bmatrix} U_{xi} \\ U_{yi} \\ U_{\theta i} \\ U_{xj} \\ U_{yj} \\ U_{\theta j} \end{bmatrix}$$

LOADS,M1,M2,N1

This operation forms a load matrix named M1 of N1 columns (N1 load conditions) where M2 is the name of the boundary condition array generated by the operation BOUND. This operation is followed by a series of cards - one for each loaded joint for each load condition. These cards are punched in (2I5,6F10.0) format as follows:

Columns	Contain
1 - 5	Joint number
6 - 10	Load condition number
11 - 20	Load in X-direction
21 - 30	Load in Y-direction
31 - 40	Load in Z-direction
41 - 50	Moment about X-axis
51 - 60	Moment about Y-axis
61 - 70	Moment about Z-axis

This series of cards must be terminated by a blank card.

ADDSF,M1 or ADDSF,M1,M2

This operation forms the total stiffness matrix named M1 and a lumped mass matrix named M2 for the structural system from the element stiffness and mass matrices which are stored on low speed storage. These matrices can be printed with the PRINT operation. If M2 is not specified, the row mass matrix M2 will not be formed.

ADDK,M1,M2,M3,N1

This operation adds the element stiffness matrix named M2 to the total stiffness matrix named M1, where M1 was previously defined and initially set to zero. M3 is the name of the integer array in which the column number N1 contains the row or column numbers in the total stiffness matrix where the element stiffness terms are to be added.

MEMFRC,M1,M2,M3,M4,N1

This operation multiplies the element stiffness matrix named M1 by the joint displacement matrix named M2. M3 is the name of the integer array in which the column number N1 contains the row numbers in the displacement matrix, M2, which are to be multiplied by the element stiffness (or force-displacement) matrix, M1. The results of this multplication are stored in the array named M4.

DISPL,M1,M2

This operation prints the displacement named M1 in joint sequence order, where M2 is the name of the boundary condition array.

FORCE,M1,M2,M3 or FORCE,M1;M2

This operation calculates the member forces for a group of elements in whcih

M1 is the name of the element group

M2 is the displacement matrix

M3 is the name of the matrix in which the forces are stored in the order calculated. If this array is not specified, the element forces will be printed only and will not be retained in storage. For the TRUSS element, only the member axial force, F, will be calculated for each member. For the BEAM element, eight forces will be printed with reference to the positive definiton shown in the BEAM operation.

C. DYNAMIC ANALYSIS OPERATIONS

The following operations were designed to evaluate the dynamic response of structures subjected to arbitrary time-dependent loads. If these operations are used in connection with the standard matrix operations and the sturctural analysis operation, a dynamic analysis is a relatively simple procedure. The user has the option of using the mode superposition method or a direct step-by-step integration of the dynamic equations of motion. The user may examine the spectra of both input loading and calculated displacements. In addition, the contributions of the individual modes may be evaluated and compared.

The most common and convenient form for time-dependent data to be specified is as straight line segments between given time points. Therefore, an operation which generates values at equal intervals is necessary. Another common characteristic of time-varying loads on structures is that it is normally possible to represent the loads at all points on the structure by the product of two matrices, a column matrix indicating the spacial distribution of loads times a row matrix which indicates the values as a function of various times. If a more complicated loading is required, it is possible to perform more analyses, each within the restrictions of the program, and then add the results of each analysis.

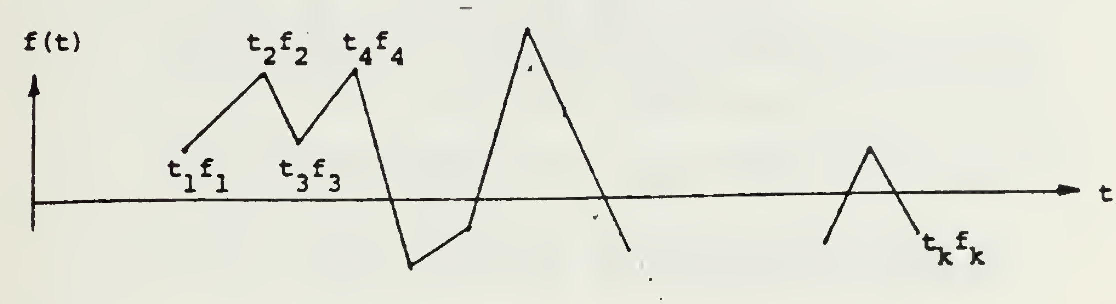
The following operations have been added for the major purpose of performing dynamic analysis.

FUNG,M1,$\overset{+}{M}$2,M3,N1,N2

This operation generates a matrix named M2 which contains values at equal intervals of the function specified in the array named M1. The array M1 must be a 2 by k array of the form:

$$M1 = \begin{bmatrix} t_1 & t_2 & t_3 & \cdots\cdots\cdots t_k \\ f_1 & f_2 & f_3 & \cdots\cdots\cdots f_k \end{bmatrix}$$

which numerically represents a function of the form shown below:



The time interval t is specified in the 1 x 1 matrix named M3. N1 specifies the total number of values to be generated, and is the number of columns in M2. If N2 = 0, the array M2 will be a 1 x N1 row matrix in which the first value will be f . If N2 0, the array M2 will be a 2 x N1 matrix of the following form:

$$M2 = \begin{bmatrix} t_1 & t_1 + \Delta t & t + 2\Delta t \cdots\cdots\cdots\cdots \\ f_1 & f(t_1+\Delta t) & f(t_1 + 2\Delta t)\cdots\cdots\cdots \end{bmatrix}$$

STEP,M1,M2,M3,M4,M5,M6,M7,M8,N1,N2

This operation calculates the dynamic response of a sturctural system usnig direct step-by-step integration of the following linear matrix equation of motion:

$$\underline{M\ddot{U}} + \underline{C\dot{U}} + \underline{KU} = \underline{R}(t) = \underline{PF}(t)$$

Where
M1 is the name of the N x N stiffness matrix K
M2 is the name of the N x N mass matrix M
M3 is the name of the N x N damping matrix C
M4 is the name of the N x 3 initial condition matrix U in which:

U (I,1) is a vector of displacements U
U (I,2) is a vector of velocities $\dot{U}$
U (I,3) is a vector of accelerations $\ddot{U}$

M5 is the name of the N x N2 matrix of calculated displacements in which column "i" represents the displacements at time i*N1*Δt
M6 is the name of the N x 1 load distribution matrix P
M7 is the name of the 1 x k row matrix representing the load multipliers at equal time increments F, where k = N2/N1
M8 is the name of the 1 x 1 matrix containing Δt
N1 is the output interval for the displacements
N2 is the total number of displacement vectors to be calculated.

The total time for which results will be calculated by this operation is N1*N2*Δt. This operation must be followed with one data card in (3F10.0) format containing the following information:

Columns	Contain
1 - 10	DELTA
11 - 20	ALPHA
21 - 30	THETA

Different values of delta, alpha and theta will allow the user to select different methods of step-by-step integration. The following table lists some possibliities:

	DELTA	ALPHA	THETA
Newmarks Average Acceleration	1/2	1/4	1.0
Linear Acceleration	1/2	1/6	1.0
Wilson's Theta Metnod (low damping)	1/2	1/6	1.42
Wilson's Theta Method (high damping)	1/2	1/6	2.0

EIGEN,$\bar{M}1$,$\overset{+}{M}2$,$\bar{M}3$,N1

This operation solves the following eigenvalue problem:

$$K\phi = M\phi\lambda$$

In which the N x N, symmetric, positive-semidefinite matrix K is named M1. The matrix M is a diagonal matrix of nonzero, positive terms designated by M3. The matrix M3 must be a row or column matrix containing only the diagonal terms of M. The eigenvalues, λ, are stored in matrix M3. The eigenvalues are ordered in numerically increasing order and the eigenvectors, ϕ, are stored in the corresponding columns of the matrix M2. The number N1 specifies the approximate number of significant figures of the eigenvalues. If N1 is zero or blank, 4 figure accuracy will be used. The maximum accuracy possible is 16 figures. The use of more than 12 figure accuracy is not recommended.

The program reduces the problem to standard eigenvalue form by the following transformation

where

$$K^* = m^t K m$$

$$I = m^t M m$$

in which

$$m_i = 1/\sqrt{M_{ii}}$$

The calculated mode shapes, ϕ, are normalized as follows:

$$\phi^t M\phi = I \qquad \phi^t K \phi \quad \lambda$$

The program uses the standard Jacobi diagonalization method to solve for all eigen values and eigenvectors.

DYNAM,M1,M2,M3,M4,M5,M6,N1

This operation evaluates the following set of uncouples second order differential equations associated with the mode superposition method for the dynamic analysis of a structural system.

$$\ddot{x}_i + 2\lambda_i \omega_i \dot{x}_i + \omega_i^2 x_i = P_i(t) \qquad i = 1 \text{ to } N \text{ nodes}$$

M1 is the name of a row or column matrix which contains the N terms (frequencies in rad/sec). M2 is the name of a row or column matrix which contain the N λ_i terms (ratio of modal damping to critical damping).

The generalized time-varying forces P_i (t) are not specified directly but are evaluated from more fundamental information. The forces for all modes are evaluated at specific times by the program from the following matrix equation:

$$P = p*f = M3*M4$$

In which p is a specified N x 1 vector named M3, and f is a 1 x N1 row matrix which will be generated from the 2 by k array named M4. The array M4 is the same form as the input array described under the operation FUNG. It is not necessary to use FUNG before the DYNAM operation.

M5 is the name of the N x N1 array which contain the generalized displacement X_i (t).

M6 is the name of the 1 x 1 array which contains the time increment associated with the generalized displacements.

N1 is the number of displacements to be generated.

The method of integration used is exact for straight line secments.

PLOT,M1,N1

This operation will prepare a printer plot of selective rows of the matrix named M1. N1 is the number of rows of M1 which will be plotted by this operation. This operation is followed by N1 cards in (1A1,I4) format with the following information:

Columns	Contain
1	Plot symbol - any keypunch symbol
2 - 5	Row number to be ploted

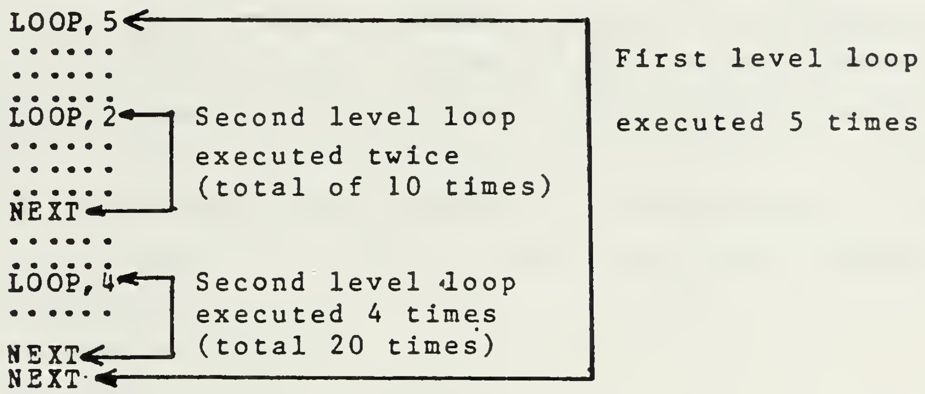
The program automatically searches the information to be plotted for the maximum and minimum values. The difference in these numbers divided by 120 spaces is selected as the plot scale.

D. LOOPING OPERATIONS

CAL has a five level looping ability. The first operation is LOOP and the last operation is NEXT. Operations withing CAL are normally executed as they are encountered. If the operation requires data, the data cards follow the operation card. In the case of looping, however, all operation cards from the first LOOP card to the last NEXT card are stored within the computer before they are executed; therefore, if operations within the loops requre data, the data cards must be supplied in the order required after the last NEXT operation. If an error is encountered while executing in a loop, the entire matrix of loop commands is deleted and the user is given the opportunity to try again. Matrices that have been modified by operations successfully completed while in the loop remain modified. After all loops are executed the computer storage required for these operations is automatically released by the program. The looping operations are:

LOOP,N1

N1 is the number of times the loop is to be executed. Associated with each LOOP operation there must be a corresponding NEXT operation which signifies the end of the loop and the return of the control to the beginning of the loop. The following is a possible series of looping operations.



data for all operations within all loops

NEXT,M1 or NEXT

The operation NEXT signifies the end of a loop. It is apparent which LOOP and NEXT cards are associated if there are an equal number of each. The operation NEXT,M1 will cause the loop to terminate if the first term in the matrix named M1 is negative.

SKIP,M1,N1

This operation will cause the skip of the next N1 operations if the first term in the matrix named M1 is negative. This level of looping.

E. USER DEFINED OPERATIONS

USERA and USERB

These names are reserved for operations to be defined and programmed by the user. In order to program these operations is is necessary to understand the internal organization of CAL. Chapter III contains details.

V. LARGE PROBLEMS

CAL is designed as an educational tool. It does not take advantage of banding and symmetry in matrix storage. Larger problems can be solved by increasing the dimension of the L array but a general purpose program that makes maximum advantage of out-of-core storage and takes advantage of banding and symmetry for in-core matrix storage is probably a better choice. With the above disclaimer, to increase problem size capability, increase the dimension of the L array and change the value of MAX to the new dimension size in the follosing:

```
C-----MAIN PROGRAM
C-----SET PROGRAM CAPACITY
          COMMON NTOT,NDP,L(6000)
          MTOT=5000
          NDP=2
          CALL SETIME
          CALL CAL1
          STOP
          END
```

SETIME is an OS routine to initialize the CPU timer and should not be used for CP/CMS programs. With the dimension of the L array as above, the program currently executes in 144k bytes for OS and 256k bytes for CP/CMS. The region necessary for execution will increase about eight times the increase in the L array.

LIST OF REFERENCES

1. Zienkiewicz, O. C., The Finite Element Method, p. 733-756, McGraw-Hill Book Company (UK) limited, 1977.
2. Little, R. D., An Interactive Matrix Interpretive System for the IBM 360/67, M.S. Thesis, Naval Postgraduate School, Monterey, CA, 1970.
3. University of California Report No. UC SESM 77-2, CAL Computer Analysis Language for the Static and Dynamic Analysis of Structural Systems, by E. L. Wilson, January 1977.
4. Day, A.C., Fortran Techniques, Cambridge University Press, 1972.
5. Kreitzberg, C.B. and Shneiderman, B., The Elements of Fortran Style: Techniques for Effective Programing, Harcourt Blace Jovanovich, Inc., 1972.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
3.	Department Chairman, Code 69 Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93940	1
4.	Professor Gilles Cantin, Code 69ci Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93940	10
5.	Professor R. E. Newton, Code 69ne Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93940	1
6.	Professor R. Franke, Code 53fe Department of Mathematics Naval Postgraduate School Monterey, California 93940	1
7.	Professor C. Wilde, Code 53wn Department of Mathematics Naval Postgraduate School Monterey, California 93940	1
8.	Professor O. C. Zienkiewicz, Code 69zw NAVSEA Research Chair, Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93940	1
9.	Professor E. L. Wilson University of California College of Engineering Department of Civil Engineering Division of Structural Engineering and Structural Mechanics Berkeley, California 94720	1

10. Assoc Professor G. Aguirre-Ramirez, Code 53rm 1
Department of Mathematics
Naval Postgraduate School
Monterey, California 93940

11. Assoc Professor A. L. Schoenstadt, Code 53zh 1
Department of Mathematics
Naval Postgraduate School
Monterey, California 93940

12. Mr. Ken Wong 1
University of California
College of Engineering
Department of Civil Engineering
Division of Structural Engineering and Structural Mechanics
Berkeley, California 94720

13. LCDR L.B. Elliott, USN 1
Long Beach Naval Shipyard
Code 338.14
Long Beach, California 90822

Thesis
E3686 Elliott 186763
c.1
Non-numerical appli-
cations of computer
programming in the con-
struction of problem
oriented languages.

10 NOV 80 27175
AUG 81 27127
5 JUN 86 13823

Thesis 186763
E3686 Elliott
c.1
Non-nemerical appli-
cations of computer
programming in the con-
struction of problem
oriented languages.

thesE3686
Non-numerical applications of computer p
3 2768 001 89301 9
DUDLEY KNOX LIBRARY