EVALUATION OF THE FRONTAL SOLVER ON THE IBM PC

by

Ahmad I. Rayyan

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Civil Engineering

APPROVED :

S. M. Holzer, Chairman

R. M. Barker

T. Kuppuswamy

January 1986
Blacksburg, Virginia

ABSTRACT

EVALUATION OF THE FRONTAL SOLVER

ON THE IBM PC

by

Ahmad I. Rayyan

Committee Chairman : S. M. Holzer

Civil Engineering

ABSTRACT

In this thesis, frontal subroutines are implemented to a plane frame analysis program for execution on the IBM PC. The resulting program solves for the unknown joint displacements of frame structures with large numbers of degrees of freedom by utilizing a peripheral back-up storage; which can not be analyzed directly in core. A comparison of the frontal solver and the out-of-core band solver is presented.

ACKNOWLEDGEMENTS

The author wishes to thank Dr. S. M. Holzer for serving as the author's major advisor and for his guidance and help throughout this thesis. He also wishes to thank Dr. R. M. Barker and Dr. T. Kuppuswamy for serving on the author's committee.

The author is also grateful to his wife, Susan, for her love and patience. The author would like to express appreciation to Mr. J. M. Elzebda and Mr. M. A. Saed for their friendship and support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

OBJECTIVES

A purpose of this thesis is the implementation of a frontal solver in FORTRAN 77 for execution on the IBM PC (IBM Personal Computer) into an already written program. A plane frame analysis program written by Holzer (11) was selected for this purpose. The resulting program will analyze frame structures with large number of degrees of freedom, which can not be analyzed directly in the central memory of a personal computer using an in-core equation solver. The program will be tested to determine the upper bounds of the personal computers in analyzing practical frame structures.

Another purpose is to conduct a literature survey and present the different points of view in regard to the evaluation of the frontal method, and in regard to comparing it with the band method.

The last purpose is to conduct a comparison between the frontal solver and the out-of-core band solver. The comparison will be conducted using the program developed in this thesis and another program that uses the out-of-core band solver, which was also implemented for execution on the IBM PC.

The basis for this comparison will be the out-of-core storage requirements and the execution time. The

comparison will be conducted twice, first using an IBM PC with double floppy disk drives, and second using an IBM PC with a ten mega byte hard disk drive.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

The frontal solver technique publicized by B. M. Irons (3) in 1970 is a Gauss elimination based technique; it assembles and solves symmetric positive definite equations generated in finite element applications. It is more involved than the band solver algorithm, but more efficient in the case of two-dimensional and three-dimensional elements with more nodes than corner nodes (3). The frontal technique is thought to be the most natural method of solution (1) in which elements, simple elements or superelements, are numbered and assembled in a natural way. The assembly of elements alternates with the elimination of variables (degrees of freedom) associated with each element at a time. Variables are eliminated as soon as the corresponding stiffness coefficients are fully summed, and then the variables and their coefficients are transferred to an out-of-core storage. Therefore, the system stiffness matrix as a whole is never assembled and contained in core. The assembly process , element by element, divides the total number of variables into three dynamic sets of variables: (1) variables that are already processed and discarded (stored in an out-of-core file), (2) variables that are being processed and are still in

core, and (3) variables that are yet to be processed. The set of variables that are in core are called active variables. Stiffness coefficients corresponding to the active variables are also contained in core and are called the frontal matrix (the front). The frontwidth, which is the number of the active variables, is continually changing as the front advances over the structure.

As the front advances over the last element and the elimination of all variables is completed, variables are considered for back substitution. To solve for the variables, stiffness coefficients along with instructions on how to solve for the variables are read from the out-of-core file. Solved for variables are stored in two vectors; the first is a running vector whose length is equal to the maximum frontwidth, and it is only used during subsequent computations in the backsubstitution process, and the second vector which spans the entire number of the structure's degrees of freedom is used for post processing.

2.2 Limitations of the frontal method

The method can fail to solve large problems due to (among other reasons) insufficient in-core and out-of-core storage (8). It is difficult to predict the frontwidth for complex structures. However, element numbering algorithms are available (7) which seek an optimum element numbering and compute the frontwidth in advance. The frontwidth, which dictate the amount of storage needed in core and out-

of-core may be used to compute the size of the largest out-of-core file needed for the solution of a large problem. The formula is:

SIZE (bytes) = 8 * NTOTV * (MFRON + 4)

in which

NTOTV = the total number of variables

MFRON = the frontwidth,

and the number 8 is used for double precision.

2.3 Flexibilities of the frontal method

S. F. Abbas (5) described the flexibilities inherent in the frontal concept of Irons. According to Abbas, the solver uses the following interrelated concepts: the element-oriented processing, the alternating of assembly and elimination, Gauss elimination (by parts, where Gauss elimination is performed before the assembly is completed), the advancing front, and most important of all is the separation of the planning and the execution (assembly, Gauss elimination, and back substitution) phases. The planning process presented in the pre-front subroutine determines the order and time in which variables enter and leave the active part of the system (the front). Because of the separation of the planning and execution phases, the frontal solver possess flexibility in the order of elimination of variables, and in the allotment of destinations of system variables in the front. This flexibility can be utilized to further improve the

efficiency of existing algorithm, or devise new applications of the algorithm. This flexibility can be evident when using controls available at the element level or the subelement level. Some examples are given.

2.3.1 The element level

In the case of iterative solutions, a great deal of effort and execution time can be saved by numbering elements last in the regions of the structure where changes are expected. Therefore, solutions subsequent to alterations do not have to be repeated for the entire structure .Where superelements are preferred over simple elements, the pre-front is applied twice; first to generate the superelements, and second to modify the element incidence matrix to reflect the joints last appearances in superelements which are treated as a simple element.

2.3.2 Longevity

Another degree of flexibility may be gained by manipulating longevity, i.e., the life span of a variable in the front. Longevity may be reduced (by using element numbering optimizer) to optimize both efficiency and core storage requirement, or it may be arbitrarily increased to advantage, for example, in partial condensation. Guyan-reduced superelement matrices may be generated by assigning long longevities to corresponding variables.

2.3.3 Destination allotment and efficiency

Efficiency can be improved by utilizing an optimum

allotment of destinations of the system variables in the front. The presence of empty destinations within the front is undesirable, even though the solver does not operate on zero elements. These zero elements can be filled when using dynamic allotment of destinations in which variables from the tail-end of the front are transferred to empty destinations within the front. This method requires that corresponding stiffness coefficients be transferred as well. However, the simultaneous use of dynamic allotment and the frontal end-elimination lead to a compact front all the time.

2.3.4 Variables within an element

At the element level, elimination can be selected at will. If the variable at the tail end of the front is making a final appearance in a particular element, then by the elimination of that variable an early contraction of the front is achieved, making the algorithm more efficient (by operating on a smaller front).

2.3.5 The front

The process of assembling stiffness coefficients is totally independent of Gauss elimination. Recognition of this fact has opened the door for a number of applications in the general finite element algorithms. The frontal matrix multiplication and frontal matrix storage are two examples both of which require no modifications of the pre-front.

2.4 Frontal matrix related activities

The following applications may be applied to the frontal matrix to enhance efficiency and overcome core storage limitation.

2.4.1 Partitioning of the frontal matrix

It was also suggested in (4) that if the frontal matrix is too large to be contained in core during decomposition, processing by blocks, analogous to the band solver, may be used to overcome the core limitation. It should be mentioned that in such a case maximum efficiency will be compromised.

2.4.2 Unsymmetric coefficient matrices

P. Hood (5) has presented a frontal program that may be used for the solution of unsymmetric matrix equations generated in certain applications of the finite element method to boundary value problems. In his study the frontal solver was chosen over the band solver for two reasons: (1) the frontal is faster and requires less core storage, (2) no restrictions on the node numbering scheme is required.

The program first assembles element stiffness matrices into the frontal matrix. Once the memory storage allocated to the frontal matrix is filled, a search is conducted for a pivotal element in the fully summed columns and rows to which no more contributions will be added when more elements are assembled. The pivotal row is used to eliminate all coefficients in the pivotal column. Once the

elimination is completed, the pivotal row is transferred to a backup disc. When sufficient coefficients have been eliminated, more elements are assembled and so on. The solution is obtained by a back substitution routine after all the variables are eliminated.

2.4.3 Skyline storage

In a study done by Thompson and Shimazak (6), the frontal skyline method was presented. The method, while using the standard frontal procedures, allows for compact skyline storage of the frontal matrix in core. It is advantageous to use this method when the frontal matrix size varies greatly from one position in the mesh to another. When the frontal matrix size is small, the frontal skyline requires the same amount of core storage as the standard frontal method. However, the efficiency in core storage requirement tends to grow, to the advantage of the frontal skyline, as the frontal matrix size increases.

CHAPTER 3

THE FRONTAL METHOD

3.1 Introduction

The solution of a set of simultaneous algebraic equations is a key task of any finite element program. Consequently, the efficiency of programs is greatly influenced by the method adopted for equations solution. Programmers have several options ranging from iterative methods, such as Gauss–Seidel, to direct methods such as Gauss elimination. In this chapter a direct method, namely the frontal method, will be described.

The frontal method was first developed around 1958 by engineers in the Boeing Company in Seattle; Bob Melosh and Fred Magness are thought to be two of the pioneers of this method (1).

The frontal method is a particular form of Gauss elimination where the assembly process alternates with the elimination process for each element at a time. Therefore, the whole stiffness matrix is never contained in core.

It is designed to minimize the central memory requirements and the number of arithmetic operations by utilizing out–of–core temporary storage (2).

3.2 General description

The frontal method presented in this chapter is applicable only to solutions of symmetric systems of linear

equations. It alternates between the assembly of equations and the elimination of variables. A variable is eliminated as soon as the stiffness coefficients of its equation are fully summed, that is, after the last element containing the variable is assembled.

The core contains only the upper triangular half of the global stiffness submatrix which contains the stiffness coefficients of variables that are being assembled at a particular time. The stiffness coefficients contained in core are called the frontal matrix (the front). The number of variables (degrees of freedom) whose stiffness coefficients are stored in core is called frontwidth, which is continually changing as each element is processed, and which is optimized by choosing a good element numbering system. The maximum frontwidth controls the amount of core storage used. Variables whose stiffness coefficients make up the front are termed active; those which have been reduced out of the front are called deactive; and the rest of the variables are called inactive (2).

During the assembly process, elements are assembled in the order in which they are numbered. Once an element is introduced for assembly, the three housekeeping vectors are computed, and its stiffness coefficients are either summed to coefficients already existing in core (that is, for already active variables) or assembled into new positions for variables being activated for the first time, the

element applied load vector is also assembled into the global load vector in the same fashion as the stiffness coefficients. As for the elimination process, variables with fully assembled stiffness coefficients are eliminated by Gauss elimination and their coefficients are transferred to an out-of-core storage. Therefore, more space in core is made available for new elements yet to be assembled. For example, consider the frame problem shown in figure 3.1.

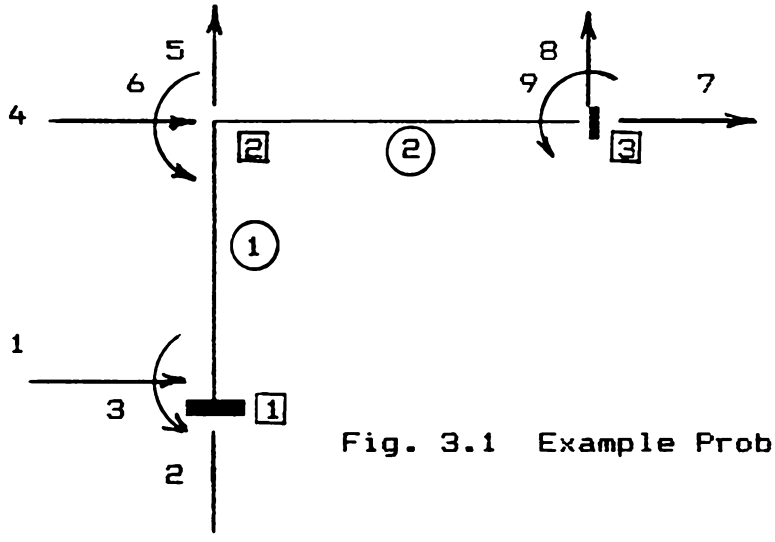


Fig. 3.1 Example Problem

Before starting the assembly process, it should be noted that all variables whether prescribed or not are treated as unknowns during this process.

Assemble element 1: Write the equations in variables 1 – 6

$$\begin{bmatrix} K^I_{11} & K^I_{12} & K^I_{13} & K^I_{14} & K^I_{15} & K^I_{16} \\ & K^I_{22} & K^I_{23} & K^I_{24} & K^I_{25} & K^I_{26} \\ & & K^I_{33} & K^I_{34} & K^I_{35} & K^I_{36} \\ & & & K^I_{44} & K^I_{45} & K^I_{46} \\ & & & & K^I_{55} & K^I_{56} \\ & & & & & K^I_{66} \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \end{bmatrix} = \begin{bmatrix} F^I_1 \\ F^I_2 \\ F^I_3 \\ F^I_4 \\ F^I_5 \\ F^I_6 \end{bmatrix} \qquad (3.1)$$

Stiffness coefficients of variables associated with node 1 are fully summed and can be eliminated via Gauss elimination. The eliminated stiffness coefficients, the right hand side, and the position within the front are transferred to an out-of-core file for later use in the back-substitution process. Variables of node 2, whose stiffness coefficients have not yet been fully summed pending the assembly of element 2, remain active in the front.

$$\begin{bmatrix} K^{I\prime}_{44} & K^{I\prime}_{45} & K^{I\prime}_{46} \\ & K^{I\prime}_{55} & K^{I\prime}_{56} \\ & & K^{I\prime}_{66} \end{bmatrix} \begin{bmatrix} q_4 \\ q_5 \\ q_6 \end{bmatrix} = \begin{bmatrix} F^{I\prime}_4 \\ F^{I\prime}_5 \\ F^{I\prime}_6 \end{bmatrix} \qquad (3.1a)$$

Equation 3.1a shows what is contained in the global stiffness submatrix (the front) after the elimination of variables 1, 2, and 3 the primes indicate modifications due to the elimination of variables of node 1.

Assemble element 2 : write equations in variables 4 – 9

$$\begin{bmatrix} K^{II}_{12} & K^{II}_{12} & K^{II}_{13} & K^{II}_{14} & K^{II}_{15} & K^{II}_{16} \\ & K^{II}_{22} & K^{II}_{23} & K^{II}_{24} & K^{II}_{25} & K^{II}_{26} \\ & & K^{II}_{33} & K^{II}_{34} & K^{II}_{35} & K^{II}_{36} \\ & & & K^{II}_{44} & K^{II}_{45} & K^{II}_{46} \\ & & & & K^{II}_{55} & K^{II}_{56} \\ & & & & & K^{II}_{66} \end{bmatrix} \begin{bmatrix} q_4 \\ q_5 \\ q_6 \\ q_7 \\ q_8 \\ q_9 \end{bmatrix} = \begin{bmatrix} F^{II}_1 \\ F^{II}_2 \\ F^{II}_3 \\ F^{II}_4 \\ F^{II}_5 \\ F^{II}_6 \end{bmatrix} \qquad (3.2)$$

Stiffness coefficients corresponding to variables 4, 5, and 6 are added directly to those coefficients in core contributed by element 1 . Coefficients corresponding to the rest of the variables of element 2 are assembled into new positions in the front.

$$\begin{bmatrix} G_{11} & G_{12} & G_{13} & K^{II}_{14} & K^{II}_{15} & K^{II}_{16} \\ & G_{22} & G_{23} & K^{II}_{24} & K^{II}_{25} & K^{II}_{26} \\ & & G_{33} & K^{II}_{34} & K^{II}_{35} & K^{II}_{36} \\ & & & K^{II}_{44} & K^{II}_{45} & K^{II}_{46} \\ & & & & K^{II}_{55} & K^{II}_{56} \\ & & & & & K^{II}_{66} \end{bmatrix} \begin{bmatrix} q_4 \\ q_5 \\ q_6 \\ q_7 \\ q_8 \\ q_9 \end{bmatrix} = \begin{bmatrix} Q_1 \\ Q_2 \\ Q_3 \\ F^{II}_4 \\ F^{II}_5 \\ F^{II}_6 \end{bmatrix} \qquad (3.3)$$

Equation 3.3 shows what is contained in the front after the addition of the stiffnesses and loads of element 2 to the reduced equations of element 1, where

$$G_{11} = K^{II}_{11} + K^{I\prime}_{44}$$

$$G_{12} = K^{II}_{12} + K^{I\prime}_{45}$$

$$G_{13} = K^{II}_{13} + K^{I\prime}_{46}$$

$$
\begin{aligned}
G_{22} &= K^{II}_{22} + K^{I)}_{55} \\
G_{23} &= K^{II}_{23} + K^{I)}_{56} \qquad (3.4) \\
G_{33} &= K^{II}_{33} + K^{I)}_{66} \\
Q_1 &= F^{II}_{1} + F^{I)}_{4} \\
Q_2 &= F^{II}_{2} + F^{I)}_{5} \\
Q_3 &= F^{II}_{3} + F^{I)}_{6}
\end{aligned}
$$

The variation of the front as it advances from element 1 to element 2 is graphically presented in figure 3.2, where variables 4, 5, and 6 are contained in the front as it moves from element 1 to element 2.

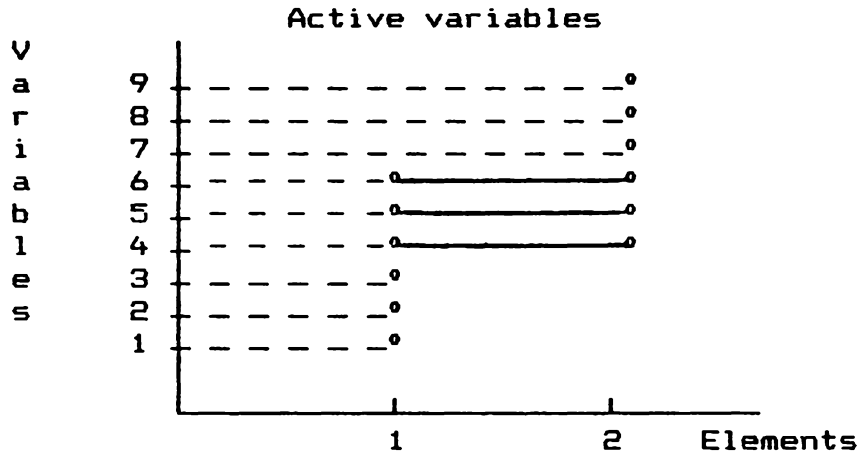


Fig.3.2 variation of the front

After the last element is assembled, the frontwidth is reduced to zero by eliminating all the variables whose stiffness coefficients are still in core. The back-substitution process begins and variables are backsolved for by reading their numbers (nicknames), and positions in the front, and their coefficients from the out-of-core file.

A more detailed description of the individual components of the method is presented in sections 3.3 to 3.8.

3.3 The prefront

Because of the nature of this technique, it is essential to know when a variable can be eliminated. To accomplish this, each node number is recorded with a negative sign upon its last appearance in the element incidence matrix, LNODS(I,N), in which I is the element number and N is the number of nodes to which the element is connected . For frame element N is equal to 2; 1 for the a-end and 2 for b-end. For example, consider the three elements frame shown in figure 3.3

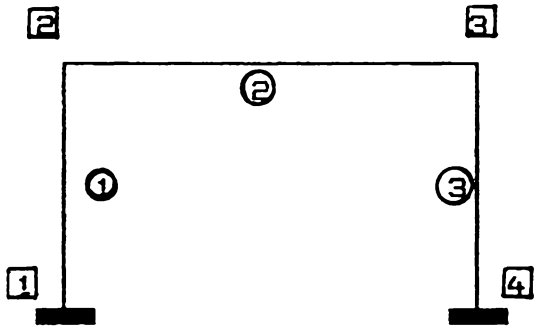


Figure 3.3 Example Problem

After performing the prefront operation, LNODS(I,N) is modified to contain negative signs as shown below.

ELEMENT	NODES 1	2
1	-1	2
2	-2	3
3	-3	-4

3.4 Housekeeping

To keep track of element degrees of freedom as the front moves over the elements, three addresses of each degree of freedom have to be recorded. These addresses are monitored by the vectors described below.

3.4.1 The Local Element Variables Vector, LOCEL.

LOCEL is a vector which contains the system degrees of freedom associated with a particular element. The length of this vector is equal to the number of the element degrees of freedom. It is computed for each element in turn prior to the assembly of that element. Each degree of freedom making a final appearance (detected by the negative number of the node associated with that degree of freedom) is entered in LOCEL with a negative sign. Variables are computed according to the following algorithm:

```
LOCEL(J) = (LNODS(I,N)-1)*NDN+K
```

This formula states that the degree of freedom associated with the Kth displacement of the Nth node of the Ith element is stored in the Jth location in LOCEL, where NDN is the number of the degrees of freedom per joint, and J ranges over the number of element degrees of freedom. For

frame elements the degrees of freedom of the a-end are assigned in the first three positions of LOCEL, and the degrees of freedom of the b-end are assigned in the second three positions of LOCEL.

3.4.2 The Vector of Active Variables, NACVA.

NACVA is a vector of active variables contained in core. The length of this vector changes as the front moves over the structure; its maximum length is structure dependent, and it is difficult to predict for complex structures. Therefore, a machine dependent upper limit known as the maximum frontwidth is predetermined; this maximum front is the maximum number of variables that can be contained in core.

As soon as the variables in NACVA are eliminated they are replaced by zeros; zeros serve as indicators of available space that can be used for the assembly of the next element.

3.4.3 The Destination Vector, NDEST.

The positions of newly assigned variables in the NACVA vector are recorded in this vector. Like LOCEL it has a fixed length which is equal to the number of degrees of freedom of the element, and it is computed for every element prior to the assembly process of the element. The frame in figure 3.1 is used to illustrate how the three vectors monitor the variables of each element. Before the computation of any housekeeping vector, the element

incidence matrix has to be modified as shown below. 1. The element incidence matrix for the frame in figure 3.1 is

LNODS(2,2)=

-1	2
-2	-3

1. Initialize LOCEL, NACVA and NDEST to zero. The current frontwidth is zero.
2. Assemble element 1.
   a. Node 1 and 2 become active.
   b. Variables 1 through 6 are stored in LOCEL.

LOCEL =

-1	-2	-3	4	5	6

   c. Since NACVA is a null vector, the 6 variables are stored in the first 6 positions, and the front-width is increased to 6.

NACVA =

1	2	3	4	5	6

   d. The addresses of the variables in NACVA are stored in NDEST.

NDEST =

1	2	3	4	5	6

   e. Since variables 1, 2, and 3 are preceded by negative signs in LOCEL, they can be eliminated and replaced by zeros in NACVA.

NACVA = | Ø | Ø | Ø | 4 | 5 | 6 |

The frontwidth would have been adjusted had the positions of the eliminated variables in NACVA been at the end.

3. Assemble element 2.

a. Node 3 is activated.

b. Variables 4 through 9 are stored in LOCEL.

LOCEL = | -4 | -5 | -6 | -7 | -8 | -9 |

C. For each variable in turn, an available space in NACVA is searched for. Variables 4, 5, and 6 are already in NACVA; therefore, 7, 8, and 9 will be stored in the first available three positions.

NACVA = | 7 | 8 | 9 | 4 | 5 | 6 |

d. The addresses in NACVA of the variables of element 2 as they appear in LOCEL will be stored in NDEST.

NDEST = | 4 | 5 | 6 | 1 | 2 | 3 |

As soon as the last element is assembled, the active variables are eliminated and their stiffness coefficients are transferred to an out-of-core file.

3.5 The assembly process

Using the NDEST vector, the element stiffness matrix, ESTIF, and the load vector, ELOAD, are assembled into the

global stiffness matrix, GSTIF, and load vector, GLOAD, in much the same fashion as in the direct stiffness method (11). Because of symmetry, only the upper triangular half is assembled into GSTIF. To improve program efficiency, GSTIF is converted from a two dimensional matrix to a one dimensional array (2). The conversion is achieved via the formula

$$N = (J(J-1)/2)+I$$

in which I and J are the row and column numbers in the two dimensional matrix, and N is the position in the one dimensional array.

3.6 The elimination process

In the frontal technique, unlike ordinary Gauss elimination, variables are not eliminated in the same order in which they are encountered; they are eliminated upon last appearance. To eliminate a prescribed variable, the right hand side is modified by subtracting the effect of the prescribed variable, and setting to zero the column, except the diagonal element which corresponds to that variable.

$$\begin{bmatrix} K_{11} & K_{12} & K_{13} & K_{14} & K_{15} & K_{16} \\ & K_{22} & K_{23} & K_{24} & K_{25} & K_{26} \\ & & K_{33} & K_{34} & K_{35} & K_{36} \\ & & & K_{44} & K_{45} & K_{46} \\ & & & & K_{55} & K_{56} \\ & & & & & K_{66} \end{bmatrix} \cdot \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \end{bmatrix} \qquad (3.5)$$

If, for example, q is a prescribed displacement, Eq. 3.5 is reduced to Eq. 3.6.

$$\begin{bmatrix} K_{11} & K_{12} & \emptyset & K_{14} & K_{15} & K_{16} \\ K_{21} & K_{22} & \emptyset & K_{24} & K_{25} & K_{26} \\ K_{31} & K_{32} & K_{33} & K_{34} & K_{35} & K_{36} \\ K_{41} & K_{42} & \emptyset & K_{44} & K_{45} & K_{46} \\ K_{51} & K_{52} & \emptyset & K_{54} & K_{55} & K_{56} \\ K_{61} & K_{62} & \emptyset & K_{64} & K_{65} & K_{66} \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \end{bmatrix} = \begin{bmatrix} F_1 - q_3 * K_{13} \\ F_2 - q_3 * K_{23} \\ R_3 \\ F_4 - q_3 * K_{43} \\ F_5 - q_3 * K_{53} \\ F_6 - q_3 * K_{63} \end{bmatrix} \qquad (3.6)$$

where R_3 is the applied concentrated force that causes the prescribed displacement; its value can only be computed after the back-substitution process.

Free variables are eliminated by Gauss elimination; for example, if q_3 is a free variable and to be eliminated, then Eq. 3.5 is reduced to Eq. 3.7.

$$
\begin{bmatrix}
K_{11}-\frac{K_{31}}{K_{33}}K_{13} & K_{12}-\frac{K_{32}}{K_{33}}K_{13} & \phi & K_{14}-\frac{K_{34}}{K_{33}}K_{13} & K_{15}-\frac{K_{35}}{K_{33}}K_{13} & K_{16}-\frac{K_{36}}{K_{33}}K_{13} \\
K_{21}-\frac{K_{31}}{K_{33}}K_{23} & K_{22}-\frac{K_{32}}{K_{33}}K_{23} & \phi & K_{24}-\frac{K_{34}}{K_{33}}K_{23} & K_{25}-\frac{K_{35}}{K_{33}}K_{23} & K_{26}-\frac{K_{36}}{K_{33}}K_{23} \\
K_{31} & K_{32} & K_{33} & K_{34} & K_{35} & K_{36} \\
K_{41}-\frac{K_{31}}{K_{33}}K_{43} & K_{42}-\frac{K_{32}}{K_{33}}K_{43} & \phi & K_{44}-\frac{K_{34}}{K_{33}}K_{43} & K_{45}-\frac{K_{35}}{K_{33}}K_{43} & K_{46}-\frac{K_{36}}{K_{33}}K_{43} \\
K_{51}-\frac{K_{31}}{K_{33}}K_{53} & K_{52}-\frac{K_{32}}{K_{33}}K_{53} & \phi & K_{54}-\frac{K_{34}}{K_{33}}K_{53} & K_{55}-\frac{K_{35}}{K_{33}}K_{53} & K_{56}-\frac{K_{36}}{K_{33}}K_{53} \\
K_{61}-\frac{K_{31}}{K_{33}}K_{63} & K_{62}-\frac{K_{32}}{K_{33}}K_{63} & \phi & K_{64}-\frac{K_{34}}{K_{33}}K_{63} & K_{65}-\frac{K_{35}}{K_{33}}K_{63} & K_{66}-\frac{K_{36}}{K_{33}}K_{63}
\end{bmatrix}
\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \end{bmatrix}
=
\begin{bmatrix}
F_1-\frac{F_3}{K_{33}}K_{13} \\
F_2-\frac{F_3}{K_{33}}K_{23} \\
F_3 \\
F_4-\frac{F_3}{K_{33}}K_{43} \\
F_5-\frac{F_3}{K_{33}}K_{53} \\
F_6-\frac{F_3}{K_{33}}K_{63}
\end{bmatrix}
\qquad (3.7)
$$

3.7 The backsubstitution process

This process is identical to that in Gauss elimination, except that in the frontal technique equations are read from an out-of-core file.

3.8 Algorithm of the frontal method

The algorithm of the frontal method may be presented in a nine step algorithm as shown below. The first eight steps are repeated for every element in turn, while step 9 is repeated for every load condition.

1. The three housekeeping vectors are computed:
   a. Local element variable vector, LOCEL
   b. Global active variable vector,NACVA
   c. Global destination vector, NDEST
2. The local element forces are either computed or read from a file and transformed to global element forces.
3. The element stiffness matrix is either computed or read from a file.
4. The element stiffness matrix and load vector are assembled into the global stiffness matrix and load

vector via the destination vector.

5. Each element degree of freedom is tested to determine if it can be eliminated.
6. Any prescribed degrees of freedom qk is eliminated by subtracting its effect from the right hand side:

   $Q_i = Q_i - K_{iK} * q_K$, and by setting to zero the column that corresponds to that degree of freedom except the diagonal element.
7. Free variables are eliminated by Gauss elimination. To eliminate qk, all coefficients in the front, except those in Kth row, are modified as follow:

   $K_{ij} = K_{ij} - K_{kj} * K_{iK} / K_{KK}$

   $Q_i = Q_i - Q_K * K_{iK} / K_{KK}$
8. Eliminated degrees of freedom, their positions in NACVA, and their corresponding stiffness coefficients are transferred to a peripheral storage.
9. Starting with the degree of freedom eliminated last, degrees of freedom and their stiffness coefficients are extracted from the peripheral storage and solved for.

3.9 Numerical example

The cable shown in figure 3.4 has been discretized to five elements, with each element having two degrees of freedom. The frontal technique is used to solve for the nodal displacements. The extensional stiffness of the cable is a unity. The reduced equations are stored in an out-of-core file. The modified element incidence matrix

for the cable is shown below.

LNODS(5,2) =

-1	2
-2	3
-3	4
-4	5
-5	-6

The following three steps are followed for every variable to be eliminated.

1.Transfer the corresponding row of stiffness coefficients to an out_of_core storage.

2.Eliminate the variable.

3.Replace the variable by a zero in NACVA.

Assemble element 1 :

LOCEL =

-1	2

NACVA =

1	2

NDEST =

1	2

$$\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} = \begin{bmatrix} \emptyset \\ \emptyset \end{bmatrix} \qquad (3.8)$$

$$\begin{bmatrix} 1 \end{bmatrix} \begin{bmatrix} q2 \end{bmatrix} = \begin{bmatrix} \emptyset \end{bmatrix} \qquad (3.8a)$$

NACVA =

ø	2

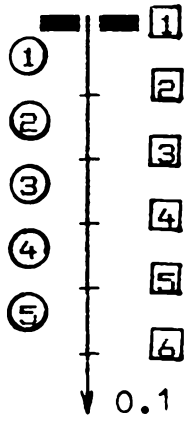


Figure 3.4 Example Problem

Assemble element 2 :

LOCEL =	-2	3
NACVA =	3	2
NDEST =	2	1

$$\begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix}\begin{bmatrix} q_3 \\ q_2 \end{bmatrix}=\begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad (3.9)$$

$$\begin{bmatrix} 1/2 \end{bmatrix}\begin{bmatrix} q_3 \end{bmatrix}=\begin{bmatrix} 0 \end{bmatrix} \qquad (3.9a)$$

NACVA =	3	0

Assemble element 3 :

LOCEL =	-3	4
NACVA =	3	4
NDEST =	1	2

$$\begin{bmatrix} 3/2 & -1 \\ -1 & 1 \end{bmatrix}\begin{bmatrix} q_3 \\ q_4 \end{bmatrix}=\begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad (3.10)$$

$$\begin{bmatrix} 1/3 \end{bmatrix}\begin{bmatrix} q_4 \end{bmatrix}=\begin{bmatrix} 0 \end{bmatrix} \qquad (3.10a)$$

NACVA =	0	4

Assemble element 4 :

LOCEL =	-4	5
NACVA =	5	4
NDEST =	2	1

$$\begin{bmatrix} 1 & -1 \\ -1 & 4/3 \end{bmatrix} \begin{bmatrix} q_5 \\ q_4 \end{bmatrix} = \begin{bmatrix} \emptyset \\ \emptyset \end{bmatrix} \qquad (3.11)$$

$$\begin{bmatrix} 1/4 \end{bmatrix} \begin{bmatrix} q_5 \end{bmatrix} = \begin{bmatrix} \emptyset \end{bmatrix} \qquad (3.11a)$$

NACVA =	5	Ø

Assemble element 5 :

LOCEL =	-5	6
NACVA =	5	6
NDEST =	1	2

$$\begin{bmatrix} 5/4 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} q_5 \\ q_6 \end{bmatrix} = \begin{bmatrix} \emptyset \\ \emptyset.1 \end{bmatrix} \qquad (3.12)$$

$$\begin{bmatrix} 1/5 \end{bmatrix} \begin{bmatrix} q_6 \end{bmatrix} = \begin{bmatrix} \emptyset.1 \end{bmatrix} \qquad (3.12a)$$

NACVA =	Ø	6

As q_6 is eliminated, the frontwidth reduces to zero. The backup file contains the following information.

1	−1	q_1	Ø
2	−1	q_2	Ø
3/2	−1	q_3	Ø
4/3	−1	q_4	Ø
5/4	−1	q_5	Ø
	1/5	q_6	Ø.1

The back substitution results

$q_6 = Ø.5$, $q_5 = Ø.4$, $q_4 = Ø.3$, $q_3 = Ø.2$, $q_2 = Ø.1$, $q_1 = Ø.Ø$.

CHAPTER 4

PROGRAM DEVELOPMENT

4.1 Introduction

In this chapter, a design of a computer program to analyze frame structures with a large number of degrees of freedom is presented. The program is based on the program (CE4002) presented in figure 4.1 and written by Holzer (11). The Frontal technique is employed to solve for the unknown joint displacements.

4.2 Program Design

The program structure is presented in figure 4.2 . A list of variables names is presented, and the modified and new subroutines are described by Nassi-Schneiderman (NS) diagrams (11) with a list of input and output arguments. The program listing is presented in Appendix A. Dimensions of all arrays in the main program are controlled by the parameters MXE, MXJ, MFRON, MSTIF, and MXNEQ. Adjustable arrays are used throughout the program. The program is written in FORTRAN 77.

4.3 Symbolic Names

Listed below are the names of the program parameters and variables. The list includes the names used in CE4002 program and other new descriptive names used in the new subroutines.

Parameters

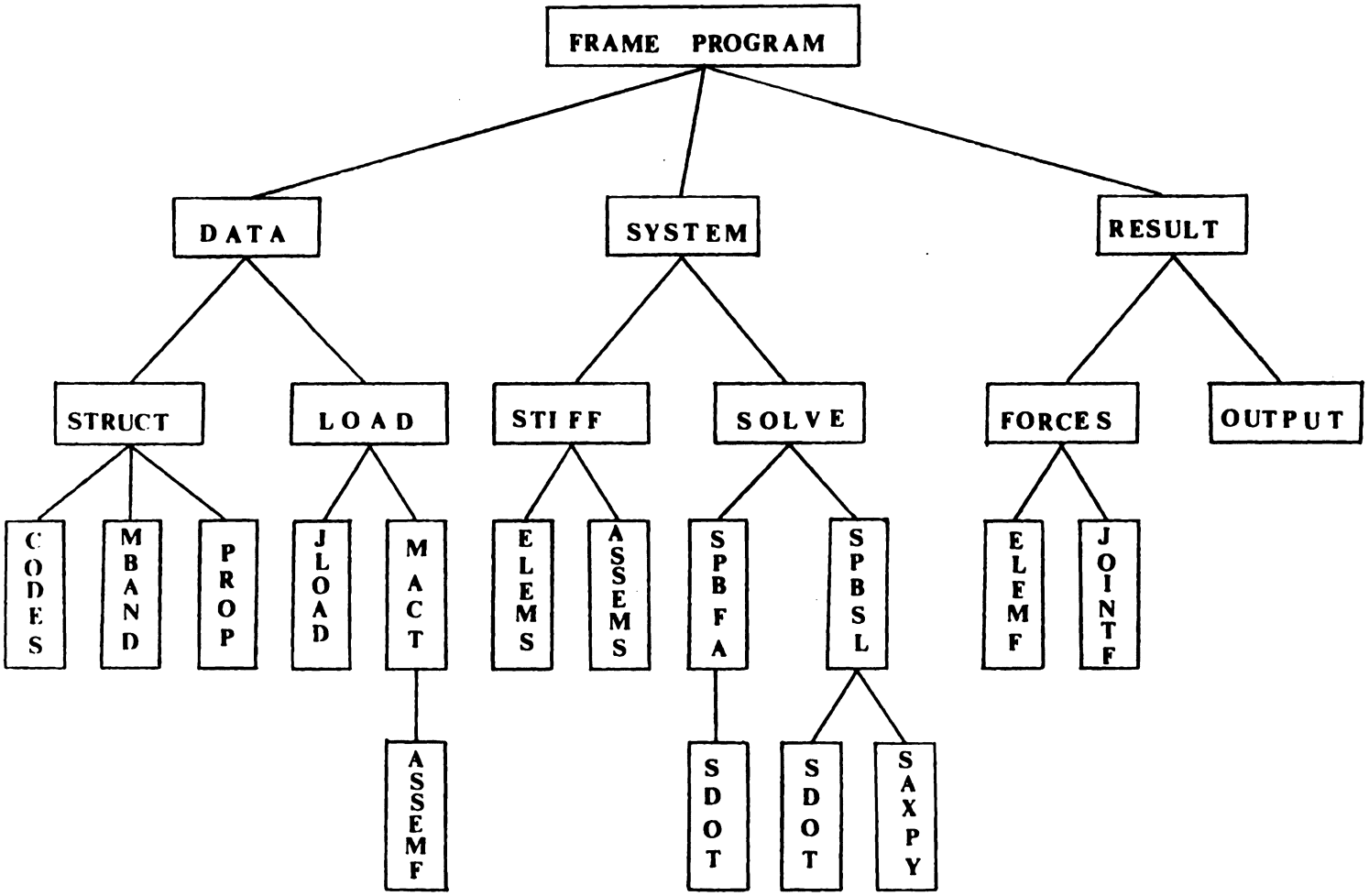


Fig. 4.1 CE4002 Program Structure

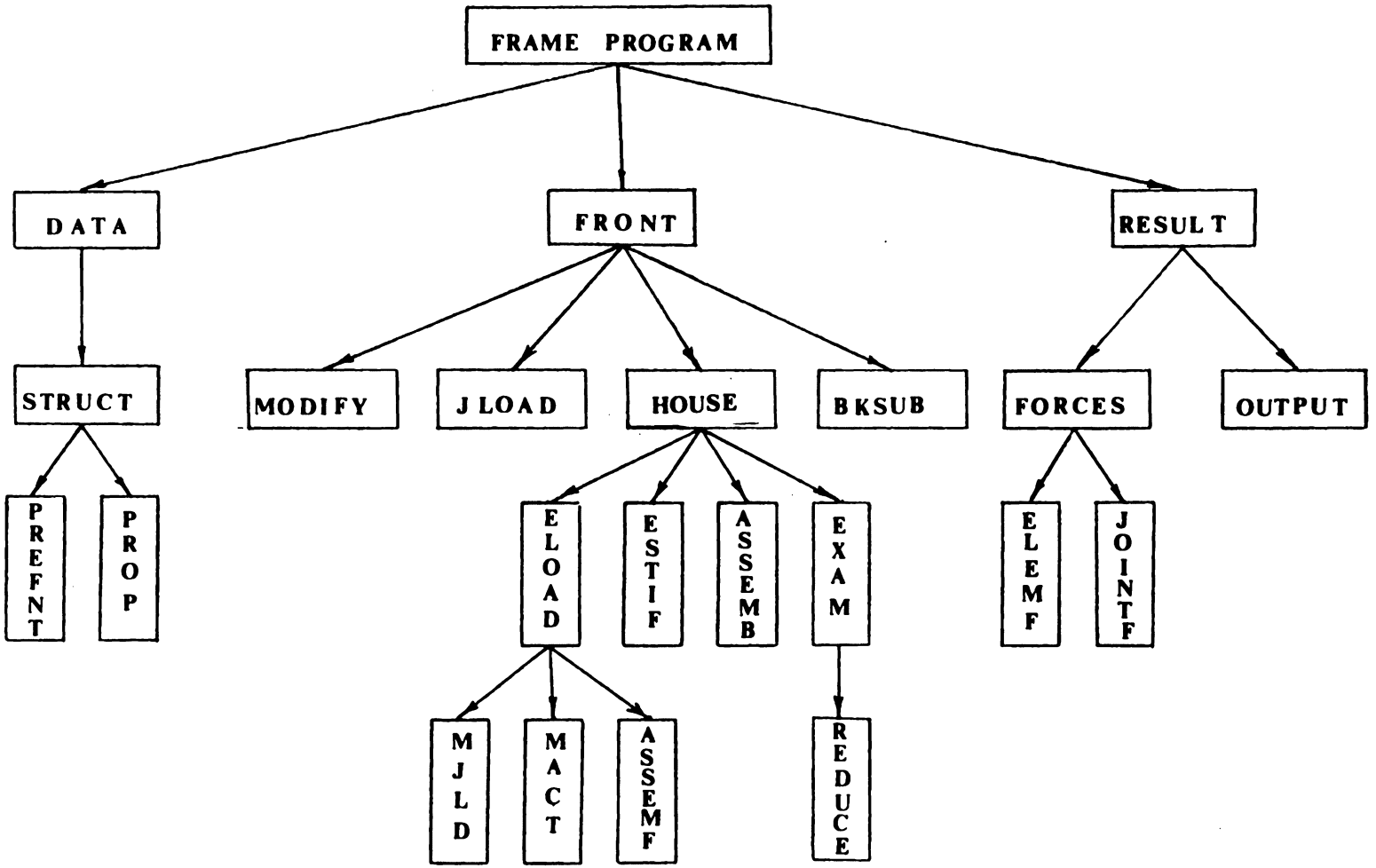


Fig. 4.2 Frontal Solver Structure

MXE maximum number of elements

MXJ maximum number of joints

MFRON maximum number of variables that can be contained in core

MSTIF maximum length of the global stiffness vector

MXNEQ maximum number of equations

Variables

AREA	Element cross sectional area
C1 ,C2	Element directional cosines
CTE	coefficient of thermal expansion
DISP	prescribed joint displacement
ELENG	Element length
EMOD	Element modulus of elasticity
ESTIF(6,6)	Element stiffness matrix
EQ(MFRON)	Vector in which the eliminated equations are stored prior to writing to a backup file
F(6)	Global element end forces including applied joint forces
FFIX(MXNEQ)	Global vector that defines free and prescribed degrees of freedom
FORCE	Applied joint force
G(7)	Global element stiffness coefficients
GL(MFRON)	Global load vector contained in core
GSTIF(MSTIF)	Global stiffness matrix stored in one dimensional vector in core (the front)
IACT	Counter for heading control

INDEX(6,6)	Index matrix
INE	Index of element number
JDIR	Joint direction
JNUM	Joint number
LC	Load condition
LOCEL(6)	Local element variables vector
MAT	Member action type
MINC(2,MX)	Number incidence matrix
	MINC(1,I) = joint at the a-end of element I
	MINC(2,I) = joint at the b-end of element I
MN	Member number
NA	Counter for number of actions on element
NACVA(MFRON)	Vector containing a list of the
	active variables in the front
NDOFE	Index of degrees of freedom per element
INDOFJ	Index of degrees of freedom per joint
NDEST(6)	Destination vector for element variables
NE	Number of elements
NJ	Number of joints
NLC	Number of load conditions
Q(MXNEQ)	Joint loads vector, displacement vector
QBAR(6)	Local element applied joint load
QHAT(6)	Element fixed end forces
P(3)	Joint force vector
VECRV(MFRON)	Vector of running variables in which
	the solved displacements are temporarly stored

X(2),Y(2) Joint coordinates vectors

Z Moment of inertia about the local Z(3) axis of elements

4.4 Modifications of CE4002

Although a number of the original subroutines of CE4002 are employed in this program, minor modifications have been made. The joint code array, JCODE, has been implicitly used through the formula

$$K = (J-1)*3+I$$

Where K is the number of the system degree of freedom corresponding to the Ith displacement the Jth joint. Subroutine MACT has been expanded to compute fixed-end forces of elements due to five different action types. These member action types are presented in table 4.1 . The fixed-end forces due to the fourth member action type can be computed by equations 4.1 and 4.2.

$$Fa = -P1(X2-X1)(1-(X1+X2)/2EL)- (P2-P1)((X2-X1)/2)(1-(X1+2X2)/2EL) \quad (4.1)$$

$$Fb = -P1(X2-X1)((X1+X2)/(2EL))-(P2-P1) (X2-X1)/2)((X1+2X2)/3EL \quad (4.2)$$

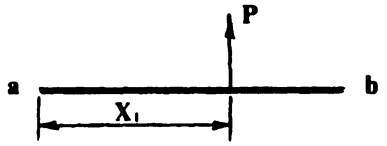
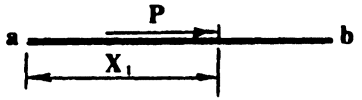
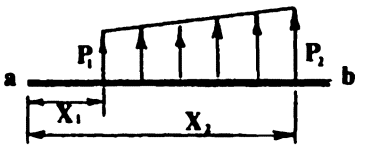
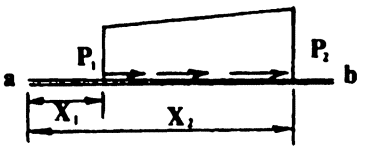
TYPE	MEMBER ACTIONS	X_1	P_1	X_2	P_2
1		X_1	P	0	0
2		X_1	P	0	0
3		X_1	P_1	X_2	P_2
4		X_1	P_1	X_2	P_2
5	UNIFORM TEMPERATURE CHANGE	0	ΔT	0	0

Table 4.1 Member Action Types

MODIFIED SUBROUTINES

Subroutine Struct

Function: Read and echo the member incidences, MINC(L,I); Initialize the fixity code array,FFIX, to 10000; read and echo for each joint constraint the joint number, JNUM, and the joint direction, JDIR, and store a zero in the corresponding location in FFIX. Call PREFNT and PROP.

Input arguments : NE,NJ,MINC,FFIX

Output arguments: FFIX

NS_ diagram

WRITE HEADING FOR MEMBER INCIDENCES
DO FOR I = 1,NE
READ MINC(1,I), MINC(2,I)
WRITE MINC(1,I), MINC(2,I)
DO FOR J = 1,3*NJ
FFIX(J) = 10000.
WRITE HEADING FOR JOINT CONSTRAINTS
WHILE JNUM IS NOT EQUAL TO 0
WRITE JNUM,JDIR
JLOCA = (JNUM-1)*3+JDIR
FFIX(JLOCA) = 0.0
READ JNUM,JDIR
CALL PERFNT
CALL PROP

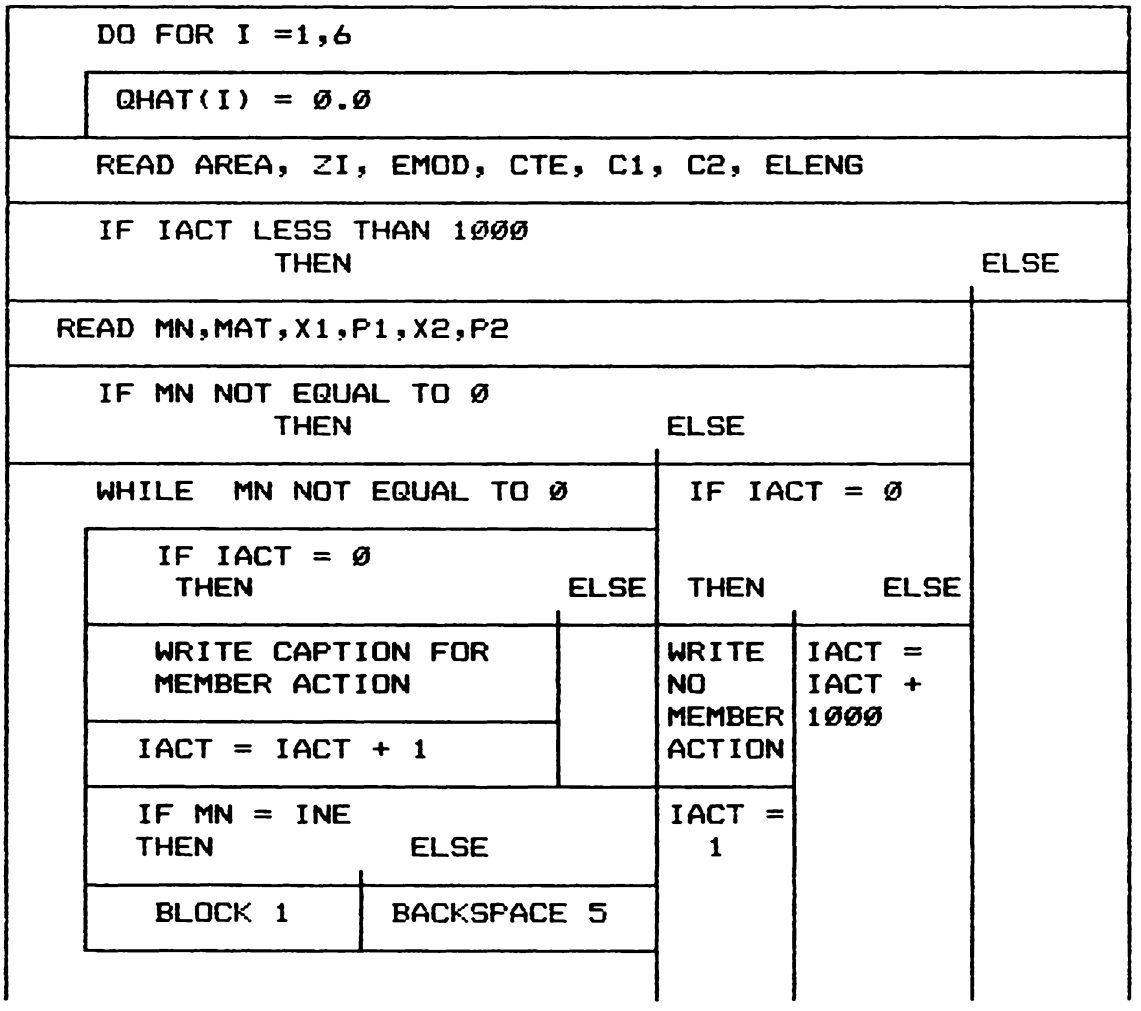
Subroutine MACT

Function: Initialize element fixed-end forces to zero. Read member properties, member number, MN, member action type, MAT, actions P1 and P2, and distances X1 and X2. Increment the action counter, compute fixed-end forces and call ASSEMF.

Input arguments :AREA, EMOD, ELENG, C1, C2, F, Qbar, QHAT, INE, IACT.

Output argument :QHAT,F

NS- diagram



WRITE QHAT(I), I=1,6

Block 1

IF MN =INE THEN				
WRITE MN,MAT,P1,X1,P2,X2				
NA = NA + 1				
SELECT CASE MAT=1	MAT=2	MAT=3	MAT=4	MAT=5
Compute and accomulate fixed-end forces by equations				
A.15 (11)	2.53 (12)	2.44 2.47 (12)	4.1 4.2	3.158 (11)
READ MN,MAT,X1,P1,X2,P2				

NEW SUBROUTINES

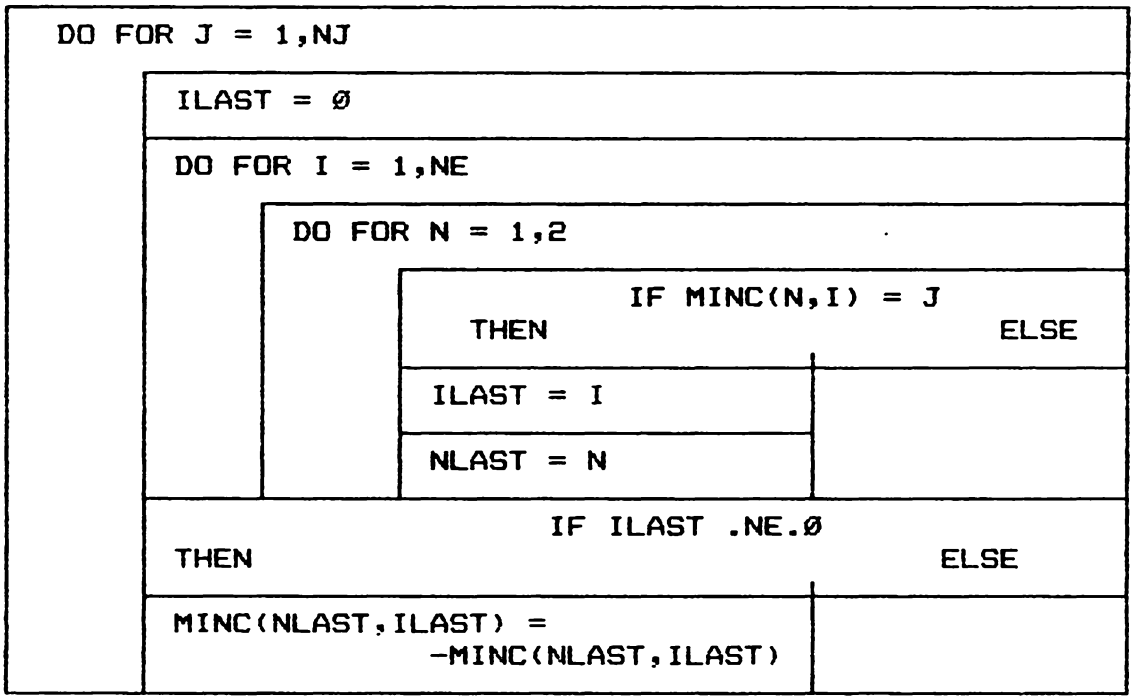
Subroutine PREFRONT

Function: Search the member incidence matrix and record the last appearance of a joint with a negative sign.

Input argument : NE, NJ, MINC

Output argument: MINC

NS_ diagram



Subroutine FRONT

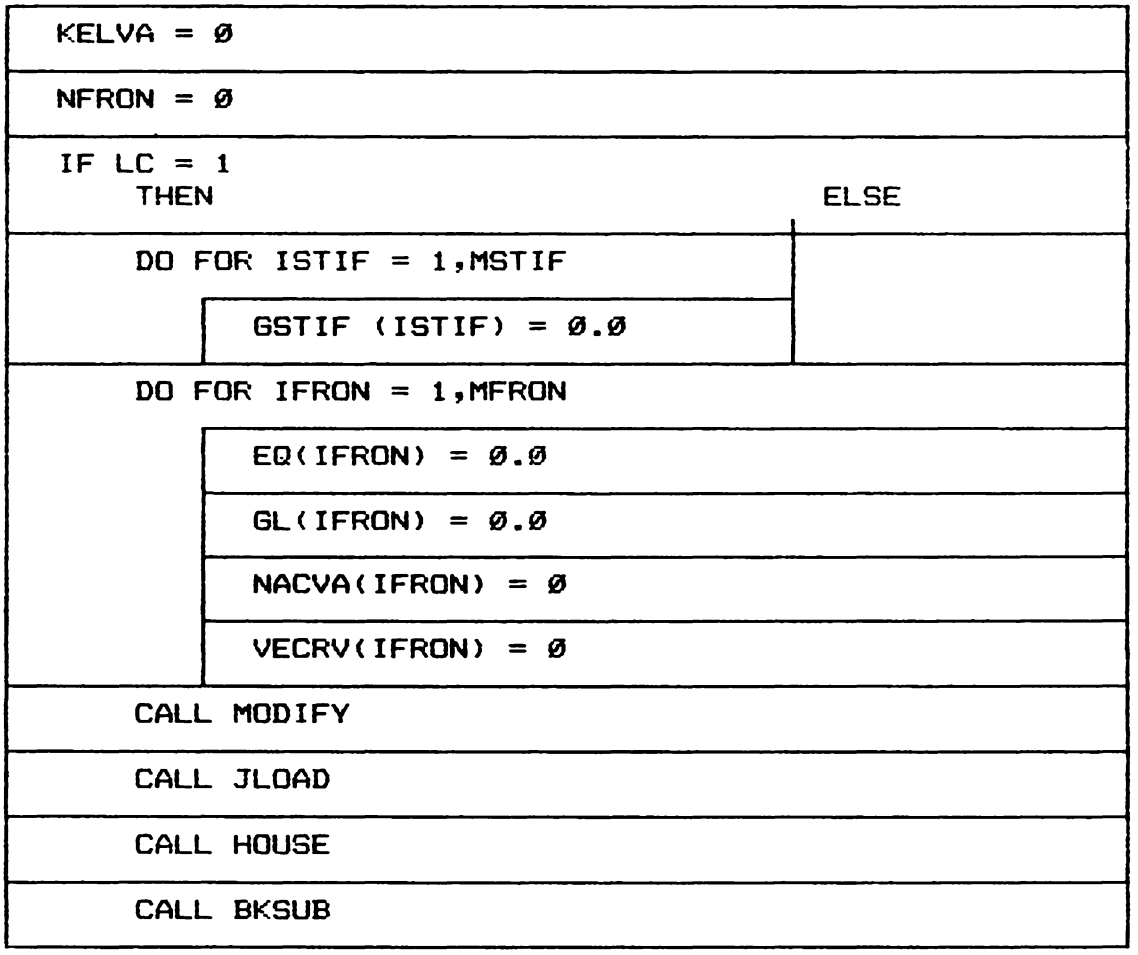
Function: Initialize to zero the number of eliminated variables, KELVA, and the number of variables in the front,NFRON. Initialize to zero the following vectors: NACVA, EQ, GL, VECRV and GSTIF.

Call subroutines MODIFY, JLOAD, HOUSE,and BKSUB

Input argument :Q, MINC, FFIX, NE, NJ, LC, MFRON, MSTIF, EQ, GL ,GSTIF, NACVA, VECRV.

Output argument: Q, FFIX.

NS_ diagram



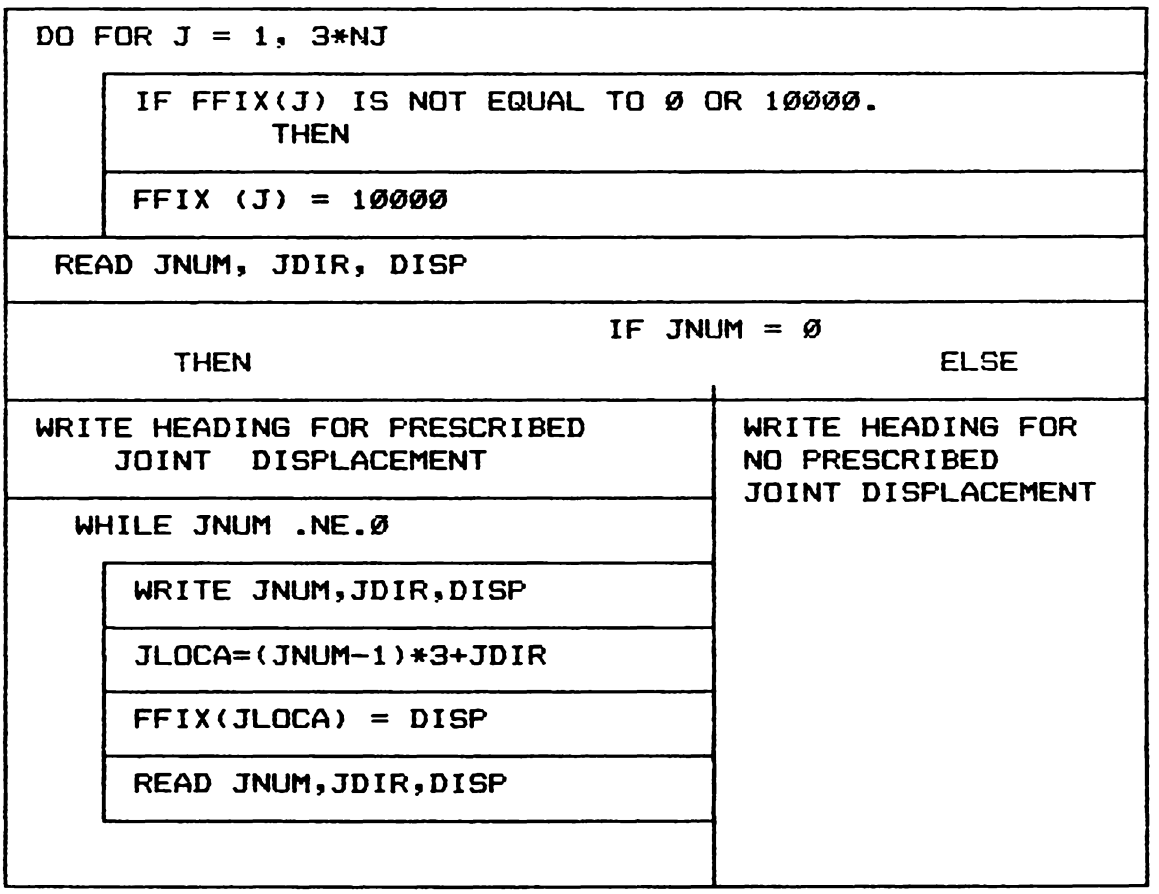
Subroutine MODIFY

FUNCTION: Prepare FFIX to receive new prescribed displacement. Read JNUM, JDIR, DISP for prescribed displacement. Store the prescribed joint displacement value of the corresponding degree of freedom in FFIX.

Input argument :NJ, FFIX

Output argument:FFIX

NS_ diagram



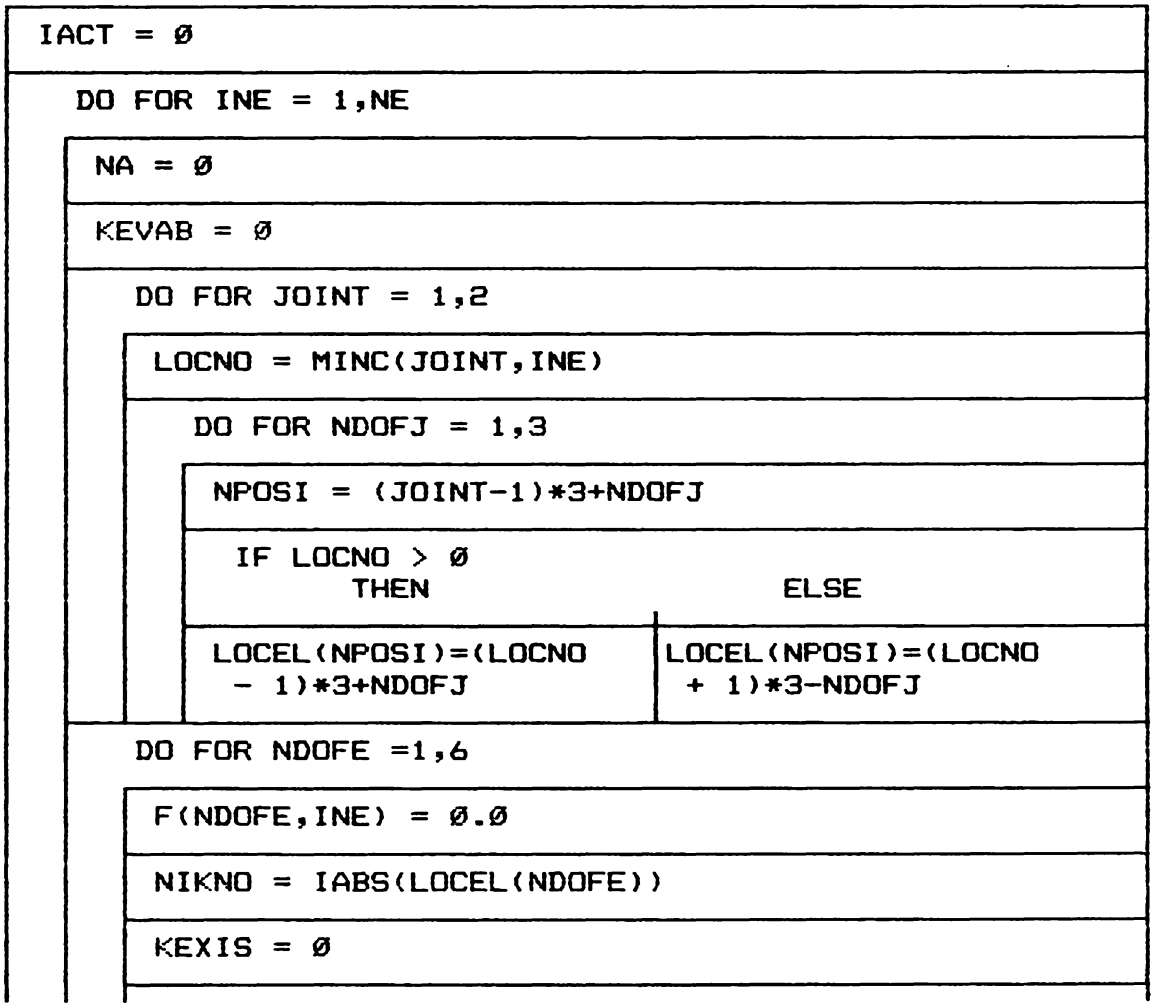
Subroutine HOUSE

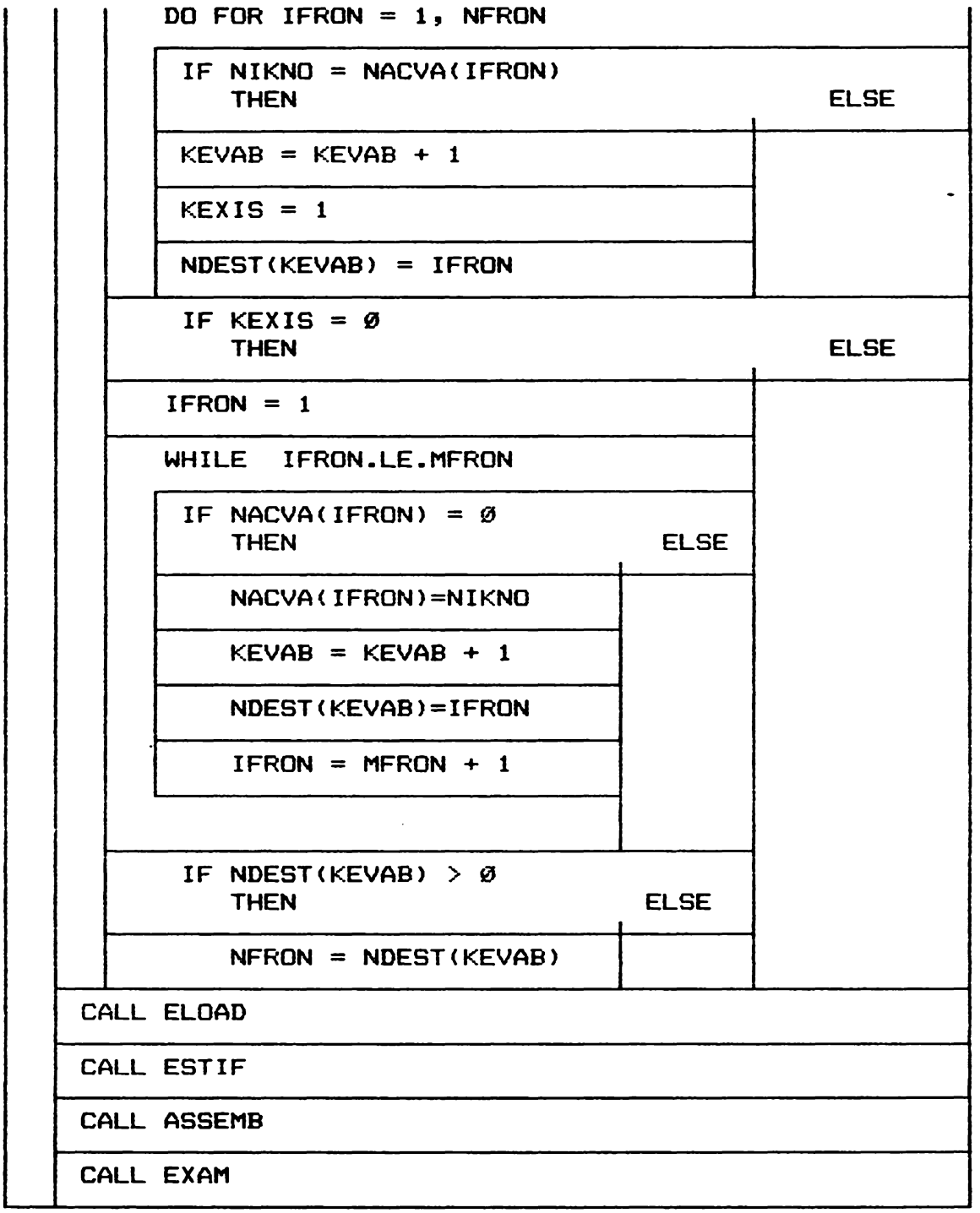
Function :Loop over all the elements and generate three arrays for each element: LOCEL,NDEST and NACVA; call subroutines ELOAD, ESTIF, ASSEMB, and EXAM.

Input arguments :MINC, FFIX, Q, NACVA, NDEST, MFRON, NFRON, NE, LC, GSTIF, GL, EQ, KELVA, F, NA

Output arguments:NACVA, NDEST, LOCEL, NFRON, and an out-of-core file containing eliminated variables, and the corresponding coefficients.

NS_ diagram




DO FOR IFRON = 1, NFRON
IF NIKNO = NACVA(IFRON)
THEN
ELSE
KEVAB = KEVAB + 1
KEXIS = 1
NDEST(KEVAB) = IFRON
IF KEXIS = Ø
THEN
ELSE
IFRON = 1
WHILE IFRON.LE.MFRON
IF NACVA(IFRON) = Ø
THEN
ELSE
NACVA(IFRON)=NIKNO
KEVAB = KEVAB + 1
NDEST(KEVAB)=IFRON
IFRON = MFRON + 1
IF NDEST(KEVAB) > Ø
THEN
ELSE
NFRON = NDEST(KEVAB)
CALL ELOAD
CALL ESTIF
CALL ASSEMB
CALL EXAM

Subroutine ELOAD

Function: Call subroutines member joint load, MJLD, and member actions, MACT, and ASSEMF.

Input arguments :FFIX,MINC, INE, IACT, Q, F, NA

Output arguments:F, QBAR, QHAT

NS- diagram

CALL MJLD
CALL MACT
CALL ASSEMF

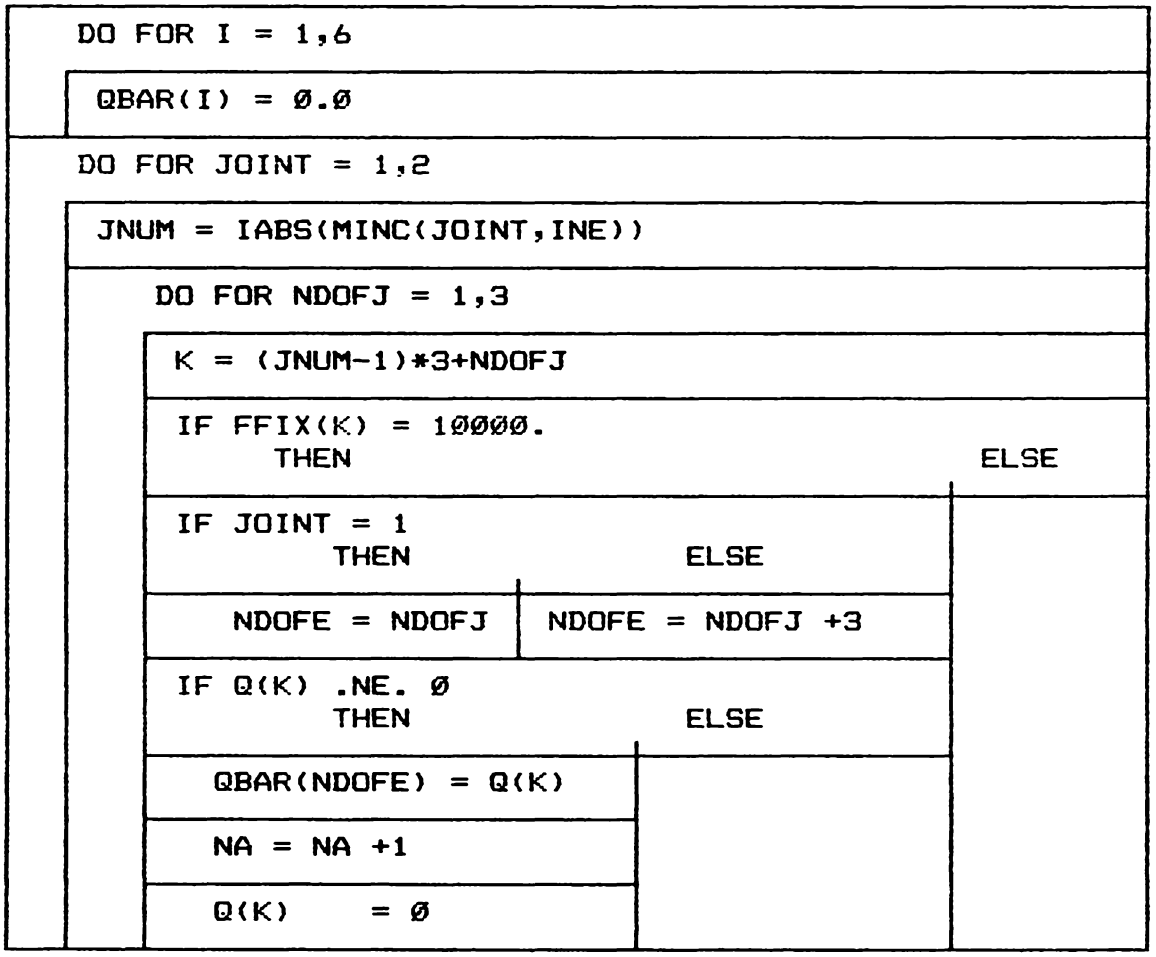
Subroutine MJLD

Function: Transform applied joint loads from global joint loads to local element joint loads. Initialize QBAR to zero.

Input arguments :Q, QBAR, FFIX, MINC, INE, NA

Output arguments:QBAR,NA

NS_ diagram



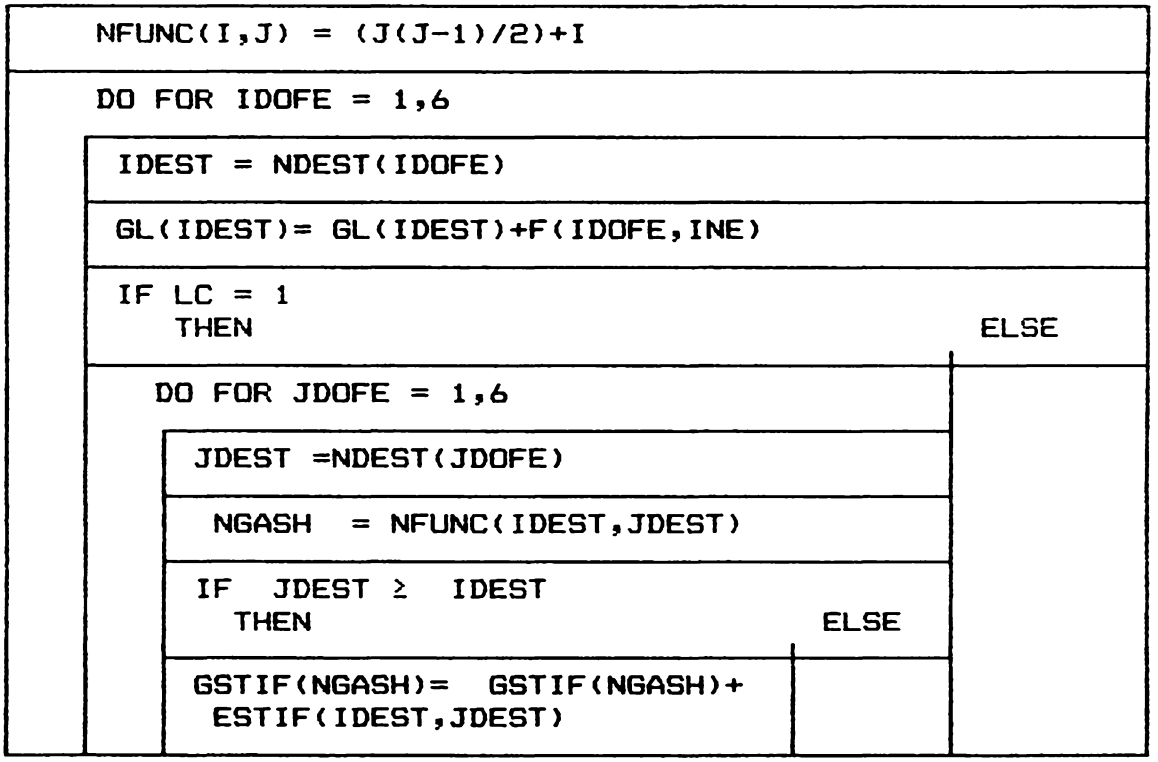
Subroutine ASSEMB

Function :Assemble element loads, F, into a global load vector and the element stiffness matrix, ESTIF(6,6), into the global stiffness array, GSTIF, via the destination vector. For subsequent load conditions assemble only element loads.

Input arguments : ESTIF, GSTIF, NDEST, GL, F, LC, INE

Output arguments: GSTIF, GL

NS_ diagram



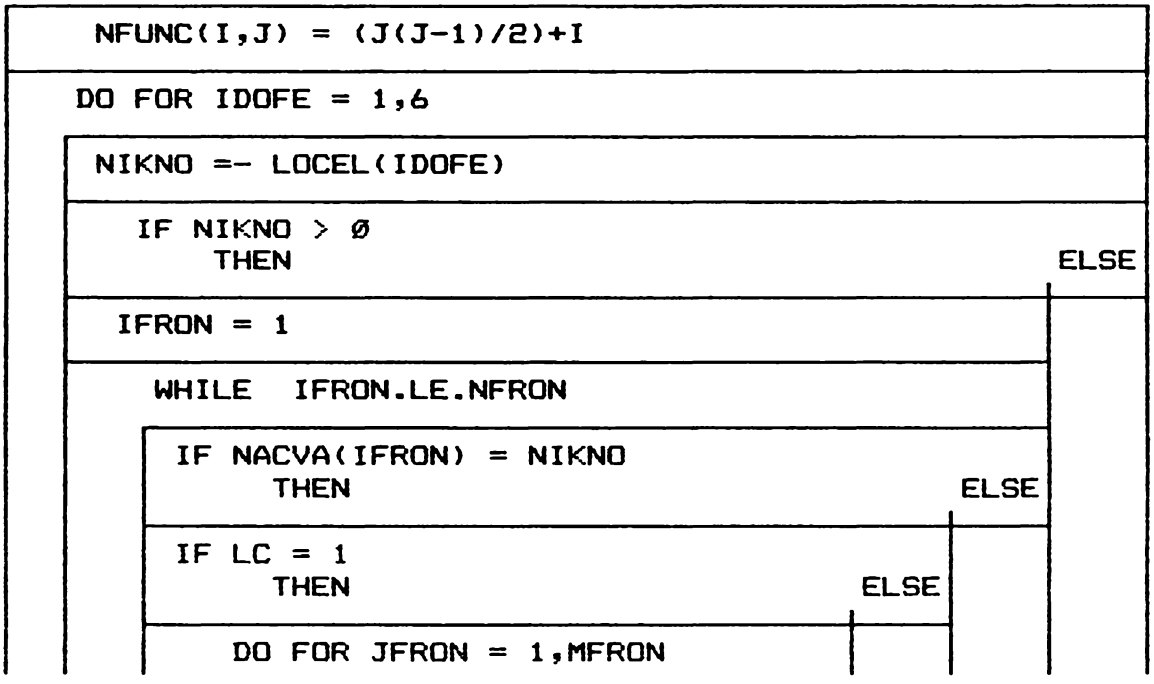
Subroutine EXAM

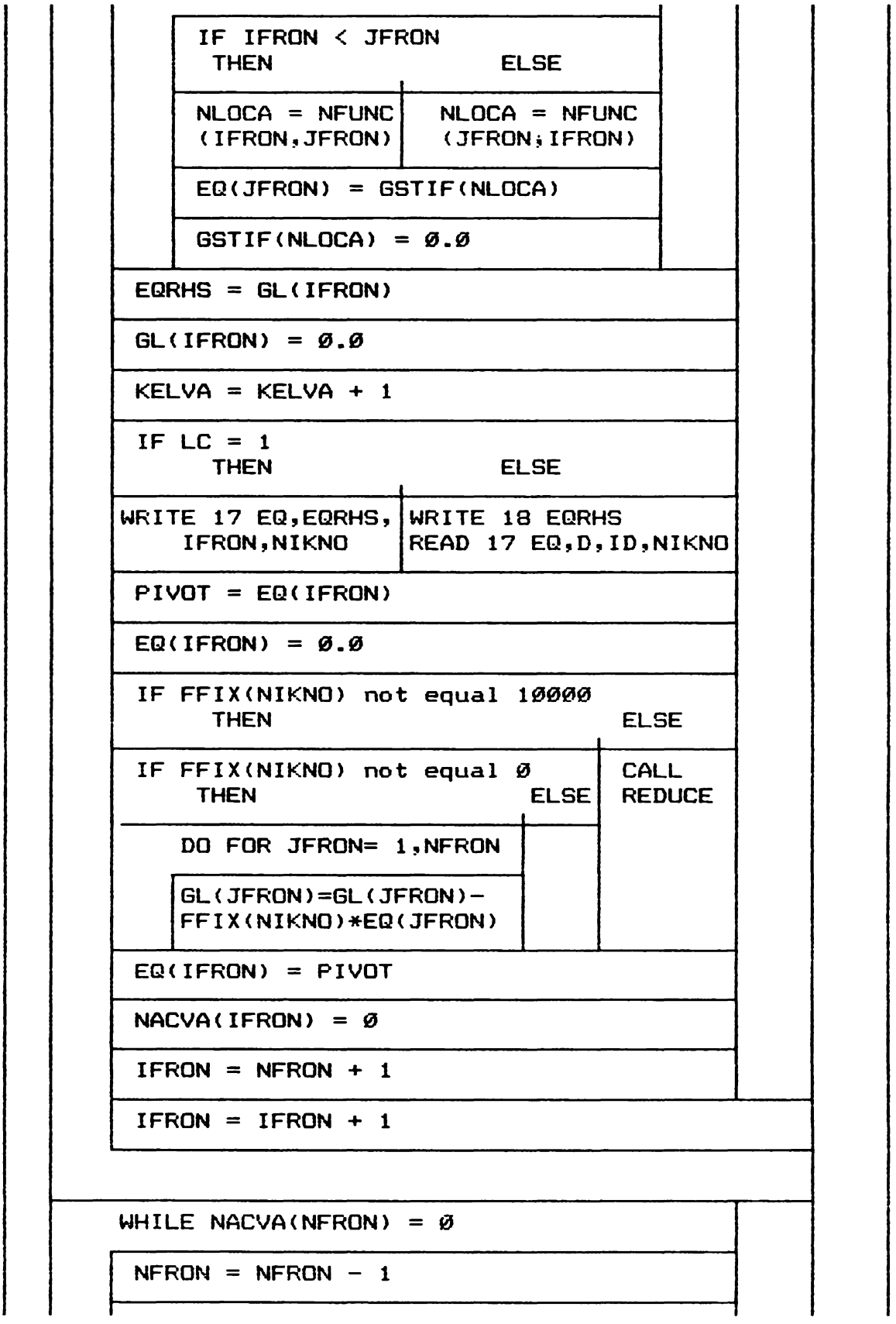
Function: Examine newly assembled variables and determine if they should be eliminated, that is, if the last element containing these variables has been assembled. Eliminate prescribed variables or call REDUCE to eliminate free variables. Write the stiffness coefficients of the eliminated variables to a temporary file; examine if the number of variables in the front can be reduced.

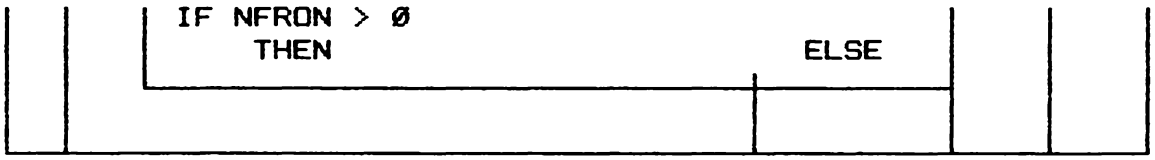
Input arguments :LOCEL, NACVA, MFRON, NFRON, EQ, GSTIF, GL, KELVA, LC, FFIX

Output arguments:GSTIF, GL, NFRON, and an out-of-core file containing the coefficients of the eliminated variables to a temporary file.

NS- diagram




IF IFRON < JFRON
THEN
ELSE
NLOCA = NFUNC (IFRON,JFRON)
NLOCA = NFUNC (JFRON;IFRON)
EQ(JFRON) = GSTIF(NLOCA)
GSTIF(NLOCA) = Ø.Ø
EQRHS = GL(IFRON)
GL(IFRON) = Ø.Ø
KELVA = KELVA + 1
IF LC = 1
THEN
ELSE
WRITE 17 EQ,EQRHS, IFRON,NIKNO
WRITE 18 EQRHS
READ 17 EQ,D,ID,NIKNO
PIVOT = EQ(IFRON)
EQ(IFRON) = Ø.Ø
IF FFIX(NIKNO) not equal 1ØØØØ
THEN
ELSE
IF FFIX(NIKNO) not equal Ø
THEN
ELSE
CALL REDUCE
DO FOR JFRON= 1,NFRON
GL(JFRON)=GL(JFRON)- FFIX(NIKNO)*EQ(JFRON)
EQ(IFRON) = PIVOT
NACVA(IFRON) = Ø
IFRON = NFRON + 1
IFRON = IFRON + 1
WHILE NACVA(NFRON) = Ø
NFRON = NFRON - 1


IF NFRON > Ø
THEN
ELSE

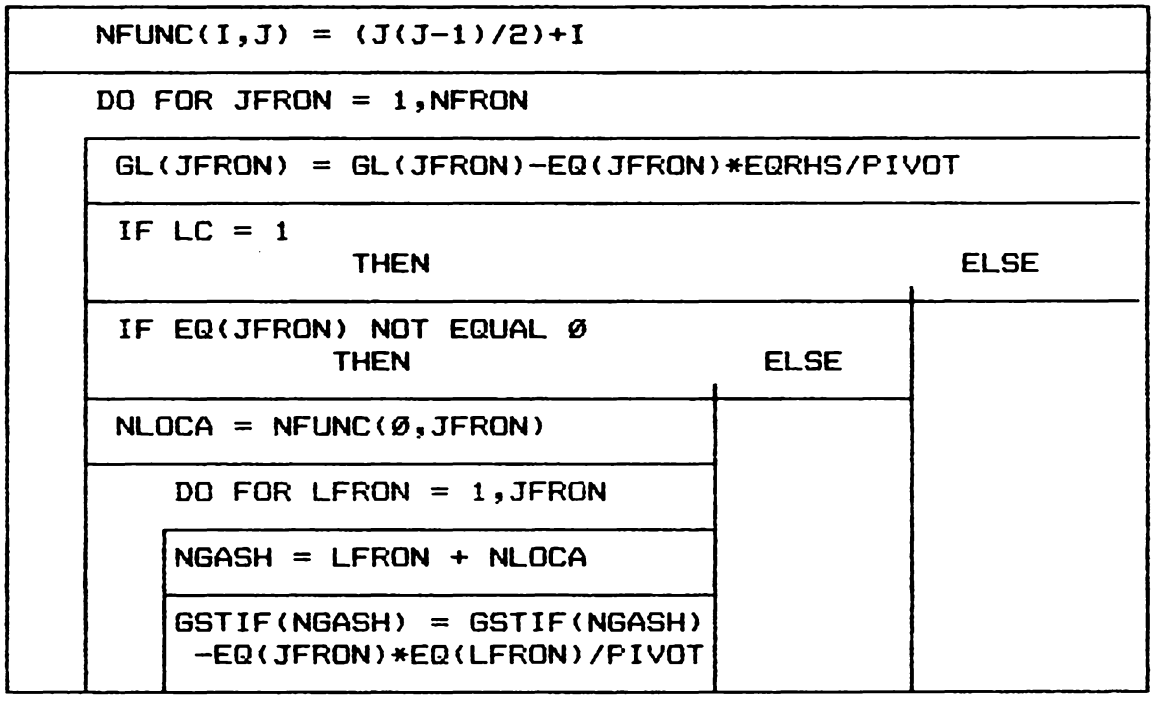
Subroutine REDUCE

Function: Eliminate free variables upon their last appearance.

Input arguments :EQ, EQRHS, GL, GSTIF, NFRON, PIVOT,LC

Output arguments:GL, GSTIF

NS_ diagram



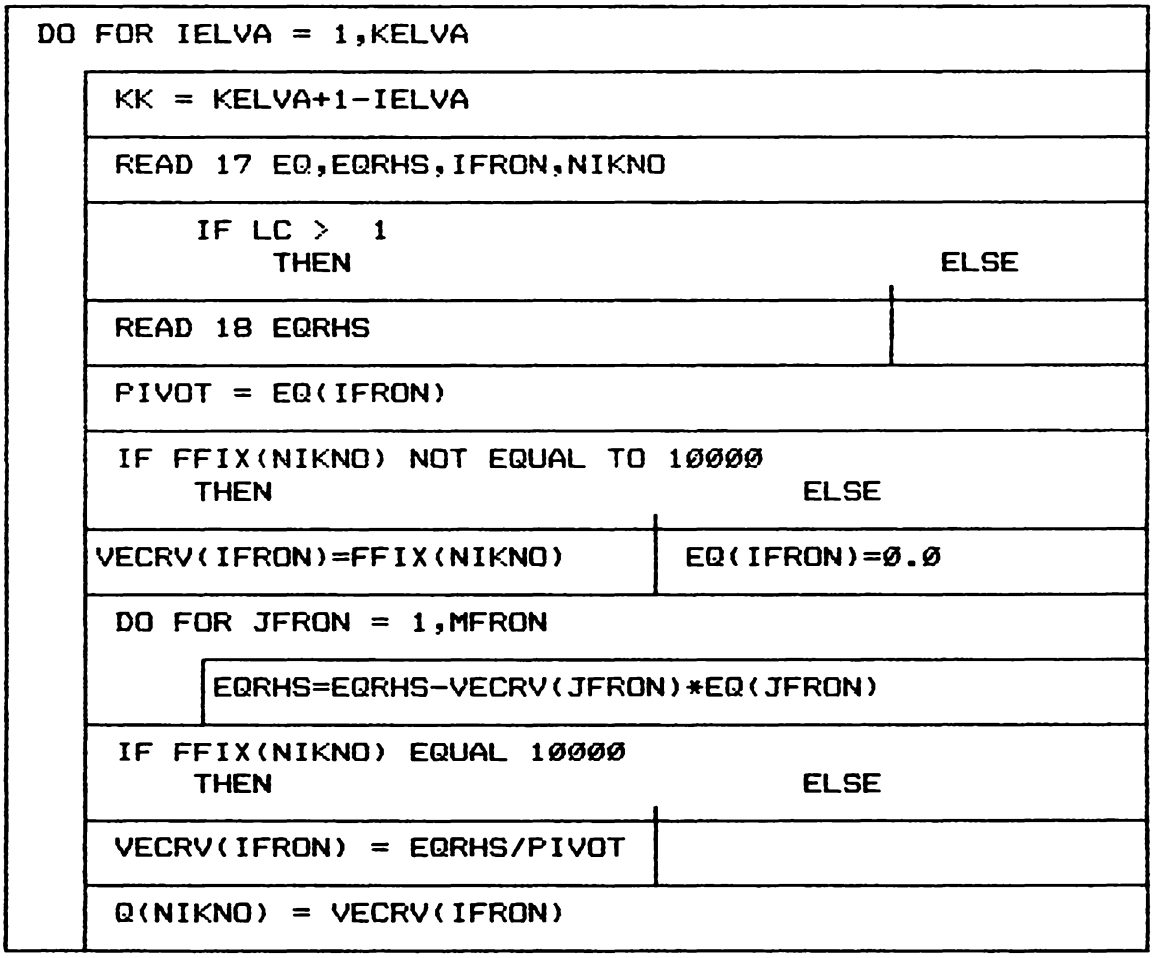
Subroutine BKSUB

Function: Loop over all the variables of the structure, read information from the back-up file, and solve for the joint displacements.

Input arguments :FFIX, KELVA, VECRV, MFRON, Q, LC.

Output arguments:State variables stored in Q.

NS- diagram



CHAPTER 5

BAND VS. FRONTAL

5.1 Introduction

In this chapter a comparison of the band solver and the frontal solver is presented. The comparison consists of presenting the different points of view regarding both methods as cited in the literature, and the results of computations using both a band solver and a frontal solver. The results are compared in respect of out-of-core storage requirements and execution time using the IBM PC-XT (IBM Personal Computer XT).

5.2 Literature Survey

In many structural analysis methods, it is essential to solve a large set of algebraic equations of the form $Ax = B$, where B is a known nodal vector, x is the unknown nodal vector to be computed, and A is a symmetric, positive definite, and sparse coefficient matrix.

The node numbering or the element numbering creates a different distribution of nonzero entries in A depending on the method of solution. In the variable band method the non-zero envelope of A is called the profile, and it is defined by the node numbers connected to each element. In the frontal method, the element numbering defines the maximum frontwidth.

The basic idea of the variable band method is to assemble the whole system stiffness matrix A, partition it

to smaller matrices (blocks) that can be contained in core, and store these blocks in an out-of-core file. To reduce A, the equation solver reads in blocks, reduces them, and writes them to an out-of-core file to be used for in the backsubstitution process. The number of equations that can be solved is no longer limited by the capacity of the central memory, but by the amount of central memory needed for the necessary housekeeping arrays and the available amount of out-of-core storage.

The main steps in solving A x = B by the variable band method may be summarized as:

1. Using the node numbers, determine both the profile of A and the block structure for triangular factors.
2. Compute elements stiffness matrices, load vectors, the numbers of equations associated with each element, and store the results in an out-of-core file.
3. Using previously computed arrays, assemble and block the system stiffness matrix and load vector, store the results in an out-of-core file.
4. Construct triangular factors of A using assembled blocks. Store blocks of factors in an out-of-core file.
5. Using the reduced blocks of A and B perform the resolution. For multiple load conditions only step 5 and those involving the load vector are repeated.

The basic idea of the frontal method is that because the assembly of elements and the elimination of variables

alternate for each element at a time, the system stiffness matrix as a whole is never assembled and contained in core. The core contains only the active part of the system stiffness matrix (the front). Equations to be eliminated may be any where in the front unlike the band method where elimination proceeds in equations order. In the frontal method, once an equation is eliminated it is no longer needed for subsequent triangular decomposition and may be removed from the front.

The basic steps of the frontal method may be summarized as:

1. In the prefront, the order for elimination of variables is determined.
2. Compute element stiffness matrices and load vectors store the results in an out-of-core file.
3. For each element compute the housekeeping vectors retrieve the element stiffness matrix and load vector to be assembled into the front, and eliminate equations whose elements are completely summed at this stage. Store eliminated equations in an out-of-core file.
4. Perform resolution steps using frontal equations in the out-of-core file.

In a comparison conducted in (10) using three-dimensional meshes constructed from 8 node Lagrangian elements, 20 node Serendipity elements, or 27 node Lagrangian elements it is concluded that both methods

require the same number of numerical operations. Based on the advantages in other areas, the band method is preferable for the solution of a broad range of problems in computational mechanics. It is also pointed out that the band method is favored especially in the case of very large problems.

The algorithm of the band method is much simpler to code than the frontal, and in the resolution it requires less indexing and no relocation of information. It is also stated (10) that neither of the two methods is optimal for three-dimensional problems. Either sparse matrix methods or iterative methods may prevail.

Although the frontal method is a particular form of Gauss elimination where the coefficient matrix has to be repeatedly reduced for different load conditions, the frontal algorithm can be coded to reduce the coefficient matrix only once for several load conditions (1).

In a comparison of the band method and the frontal method Irons (3) stated the following:

1 The flexibility inherent in the order in which variables are picked up and eliminated in the frontal method makes it more efficient than the band method.

2. Element numbering is critical and natural in the frontal method, while the node numbering is critical and artificial in the band solver.

3. Variable order is irrelevant in the frontal method. In

fact, variable numbers are some times called nicknames.

4. In the frontal method, data altering is minimal when refining a coarse mesh, While the band method requires an extensive node renumbering to preserve a small band width.
5. The changing of the active variables is achieved in the band method by the bodily shifting of coefficients which takes time, but in the frontal method this is achieved by leaving spaces in the frontal matrix.

In the case of unsymmetric matrices, both methods compare very closely for small problems. However, for medium to large problems the frontal method is faster and requires less core storage (for the same amount of pivotal choice) than the band method (5).

The graph in figure 5.1 (9) indicate that the time saving for frontal solver increases relative to band solver as the problem size increases. It is estimated that at 530 degrees of freedom the time saving is nearly 30% in favor of the frontal solver. However, the three meshes used in this comparison were prepared to produce a minimum band width but not minimum frontwidth.

Because variables are eliminated while the assembly process is in progress, and because of the compact nature of the frontal method, the operations on zero coefficients are minimized and the total arithmetic operations are fewer than with other methods (1).

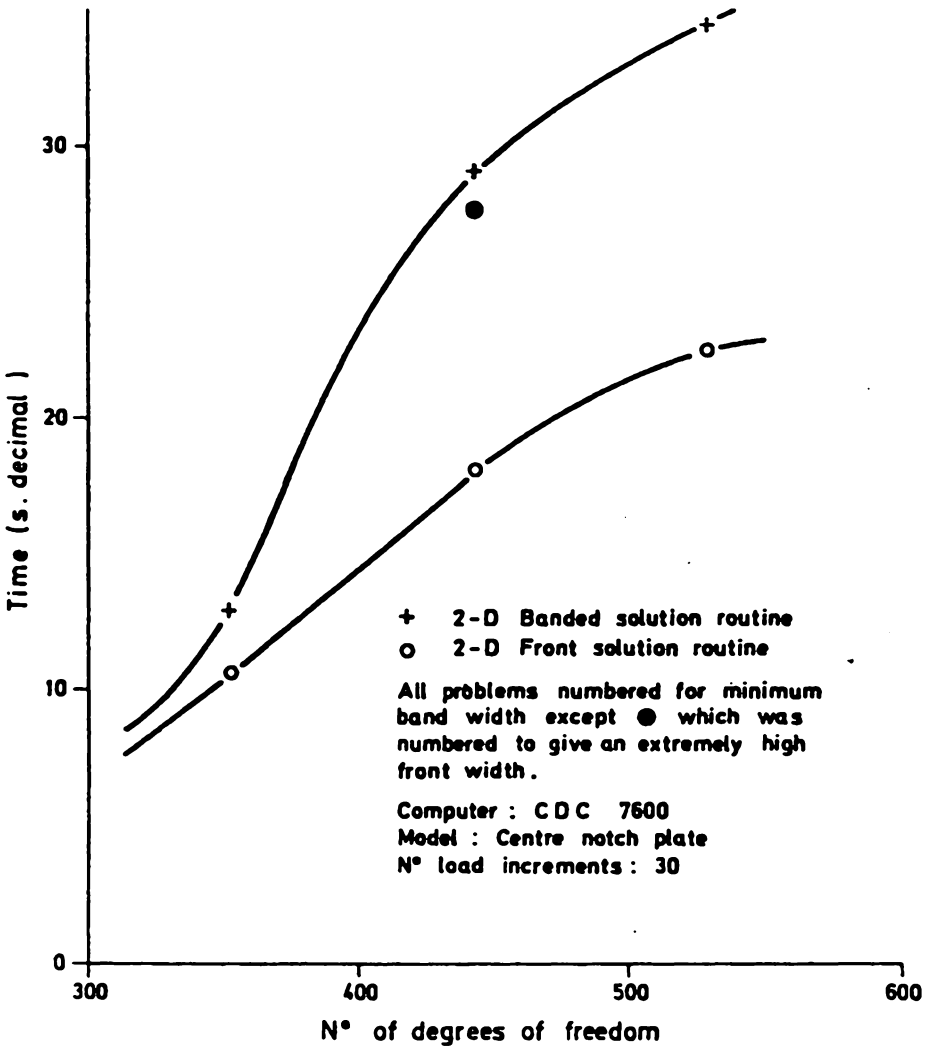


Figure 5.1 Comparison of computation times for elastoplastic programs with banded and front solution routines.

5.3 Comparison of The Two Solvers

This comparison is conducted using two frame analysis programs, one of which was described in chapter 3 which uses frontal subroutines, and the other program was developed using band subroutines (13). The basis for the comparison are the out-of-core storage requirements, the execution time, and the structure size.

The three sets of problems used for comparison were generated by considering three different frames. The number of stories in each frame varies to generate the individual problems in each set. The three frames are depicted in figures 5.2, 5.3, and 5.4. Two load conditions are applied to each frame to explore the capabilities of the two solvers to solve for multiple load conditions. To fairly compare the two solvers and because a band minimizing subroutine is implemented to the band solver, the elements in all three frames are numbered to produce a minimum frontwidth.

The comparison is conducted on the IBM PC-XT with 256 k in-core memory with two types of out-of-core storage. The first is a double sided, double density floppy disk with storage capacity of 354 k bytes, and the second is a ten mega byte hard disk. The comparison results are tabulated in tables 5.4 through 5.9 and analyzed in sections 5.3.1, and 5.3.2.

Tables 5.1, 5.2, and 5.3 contain input data for each problem in the three sets respectively. The same element

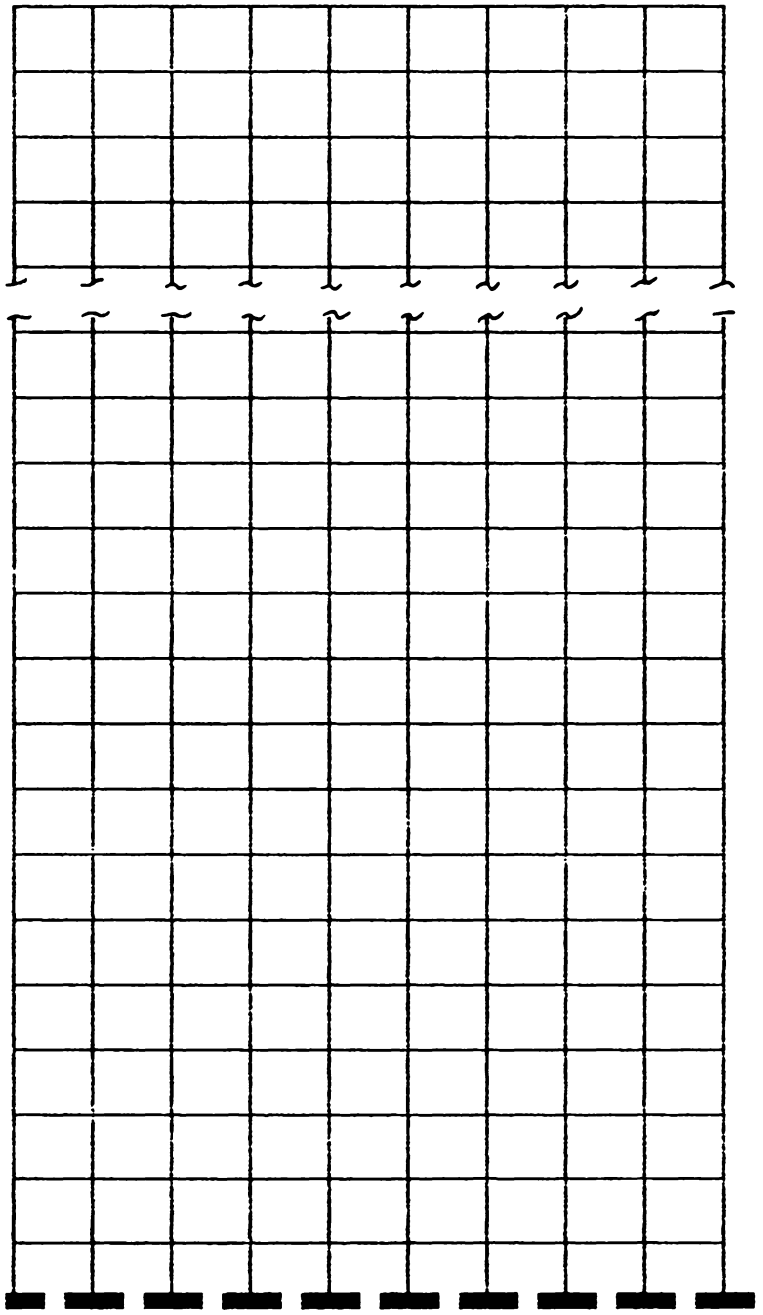


Figure 5.2 Frame used for the first set of problems.

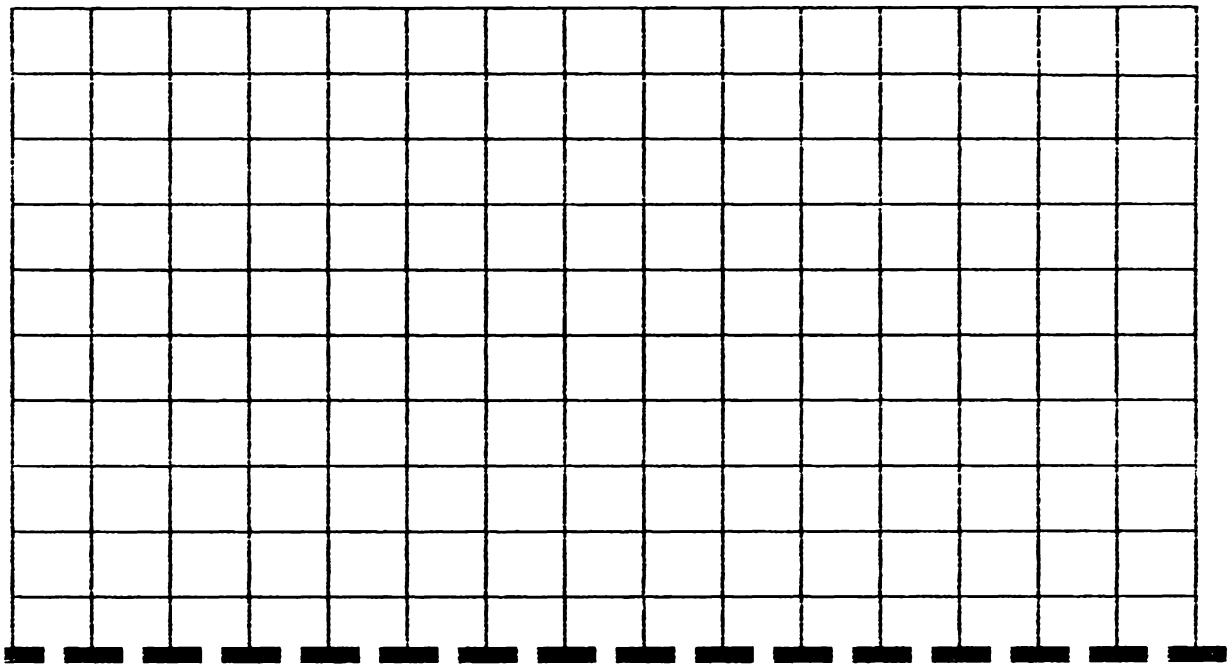


Figure 5.3 Frame used for the second set of problems

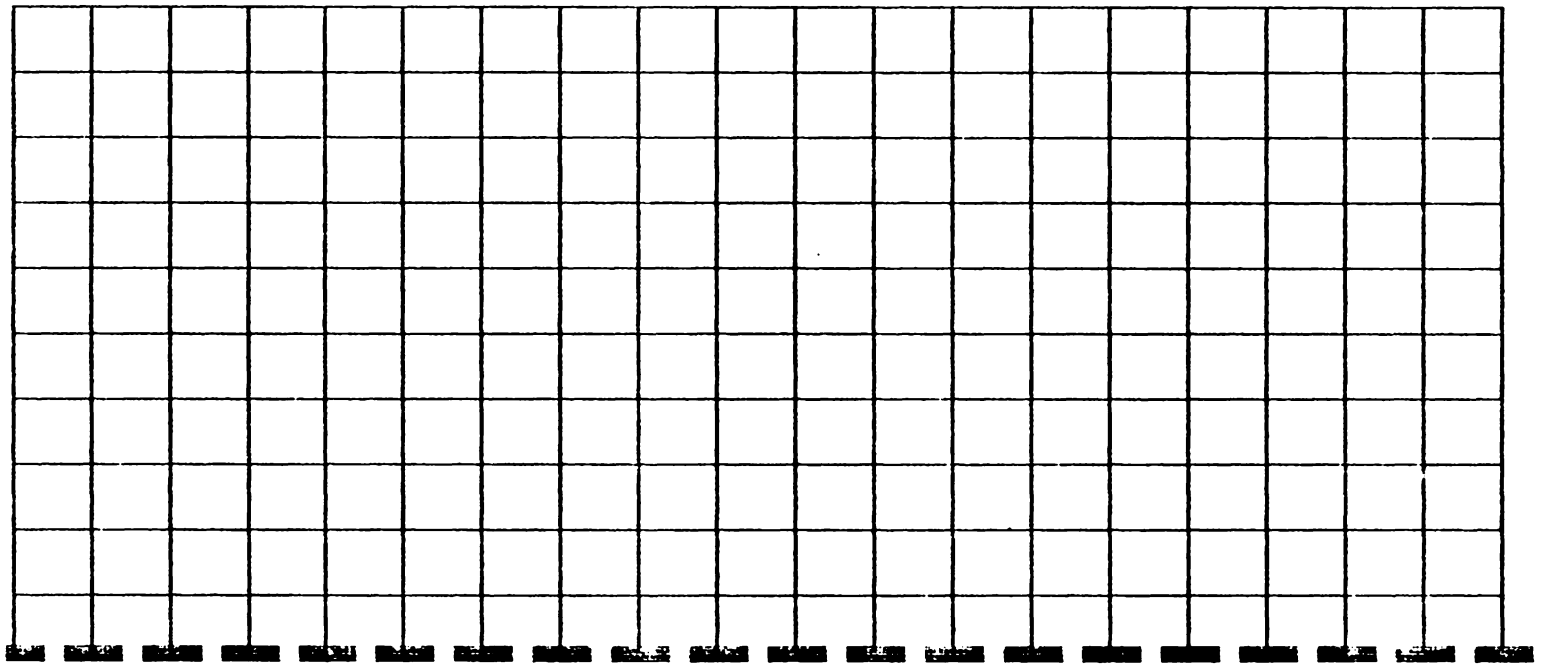


Figure 5.4 Frame used for the third set of problems

Table 5.1 Data for the first set of problems

NO.	NO. ELEMENTS	NO. JOINTS	NO. STORIES	NO. EQUATIONS
1	38	30	2	90
2	95	60	5	180
3	152	90	8	270
4	190	110	10	330
5	228	130	12	390
6	285	160	15	480
7	342	190	18	570
8	380	210	20	630
9	418	230	22	690
10	475	260	25	780
11	570	310	30	930
12	665	360	35	1080
13	760	410	40	1230

Table 5.2 Data for the second set of problems

NO.	NO. ELEMENTS	NO. JOINTS	NO. STORIES	NO. EQUATIONS
1	31	32	1	96
2	124	80	4	240
3	237	128	7	384
4	310	176	10	528

Table 5.3 Data for the third set of problems

NO.	NO. ELEMENTS	NO. JOINTS	NO. STORIES	NO. EQUATIONS
1	39	40	1	120
2	156	100	4	300
3	276	160	7	480
4	390	220	10	660

Table 5.4 Comparison data of the Band and the Frontal solvers using Floppy Disk for the first set of problems with a frontwidth of 33.

NO.	NEQ	TIME (SECONDS) BAND	TIME (SECONDS) FRONTAL	OUT-OF-CORE (K-BYTES) BAND	OUT-OF-CORE (K-BYTES) FRONTAL
1	90	/	144	/	32
2	180	259	271	36	54.5
3	270	481	469	80.5	89.5
4	330	619	580	94	109.5
5	390	770	719	120	129.5
6	480	1000	865	157.5	159
7	570	/	1031	/	188.5
8	630	1371	1134	211.5	208.5
9	690	1526	1270	237	228.5
10	780	/	1451	267	258.5
11	930	/	1870	318	308.5
12	1080	/	2199	/	357.5
13	1230	/	2557	/	408

Table 5.5 Comparison data of the Band and the Frontal solvers using Hard Disk for the first set of problems with a frontwidth of 33.

NO.	NEQ	TIME (SECONDS) BAND	TIME (SECONDS) FRONTAL	OUT-OF-CORE (K-BYTES) BAND	OUT-OF-CORE (K-BYTES) FRONTAL
1	90	/	67	/	32
2	180	142	143	36	54.5
3	270	235	225	80.5	89.5
4	330	280	322	94	109.5
5	390	344	395	120	129.5
6	480	446	495	157.5	159
7	570	/	594	/	188.5
8	630	591	663	211.5	208.5
9	690	/	729	237	228.5
10	780	732	839	267	258.5
11	930	878	1007	318	308.5
12	1080	/	1097	/	357.5
13	1230	/	1257	/	408

Table 5.6 Comparison data of the Band and the Frontal solvers using Floppy Disk for the second set of problems with a frontwidth of 51.

NO.	NEQ	TIME (SECONDS) BAND	FRONTAL	OUT-OF-CORE (K-BYTES) BAND	FRONTAL
1	96	93	128	6	43
2	240	298	482	43	108
3	378	662	814	107.5	174
4	480	704	1296	172	249

Table 5.7 Comparison data of the Band and the Frontal solvers using Hard Disk for the second set of problems with a frontwidth of 51.

NO.	NEQ	TIME (SECONDS) BAND	FRONTAL	OUT-OF-CORE (K-BYTES) BAND	FRONTAL
1	96	77	93	6	43
2	240	171	320	43	108
3	378	321	597	107.5	174
4	480	514	775	172	249

Table 5.8 Comparison data of the Band and the Frontal solvers using Floppy Disk for the third set of problems with a frontwidth of 63.

NO.	NEQ	TIME (SECONDS) BAND	FRONTAL	OUT-OF-CORE (K-BYTES) BAND	FRONTAL
1	120	110	251	10	66.5
2	300	361	776	53	169
3	480	819	1423	136	270
4	660	879	1993	213	373

Table 5.9 Comparison data of the Band and the Frontal solvers using Hard Disk for the third set of problems with a frontwidth of 63.

NO.	NEQ	TIME (SECONDS) BAND	FRONTAL	OUT-OF-CORE (K-BYTES) BAND	FRONTAL
1	120	82	129	10	66.5
2	300	202	498	53	169
3	480	298	945	136	270
4	660	648	1228	213	373

numbering scheme is used for all the problems to produce three easy to predict and different frontwidths. Knowing the frontwidth for each set, the out-of-core storage needed to store all the data files is easy to calculate and it is critical in the case of the floppy disk. However, for the hard disk, it is the size of the in-core storage that is critical and not the out-of-core because it is virtually unlimited.

5.3.1 IBM PC-XT with two floppy disk drives

1. For the set of problems in table 5.1 which has the smallest frontwidth, the two solvers compare very closely in terms of the out-of-core storage requirements. As shown in figure 5.5, the frontal solver is more efficient for the larger problems than the band solver. Consequently, a taller frame may be analyzed by the frontal solver.

2. For the sets of problems in tables 5.2 and 5.3 which have larger frontwidth than the first set, figures 5.6 and 5.7 indicate that the band solver requires less out-of-core storage than the frontal solver. It is also indicated that as the frontwidth increases the frontal solver out-of-core requirements departs from that of the band solver.

3. Figure 5.8 indicates that the frontal solver is faster than the band solver in the case of the first set, while figures 5.9 and 5.10 indicate that the band solver is faster in other two cases. It is also indicated that the time saving increases in favor of the band solver as the

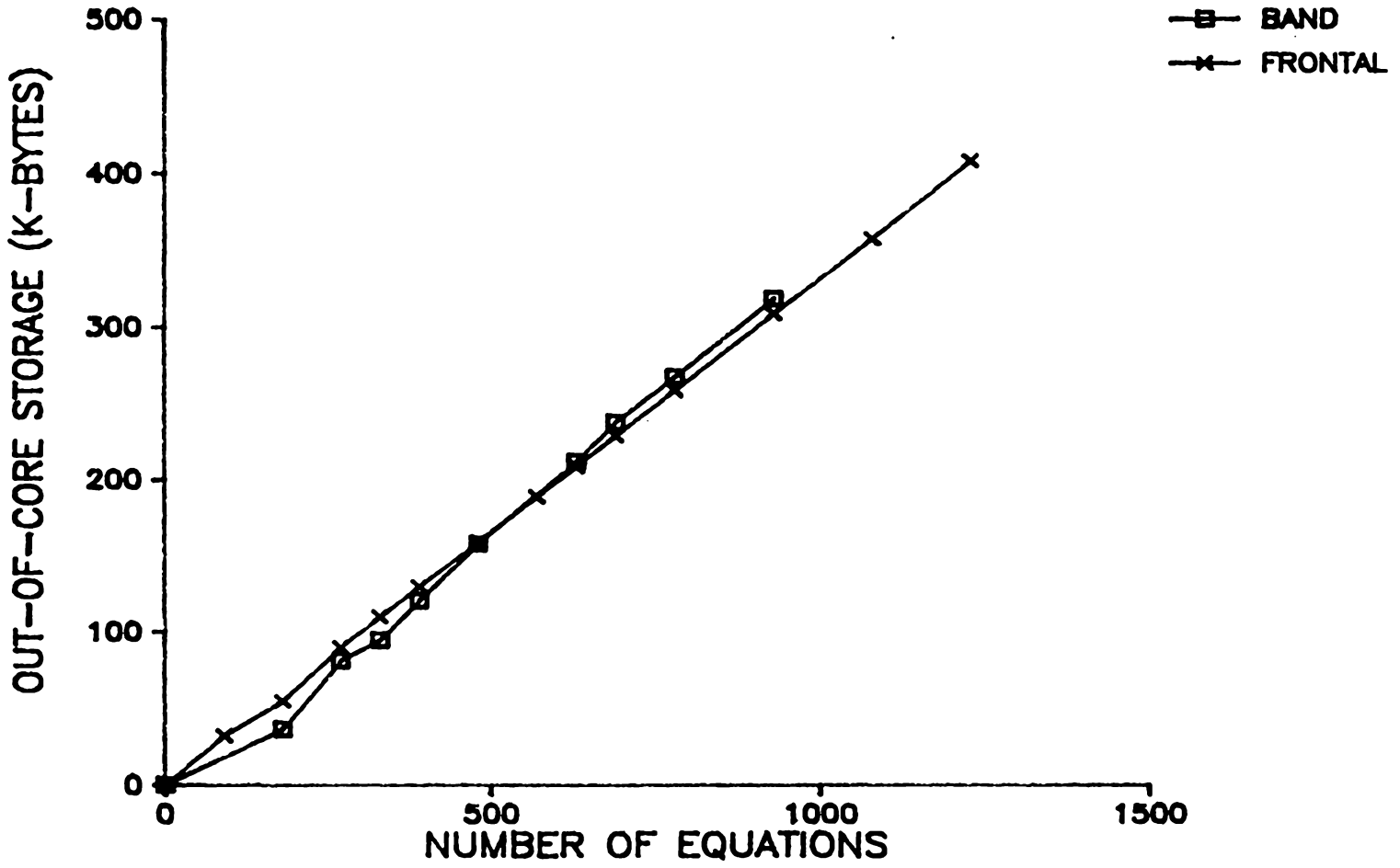


Fig. 5.5 Comparison of Out-of-Core Requirements

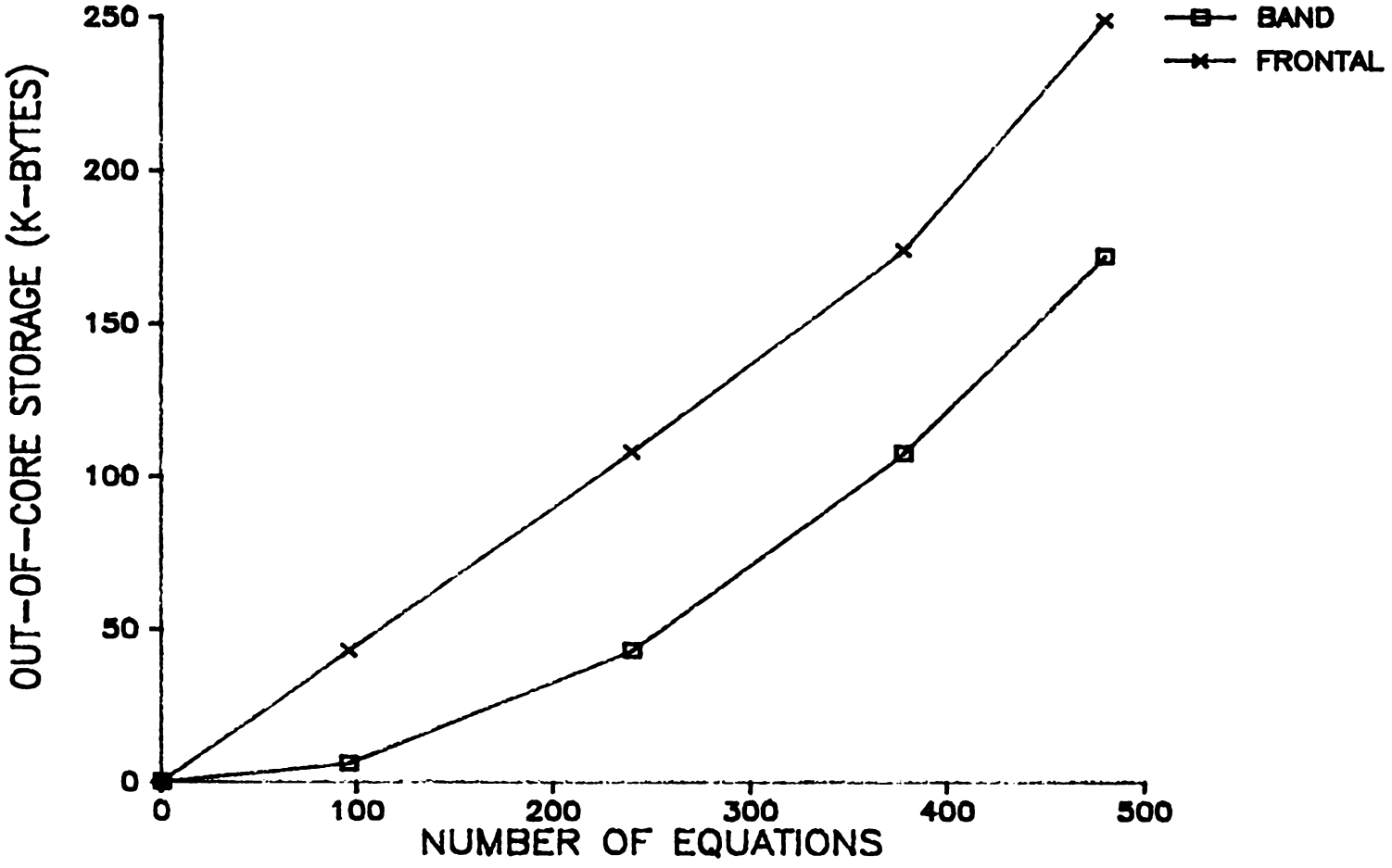


Fig. 5.6 Comparison of Out-of-Core Requirements

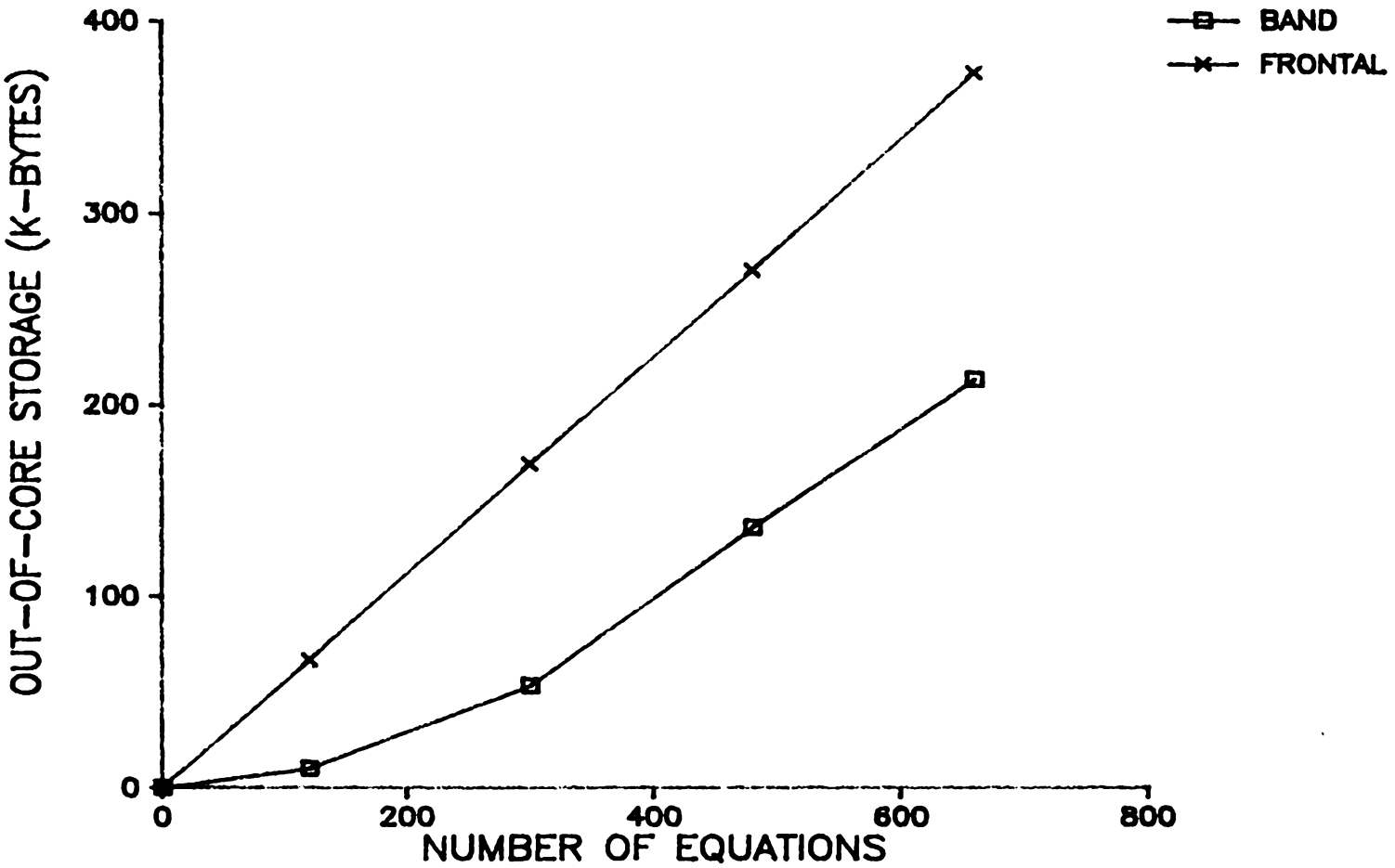


Fig. 5.7 Comparison of Out-of-Core Requirements

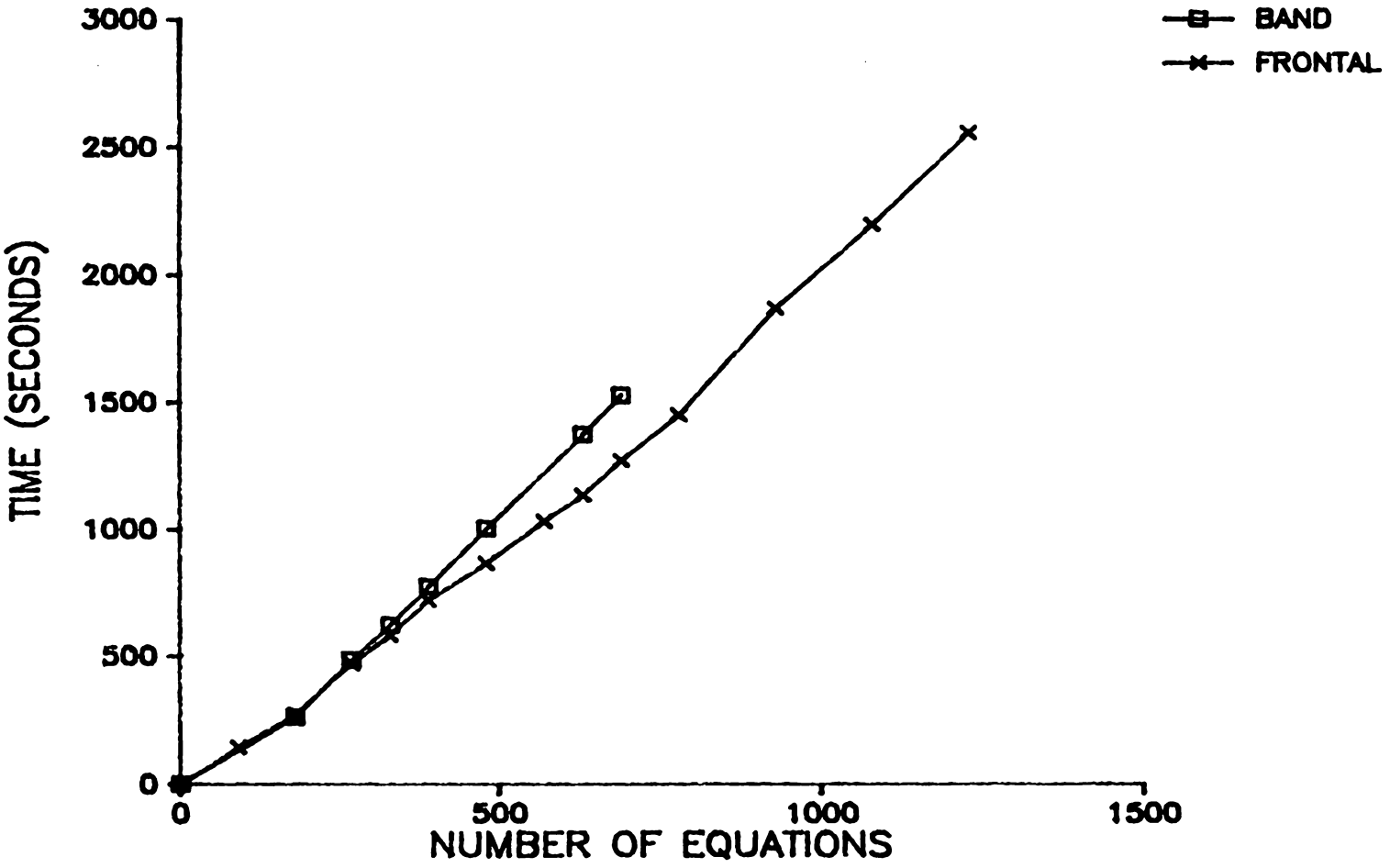


Fig. 5.8 Comparison of of Execution time (Floppy Disk)

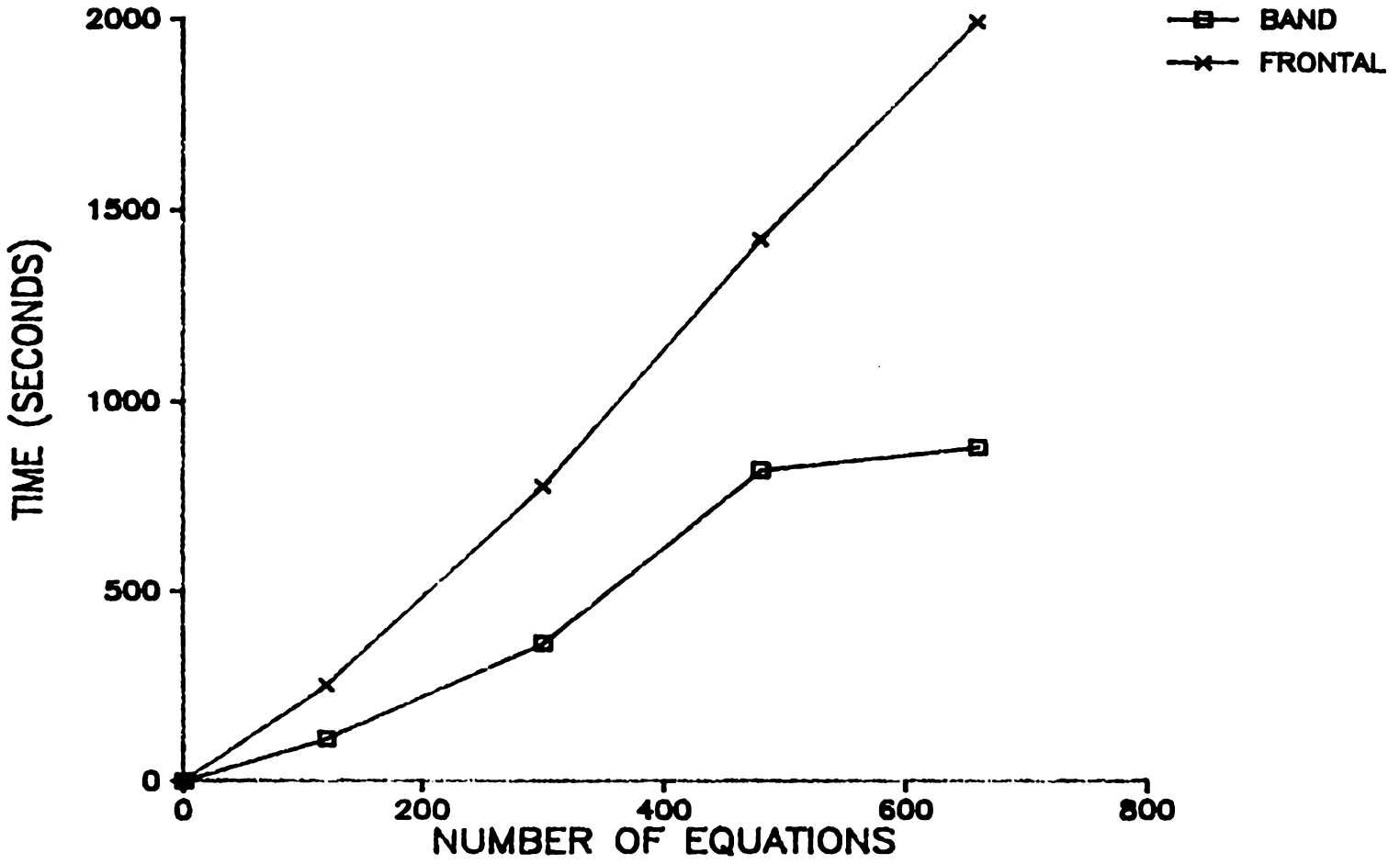


Fig. 5.9 Comparison of of Execution time (Floppy Disk)

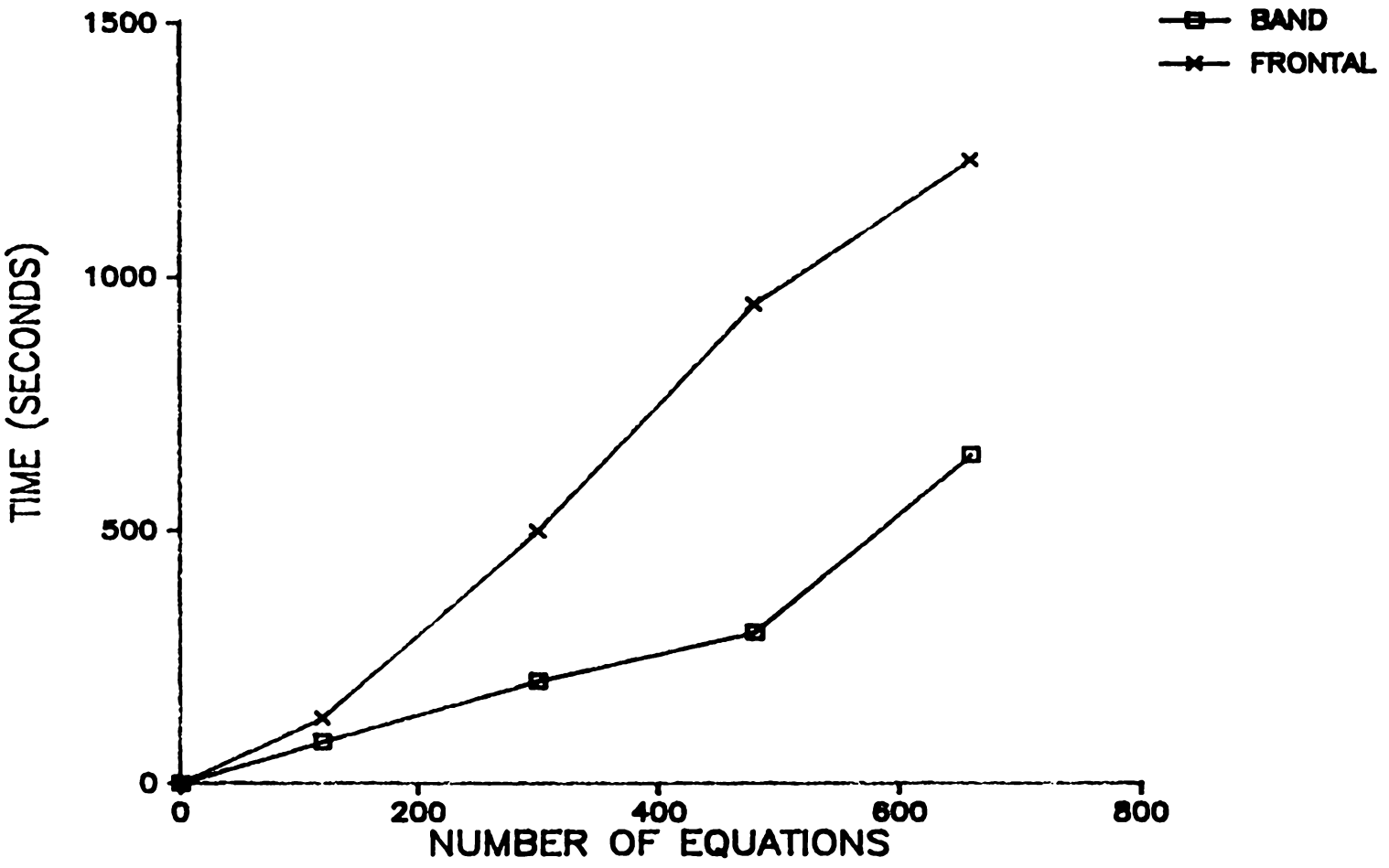


Fig. 5.10 Comparison of of Execution time (Floppy Disk)

frontwidth increases.

5.3.2 IBM PC-XT with a ten mega bytes hard disk drive

1. The same argument for out-of-core storage requirements applies here as in section 5.3.1. However, it should be pointed out that the in-core storage is critical since the out-of-core storage is virtually limitless.

2. Using an IBM PC with a hard disk drive which spins faster than the floppy disk, it is clear from figures 5.11, 5.12, and 5.13 that the band solver is more efficient in terms of execution time. Again the efficiency of the band solver over the frontal solver increases as the frontwidth increase.

5.4 Conclusion

When using the frontal method the user should try to keep the frontwidth to a minimum. This can be done by either using an element numbering subroutine, or simply by inspecting the structure and how the elements are numbered which is rather easy to do for simple frames. PCs with hard disk drives are more efficient than those with double floppy disk drives in terms of execution time and out-of-core storage, which makes it possible to run larger problems. Although it has been cited in several publications (1, 2, 3, 4) that the frontal method is superior to the band method for two and three dimensional elements with more nodes than just the corner nodes, the comparison as conducted in this thesis indicates that the band method is more efficient than

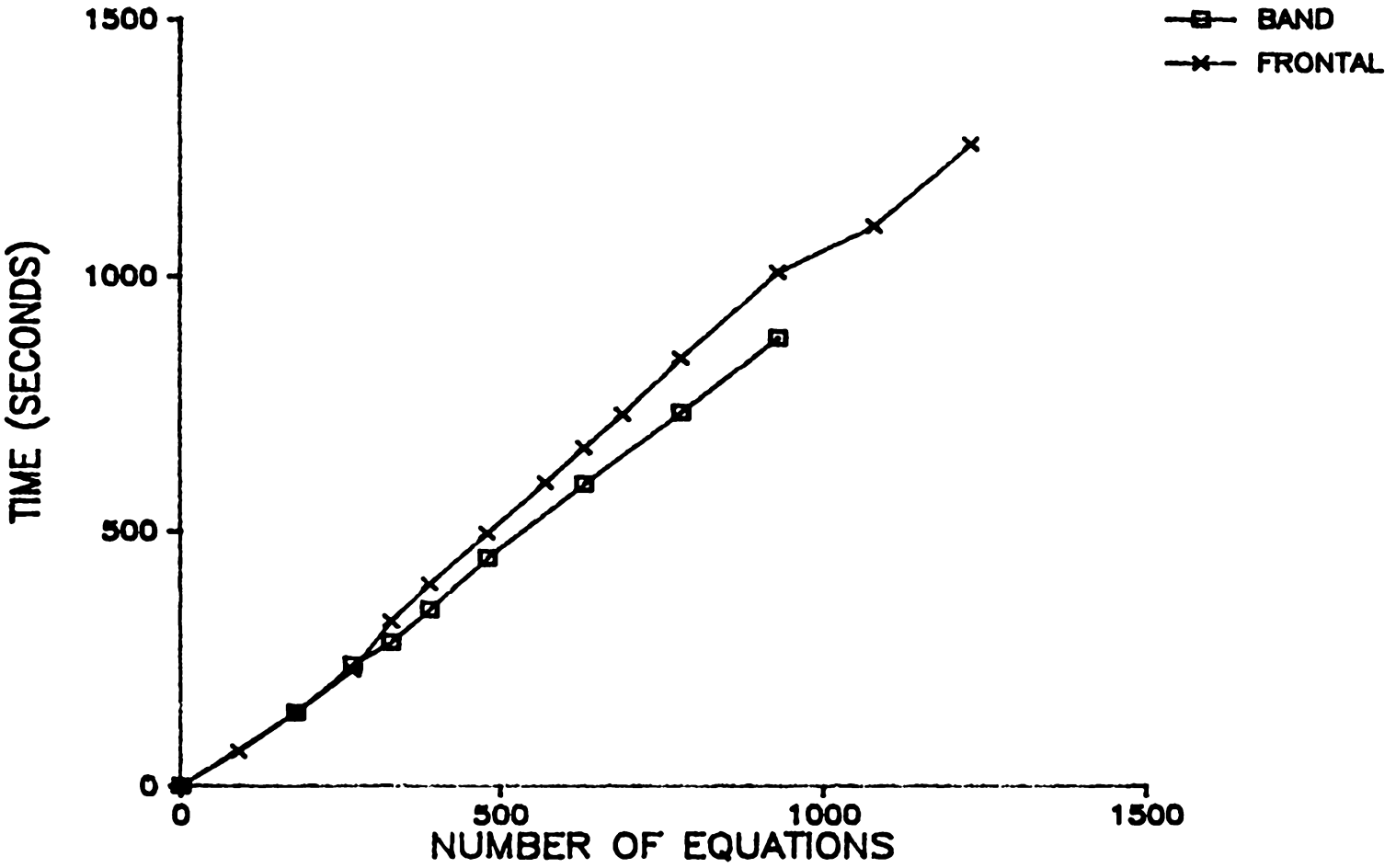


Fig. 5.11 Comparison of of Execution time (Hard Disk)

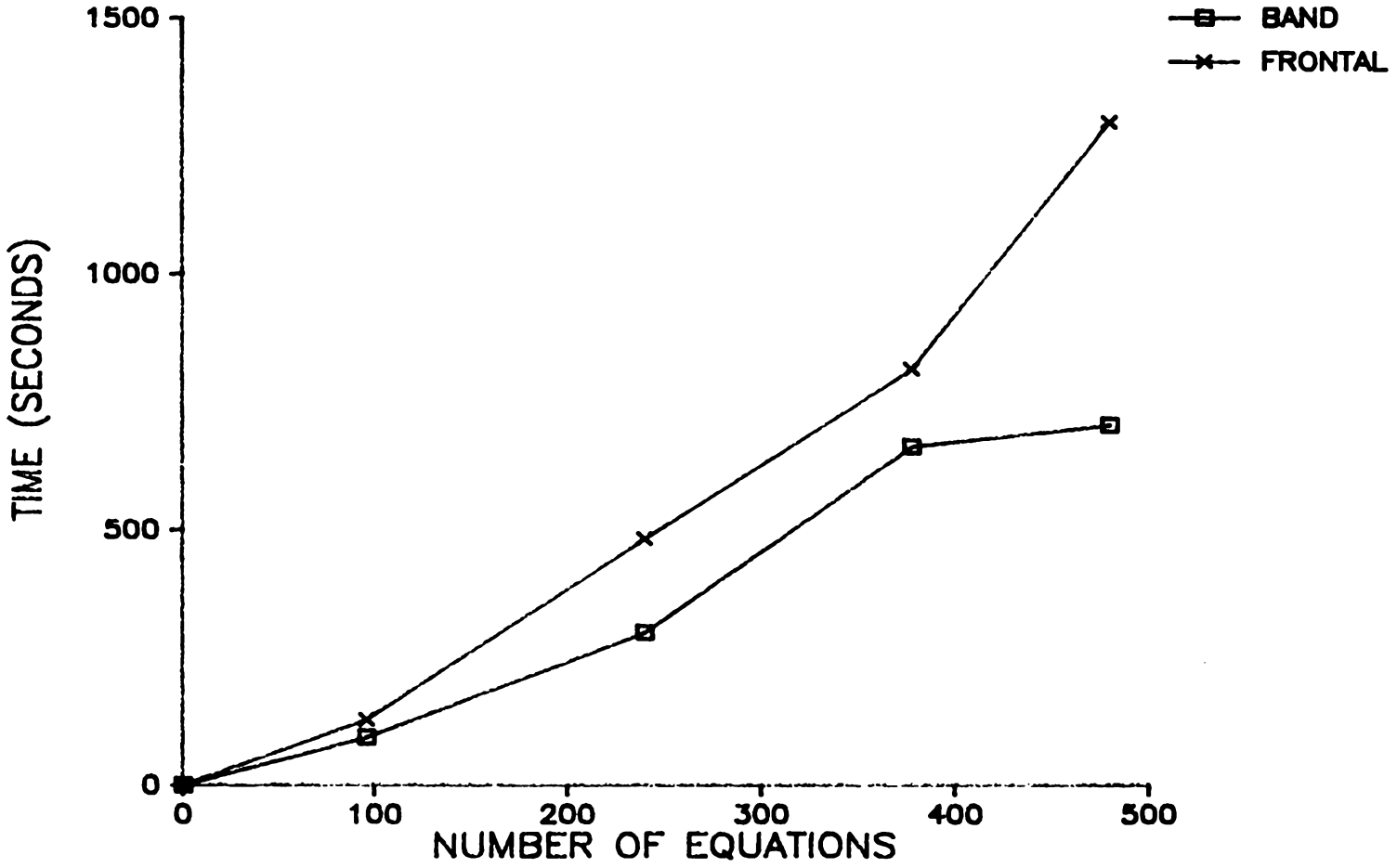


Fig. 5.12 Comparison of of Execution time (Hard Disk)

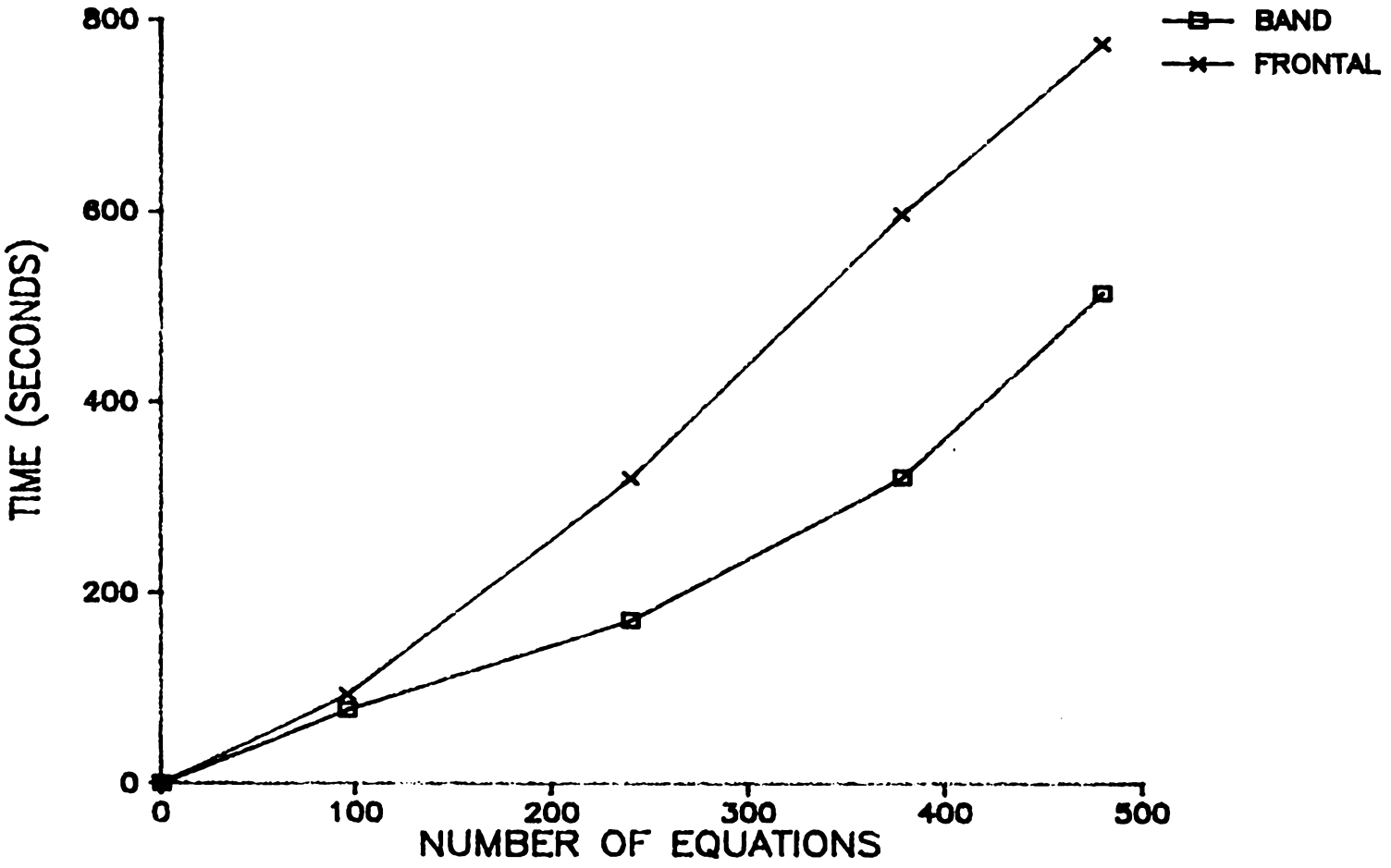


Fig. 5.13 Comparison of of Execution time (Hard Disk)

the frontal method for the analysis of frames.

REFERENCES

1. B. M. Irons, S. Ahmad, Techniques of Finite Elements, Ellis Horwood Ltd, 1980.

2. E. Hinton, D. R. J. Owen, Finite Element Programing, academic Press, 1977.

3. B. M. Irons, A Frontal Solution Program for Finite Element Analysis, Int. J. Meth. Engng, 2, 5-32 (1970).

4. Syed F. Abbas, Some Novel Applications of the Frontal Concept, Int. J. Meth. Engng, 15, 519-536 (1980).

5. P. Hood, Frontal Solution Program For Unsymmetric Matrices, Int. J. Meth. Engng, 10, 379-399 (1976).

6. E. Thompson, Y. Shimazak, A Frontal Procedure Using Skyline Storage, Int. J. Meth. Engng, 15, 889-910 (1980).

7. A. Bykat, A Note On an Element Ordering Scheme, Int. J. Meth. Engng, 11(1), 194-198 (1977).

8. B. M. Irons, E. Elkamshoshy, A Small Frontal Package For Finite Elements, Large Engineering System, Alvin Wexler, Program Press, (1976).

9. M. F. Light, A. R. Luxmore, Application of The Front Solution To Two and Three-Dimensional Elastoplastic Crack Problems, Int. J. Meth. Engng, 11, 393-395, (1977).

10. R. L. Taylor, E. L. Wilson, S. J. Sackett, Direct Solution of Equations by Frontal and Variable Band, Active Column Methods, Nonlinear Finite Element Analysis in Structural Mechanics, W. Wunderlich, E. Stein, K. J. Bathe, Spring-Verlag Berlin Heidelberg Newyork, (1981).

11. S. M. Holzer, Computer Analysis of Strutures, Elsevier, (1985).

12. M. R. Hariri, Post Processor of Plane Frame Analysis, Master's Thesis, Virginia Tech, (1984).

13. R. P. Bowman, Implementation of an Out-of-Core Band Sovler The to Matrix Displacement Method on the IBM PC, Master's Project, Virginia Tech, (1985).

APPENDIX A

PROGRAM LISTING

```
C*****************************************************************
C                        PROGRAM FUNCTION                        *
C*****************************************************************
C  THE PROGRAM PERFORMS THE MATRIX DISPLACEMENT ANALYSIS
C  (FIG. 3.9, REF 13) ØF PLANE FRAMES COMPOSED OF PRISMATIC
C  MEMBERS, WHICH MAY HAVE DISTINCT GEOMETRIC AND MATERIAL
C  PROPERTIES. IT HANDLES MULTIPLE LOAD CONDITIONS. EACH LOAD
C  CONDITION MAY CONSIST OF JOINT PRESCRIBED DISPLACEMENT,
C  JOINT LOADS, AND ANY TYPE OF MEMBER ACTIONS INCLUDING
C  A UNIFORM TEMPERATURE CHANGE (EQ. 4.238, REF.(13)
C*****************************************************************
C*                      INPUT DATA                                *
C*****************************************************************
C      LIST-DIRECTED INPUT
C      INPUT UNITS: KIP, INCH, RADIAN, FAHRENHEIT
C
C      1. ENTER DATE IN THE FORM '06/28/84'(IN MAIN)
C            DATE
C
C      2. ENTER NUMBER OF ELEMENTS, NUMBER OF JOINTS,
C         AND NUMBER OF LOAD CONDITIONS(IN MAIN)
C            NE, NJ, NLC
C
C      3. ENTER MEMBER INCIDENCES(IN STRUCT)
C            MINC(1,I), MINC(2,I)   I=1,NE
C
C      4. ENTER FOR EACH JOINT CONSTRAINT(IN STRUCT)
C            JNUM, JDIR
C            AFTER LAST JOINT CONSTRAINT ENTER
C            Ø, Ø
C
C      5. ENTER EACH JOINT COORDINATES(IN PROP)
C            X(1), X(2)
C
C      6. ENTER MEMBER PROPERTIES(IN PROP)
C            AREA, ZI, EMOD, CTE
C
C      8. DO FOR EACH LOAD CONDITION
C             IF THERE ARE PRESCRIBED JOINT DISPLACEMENT
C             IN MODIFY ENTER FOR EACH PRESCRIBED JOINT
C             DISPLACEMENT JNUM, JDIR, DISP
C                 AFTER THE LAST JOINT PRESCRIBED DISPLACEMENT
C                 ENTER Ø, Ø, Ø
C            ELSE
C                 ENTER Ø, Ø, Ø
C            ENDIF
C           IF THERE ARE JOINT LOADS ENTER(IN JLOAD)
```

```
C              JNUM, JDIR, FORCE
C              AFTER LAST JOINT LOAD ENTER
C              0, 0, 0.
C            ELSE ENTER
C              0, 0, 0.
C            END IF
C
C            IF THERE ARE MEMBER ACTIONS ENTER(IN MACT)
C              MN, MAT, ACT1, DIST1, ACT2, DIST2
C              AFTER LAST MEMBER ACTION ENTER
C              0, 0, 0., 0., 0., 0.
C            ELSE ENTER
C              0, 0, 0., 0.
C            END IF
C           END DO
C*****************************************************************
C*                          MAIN PROGRAM                         *
C*****************************************************************
      IMPLICIT REAL*8 (A-H,O-Z)
      CHARACTER TITLE*20, UNITS*36, DATE*8, FNAME*12
      PARAMETER (MXE = 1000  , MXJ = 500, MXNEQ = 3*MXJ,
     $           MFRON=51    , MSTIF=MFRON*(MFRON+1)/2)
      DIMENSION  MINC(2,MXE) ,Q(MXNEQ)   ,FFIX(MXNEQ)  ,
     $           GSTIF(MSTIF),EQ(MFRON)  ,GL(MFRON)    ,
     $           VECRV(MFRON),NACVA(MFRON)
C
      TITLE = 'PLANE FRAME ANALYSIS'
      UNITS = 'UNITS: KIP, INCH, RADIAN, FAHRENHEIT'
C
C     INITIALIZE PARAMETERS MXE,MXJ,MFRON,AND MXNEQ; READ
C     AND ECHO NE, NJ, NLC;IF NE IS LESS THAN OR EQUAL TO
C     MXE AND NJ  IS ALSO LESS THAN OR EQUAL TO MXJ, CALL
C     FOR EACH LOAD CONDITION DATA, FRONT, AND RESULT;
C     ELSE PRINT ERROR  MESSAGE AND STOP.
      L=8*(MFRON+3)
C
C     OPEN DATA FILES:
      WRITE(*,'(''0'',A )') 'INPUT DATA FILE: '
      READ(*,'(A)') FNAME
      OPEN(5 ,FILE= FNAME)
      OPEN(6 ,FILE= 'FRAME.OUT',STATUS='NEW')
      OPEN(21,FILE= 'JOINT.FIL',STATUS='NEW',ACCESS='DIRECT',
     $RECL=24)
      OPEN(16,FILE=  'PROP.FIL',STATUS='NEW',ACCESS='DIRECT',
     $RECL=56)
      OPEN(17,FILE='B:FILE.ONE',STATUS='NEW',ACCESS='DIRECT',
     $RECL=L)
      OPEN(18,FILE=  'FILE.TWO',STATUS='NEW',ACCESS='DIRECT',
     $RECL=8)
      OPEN(19,FILE= 'FORCE.FIL',STATUS='NEW',ACCESS='DIRECT',
```

```
      $RECL=48)
       READ(5,'(A)') DATE
       WRITE(6,10) TITLE,'(HOLZER,1985)','DATE: ',DATE,UNITS
    10 FORMAT('1',T5,73('*')/T5,'*',T32,A,T77,'*'/T5,'*',T35,
      $  A,T77,'*'/T5,73('*')//T54,A,T62,A//T5,A///)
       READ(5,*) NE, NJ, NLC
       WRITE(6,20) 'NUMBER OF ELEMENTS',NE,'NUMBER OF JOINTS',
      $          NJ, 'NUMBER OF LOAD CONDITIONS',NLC
    20 FORMAT(3(T5,A,T30,I4/))
C
       IF (NE .LE. MXE .AND. NJ .LE. MXJ) THEN
           DO 30 LC = 1, NLC
               CALL DATA  (    FFIX, MINC , NE, NJ, LC    )
               CALL FRONT (Q , FFIX, MINC , NE, NJ, LC    ,
      $                       MFRON, MSTIF, EQ, GL, GSTIF,
      $                       NACVA, VECRV)
               CALL RESULT(Q,         MINC, NE, NJ)
    30     CONTINUE
       ELSE
           WRITE(6,40) 'ERROR MESSAGE: AT LEAST NE OR NJ
      $                EXCEEDS EITHER MXE OR MXJ;',
      $                'INCREASE VALUE OF MX.'
    40     FORMAT(T5, A/ T20, A)
       END IF
       STOP
       END
```

```
C*****************************************************************
C*                              DATA                             *
C*****************************************************************
      SUBROUTINE DATA  (FFIX, MINC, NE, NJ, LC)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION        FFIX(*) ,MINC(2,*)
C
C     FOR THE FIRST LOAD CONDITION, LC=1, CALL STRUCT.
C
      IF(LC.EQ.1) THEN
            CALL STRUCT(FFIX, MINC, NE, NJ)
      END IF
            WRITE(6,10) 'LOAD CONDITION',LC
   10       FORMAT('1'/////T10,A,T25,I2/'+',T10,17('_'))
C
      RETURN
      END
```

```
C*****************************************************************
C*                              STRUCT                           *
C*****************************************************************
      SUBROUTINE STRUCT(FFIX, MINC, NE, NJ)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION        FFIX(*) ,MINC(2,*)
C
C     READ AND ECHO THE MEMBER INCIDENCES, MINC(L,I);
C     INITIALIZE THE FIXITY CODE, FFIX, TO 10000,  READ AND
C     ECHO FOR EACH JOINT CONSTRAINT THE JOINT NUMBER, JNUM,
C     AND JOINT DIRECTION, JDIR, AND STORE A ZERO IN THE
C     CORRESPONDING LOCATION OF FFIX (END OF DATA MARKER
C     JNUM=0); CALL PREFNT,AND PROP.
C
C     WRITE(6,10) 'MEMBER INCIDENCES','MEMBER','A-END','B-END'
C  10 FORMAT(///T10,A/T10,A,T17,A,T23,A)
      DO 30 I=1,NE
          READ(5,*) MINC(1,I),MINC(2,I)
C         WRITE(6,20) I,MINC(1,I),MINC(2,I)
C  20     FORMAT(T11,I3,T18,I3,T24,I3)
   30 CONTINUE
C
      DO 50 J=1,3*NJ
         FFIX(J)= 10000.
   50 CONTINUE
C     WRITE(6,60) 'JOINT CONSTRAINTS','JOINT','DIRECTION'
C  60 FORMAT(///T10,A/T10,A,T17,A)
      READ(5,*) JNUM,JDIR
   70 IF (JNUM.NE.0) THEN
C           WRITE(6,80) JNUM,JDIR
C  80       FORMAT(T11,I3,T19,I3)
            JLOCA=(JNUM-1)*3+JDIR
            FFIX(JLOCA)= 0.0
            READ(5,*) JNUM,JDIR
            GO TO 70
      END IF
      CALL PREFNT(NE, NJ, MINC)
      CALL PROP  (NE, NJ, MINC)
      RETURN
      END
```

```
C*****************************************************************
C*                              PREFRONT                         *
C*****************************************************************
      SUBROUTINE PREFNT(NE ,NJ ,MINC )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION       MINC(2,*)
C
C     IDENTIFY THE LAST APPEARANCE OF A JOINT NUMBER IN THE
C     MEMBER INCIDENCE MATRIX BY ADDING A NEGATIVE SIGN
C     INFRONT OF IT.
C
      DO 30 J=1,NJ
          ILAST= 0
          DO 20 I = 1,NE
             DO 10 N = 1,2
                IF (MINC(N,I).EQ.J) THEN
                    ILAST=I
                    NLAST = N
                ENDIF
  10         CONTINUE
  20      CONTINUE
C
          IF(ILAST.NE.0) THEN
              MINC(NLAST,ILAST) = - MINC(NLAST,ILAST)
          ENDIF
   30 CONTINUE
      RETURN
      END
```

```
C*****************************************************************
C*                              PRØP                             *
C*****************************************************************
      SUBROUTINE PROP (NE, NJ, MINC)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION       MINC(2,*), X(2), Y(2)
C
C     READ AND ECHO JOINT COORDINATES, X(1),X(2); COMPUTE FOR
C     EACH  ELEMENT  BY EQS. C.21  THE LENGTH, ELENG, AND THE
C     DIRECTION  COSINES, C1, C2;  READ  FOR EACH ELEMENT THE
C     CROSS SECTIONAL AREA, THE  MOMENT  OF INERTIA ABOUT THE
C     LOCAL  Z(3)-AXIS, ZI, THE  MODULUS OF ELASTICITY, EMOD,
C     AND THE  COEFFICIENT OF  THERMAL  EXPANSION, CTE; PRINT
C     ELEMENT PROPERTIES.
C
      WRITE(6,10) 'JOINT COORIDINATES','JOINT','DIRECTION-1',
     $            'DIRECTION-2'
   10 FORMAT(///T10,A/T10,A,T17,A,T30,A)
      DO 30 J=1,NJ
         READ(5,*) X(1),X(2)
         WRITE(6,20) J,X(1),X(2)
   20    FORMAT(T13,I3,T20,F8.2,T33,F8.2)
         WRITE(21,REC=J) (X(I),I=1,2)
   30 CONTINUE
C
      WRITE(6,40) 'ELEMENT PROPERTIES','MOMENT OF','ELASTIC',
     $            'THERMAL','ELEMENT','AREA','INERTIA',
     $            'MODULUS','COEFFICIENT','LENGTH'
   40 FORMAT(///T10,A/T35,A,T49,A,T62,A/T10,A,T24,A,T35,A,
     $          T49,A,T62,A,T76,A)
C
      DO 60 I=1,NE
         J=IABS(MINC(1,I))
         K=IABS(MINC(2,I))
         READ(21,REC=J) (X(N),N=1,2)
         READ(21,REC=K) (Y(N),N=1,2)
         EL1=Y(1)-X(1)
         EL2=Y(2)-X(2)
         ELENG=DSQRT(EL1**2+EL2**2)
         C1=EL1/ELENG
         C2=EL2/ELENG
         READ(5,*) AREA,ZI,EMOD,CTE
         WRITE(16,REC=I) AREA,ZI,EMOD,CTE,C1,C2,ELENG
         WRITE(6,50) I,AREA,ZI,EMOD,CTE,ELENG
   50    FORMAT(T12,I3,T19,E12.4,T33,E12.4,T47,E12.4,
     $          T61,E12.4,T75,F8.3)
   60 CONTINUE
      RETURN
      END
```

```
C*****************************************************************
C*                              FRONT                            *
C*****************************************************************
      SUBROUTINE FRONT(Q , FFIX, MINC , NE, NJ, LC    ,
     $                       MFRON, MSTIF, EQ, GL, GSTIF,
     $                       NACVA, VECRV )
      IMPLICIT REAL*8 (A-H,O-Z)
C
      DIMENSION MINC(2,*) ,FFIX(*)    ,Q(*)     ,NACVA(*),
     $          NDEST(6)  ,VECRV(*)   ,EQ(*)    ,GL(*)    ,
     $          GSTIF(*)
C  FUNCTION:
C     SET THE NUMBER OF UNKOWNS IN THE FRONT,NFRON, THE NUMBER
C     OF ELIMINATED VARIABLES,KELVA TO ZERO.
C     INITIALIZES TO ZERO GSTIF(MSTIF), GL(MFRON), EQ(MFRON),
C                        VECRV(MFRON), NACVA(MFRON)
C
      KELVA = 0
      NFRON = 0
      IF (LC.EQ.1) THEN
         DO 10 ISTIF = 1,MSTIF
            GSTIF(ISTIF) = 0.0
  10     CONTINUE
      ENDIF
      DO 15 IFRON=1,MFRON
         EQ(IFRON)=  0.0
         GL(IFRON)=  0.0
         NACVA(IFRON)=0
         VECRV(IFRON)=0.0
  15  CONTINUE
      CALL MODIFY(NJ ,FFIX)
      CALL JLOAD(Q,NJ)
      CALL HOUSE( MINC  ,FFIX ,Q  ,NACVA ,NDEST ,MFRON ,
     $            NFRON ,NE   ,LC ,GSTIF ,GL    ,EQ    ,KELVA)
      CALL BKSUB(FFIX ,KELVA ,VECRV ,MFRON ,Q ,LC, EQ)
      RETURN
      END
```

```
C*****************************************************************
C*                          MODIFY                               *
C*****************************************************************
      SUBROUTINE MODIFY(NJ   ,FFIX)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION             FFIX(*)
C
C     READ  JOINT  NUMBER,  JOINT  DIRECTION, AND DISPLACEMENT
C     GENERATE  THE  FIXITY  ARRAY, FFIX, WHICH  DEFINE  THE
C     PRESCRIBED DEGREES OF FREEDOM BY THEIR PRESCRIBED VALUES
C     AND THE FREE DEGREES OF FREEDOM BY ASSIGNING EACH DEGREE
C     OF FREEDOM THE REAL NUMBER 10000.
C
      DO 10 J = 1,3*NJ
         IF(FFIX(J).NE.10000.0.AND.FFIX(J).NE.0.0) THEN
            FFIX(J) = 10000.
         ENDIF
   10 CONTINUE
      READ(5,*) JNUM,JDIR,DISP
      IF(JNUM.NE.0) THEN
          WRITE(6,60) 'PRESCRIBED JOINT DISPLACEMENT','JOINT',
     $    'DIRECTION','DISPLACEMENT'
   60     FORMAT(///T10,A/T10,A,T17,A,T31,A)
   80     IF(JNUM.NE.0) THEN
              WRITE(6,90) JNUM,JDIR,DISP
   90         FORMAT(T10,I4,T21,I1,T28,F10.3)
              JLOCA =(JNUM-1)*3+JDIR
              FFIX(JLOCA) = DISP
              READ(5,*) JNUM,JDIR,DISP
              GO TO 80
          ENDIF
      ELSE
          WRITE(6,70) 'NO PRESCRIBED JOINT DISPLACEMENT'
   70     FORMAT(///T10,A)
      ENDIF
      RETURN
      END
```

```
C*****************************************************************
C*                            JLOAD                              *
C*****************************************************************
      SUBROUTINE JLOAD( Q, NJ)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION       Q(*)
C  FUNCTION:
C     READ  AND  ECHO  FOR  EACH JOINT LOAD JOINT NUMBER,JNUM,
C     JOINT  DIRECTION, JDIR, AND  FORCE; STORE  FORCE IN Q(K)
C     WHERE K  CORRESPOND TO A SYSTEM DEGREE OF FREEDOM.
C
      DO 15 I = 1,3*NJ
         Q(I) = 0.0
  15  CONTINUE
C
      READ(5,*) JNUM,JDIR,FORCE
      IF(JNUM.NE.0)  THEN
           WRITE(6,10) 'JOINT LOAD (GLOBAL)',
     $                 'JOINT','DIRECTION','FORCE'
  10       FORMAT(///T10,A/,T10,A,T17,A,T31,A)
C
  20       IF(JNUM.NE.0)  THEN
                WRITE(6,30) JNUM,JDIR,FORCE
  30            FORMAT(T11,I3,T21,I3,T28,F10.3)
                K =(JNUM-1)*3+JDIR
                Q(K) = FORCE
                READ(5,*) JNUM,JDIR,FORCE
                GO TO 20
           ENDIF
      ELSE
           WRITE(6,40) 'NO JOINT LOAD'
  40       FORMAT(///T10,A)
      ENDIF
      RETURN
      END
```

```
C*****************************************************************
C*                            HOUSE                              *
C*****************************************************************
      SUBROUTINE HOUSE(MINC  ,FFIX ,Q   ,NACVA ,NDEST ,MFRON ,
     $                 NFRON ,NE   ,LC ,GSTIF ,GL    ,EQ    ,
     $                 KELVA )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION MINC(2,*) ,FFIX(*)    ,ESTIF(6,6),
     $          EQ(*)     ,GL(*)      ,NACVA(*)   ,NDEST(*)   ,
     $          Q(*)      ,GSTIF(*)   ,LOCEL(6)   ,F(6)
C
C  FUNCTION:
C     FOR THE EVERY LOAD CONDITION, GENERATE FOR EACH ELEMENT
C     THREE ARRAYS LOCEL,NACVA,NDEST.THEN CALL ELEMENT LOAD,
C     ELOAD, ESTIF, ASSEMB AND EXAM.
      IACT=0
      DO 30 INE = 1,NE
          NA = 0
          KEVAB = 0
          DO 20 JOINT = 1,2
              LOCNO = MINC(JOINT,INE)
              DO 10 NDOFJ = 1,3
                  NPOSI = (JOINT-1)*3+NDOFJ
                  IF(LOCNO.GT.0)  THEN
                       LOCEL(NPOSI) = (LOCNO-1)*3+NDOFJ
                  ELSE
                       LOCEL(NPOSI) = (LOCNO+1)*3-NDOFJ
                  ENDIF
   10         CONTINUE
   20     CONTINUE
C
C     GENERATE THE ELEMENT ACTIVE VARIABLE ARRAY NACVA(*) AND
C     AN ARRAY OF THIER LOCATIONS NDEST(*)
C
          DO 70 NDOFE = 1,6
              F(NDOFE) = 0.0
              NIKNO = IABS(LOCEL(NDOFE))
              KEXIS = 0
              DO 50 IFRON = 1,NFRON
                  IF(NIKNO.EQ.NACVA(IFRON))  THEN
                       KEVAB = KEVAB+1
                       KEXIS = 1
                       NDEST(KEVAB) = IFRON
                  ENDIF
   50         CONTINUE
C
              IF(KEXIS.EQ.0)  THEN
              IFRON=1
   60             IF(IFRON.LE.MFRON) THEN
```

```
                        IF(NACVA(IFRON).EQ.Ø) THEN
                            NACVA(IFRON) = NIKNO
                            KEVAB = KEVAB+1
                            NDEST(KEVAB) = IFRON
                            IFRON = 1 + MFRON
                        ENDIF
                        IFRON = IFRON + 1
                        GOTO 6Ø
                    ENDIF
                    IF(NDEST(KEVAB).GT.NFRON)  THEN
                        NFRON = NDEST(KEVAB)
                    ENDIF
                ENDIF
7Ø      CONTINUE
        READ(16,REC=INE) AREA,ZI,EMOD,CTE,C1,C2,ELENG
        CALL ELOAD (FFIX, MINC, INE, IACT,   Q,  F, NA   ,
     $                AREA,   ZI,EMOD,  CTE,  C1, C2, ELENG)
        CALL ESTIFF(INE, ESTIF,AREA,   ZI,EMOD,CTE, C1   ,
     $                C2, ELENG)
        CALL ASSEMB(ESTIF ,GSTIF ,NDEST ,INE ,GL ,F ,LC )
        CALL EXAM   ( LOCEL ,NACVA ,MFRON ,NFRON ,EQ   ,
     $                GSTIF ,GL    ,KELVA ,LC    ,FFIX ,INE )
3Ø    CONTINUE
      RETURN
      END
```

```
C*********************************************************************
C*                              ELOAD                                *
C*********************************************************************
      SUBROUTINE ELOAD(FFIX, MINC, INE, IACT,  Q,  F, NA    ,
     $                 AREA, ZI  ,EMOD,  CTE, C1, C2, ELENG )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION   FFIX(*)  ,MINC(2,*),QHAT(6)  ,
     $            Q(*)     ,F(*)     ,QBAR(6)
C
C  FUNCTION:
C     CALL MEMBER JOINT LOAD,MJLD, AND MEMBER ACTION,MACT.
C
      CALL MJLD  (Q   , QBAR, FFIX, MINC,  INE, NA )
      CALL MACT  (F   , QBAR, QHAT,  INE, IACT, NA ,
     $           AREA,   ZI, EMOD,  CTE,   C1, C2 , ELENG )
      CALL ASSEMF( F , QHAT, QBAR,  INE,   C1, C2 , NA)
      RETURN
      END
```

```
C*****************************************************************
C*                      MEMBER JOINTS LOADS                      *
C*****************************************************************
      SUBROUTINE MJLD(Q    ,QBAR   ,FFIX    ,MINC,INE ,NA)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION       Q(*) ,QBAR(*),FFIX(*) ,MINC(2,*)
C
C   FUNCTION:
C     TRANSFORM APPLIED JOINT LOADS FROM GLOBAL JOINT LOADS
C     TO LOCAL ELEMENT JOINT LOADS.
      DO 5 I = 1,6
          QBAR(I) = 0.0
  5   CONTINUE
      DO 20 JOINT = 1,2
          JNUM = IABS(MINC(JOINT,INE))
          DO 10 NDOFJ = 1,3
  22            K    = (JNUM-1)*3+NDOFJ
              IF(FFIX(K).EQ.10000.) THEN
                  IF(JOINT.EQ.1)  THEN
                      NDOFE = NDOFJ
                  ELSE
                      NDOFE = NDOFJ +3
                  ENDIF
                  IF(Q(K).NE.0.0) THEN
                     QBAR(NDOFE) = Q(K)
                     NA = NA +1
                     Q(K) =0.0
                  ENDIF
              ENDIF
  10      CONTINUE
  20  CONTINUE
      RETURN
      END
```

```
C*****************************************************************
C*                              MACT                             *
C*****************************************************************
      SUBROUTINE MACT(F, QBAR, QHAT, INE,IACT, NA,
     $ _           AREA,   ZI, EMOD, CTE,  C1, C2,ELENG )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION   F(*)   ,QBAR(*)   ,QHAT(*)
C
C     READ THE MEMBER NUMBER, MN, THE MEMBER ACTION TYPE, MAT,
C     THE ACTIONS, P1 AND P2, THE DISTANCES ,X1 AND X2.
C     PRINT MN, MAT, P1 ,X1 ,P2 ,X2
C     INCREMENT THE ACTION COUNTER,
C     COMPUTE THE FIXED-END FORCES.
      DO 1Ø I = 1,6
         QHAT(I) = Ø.Ø
  1Ø  CONTINUE
      IF(IACT.LT.1ØØØ)THEN
         READ(5,*)  MN, MAT, X1 ,P1 ,X2 ,P2
  15     IF(MN.NE.Ø) THEN
           IF(IACT.EQ.Ø) THEN
           WRITE(6,2Ø)'MEMBER ACTIONS','MEMBER','TYPE','ACT-ONE',
     $                'DIST-ONE', 'ACT-TWO','DIST-TWO'
  2Ø       FORMAT(///T1Ø,A/T1Ø,A,T18,A,T26,A,T37,A,T5Ø,A,T62,A)
           IACT = IACT+1
         ENDIF
         IF (MN.EQ.INE) THEN
              WRITE(6,3Ø) MN,MAT,P1 ,X1 ,P2,X2
  3Ø          FORMAT(T11,I3,T19,I1,T22,F1Ø.3,T36,F8.3,T46,
     $              F1Ø.3,T61,F8.3)
              NA = NA + 1
C
              IF (MAT.EQ.1) THEN
C                CONCENTRATED LOAD
                 EL=ELENG
                 A=X1 / EL
                 QHAT(2) =QHAT(2) -P1*(1.Ø+A**2*(2.Ø*A-3.Ø))
                 QHAT(3) =QHAT(3) -P1*EL*A*(1.Ø-A)**2
                 QHAT(5) =QHAT(5) +P1*A**2*(2.Ø*A-3.Ø)
                 QHAT(6) =QHAT(6) +P1*EL*A**2*(1.Ø-A)
              ELSE IF (MAT.EQ.2) THEN
C                CONCENTRATED AXIAL LOAD
                 EL=ELENG
                 A=X1 / EL
                 QHAT(1)=QHAT(1)-P1*(1.Ø-A)
                 QHAT(4)=QHAT(4)-P1*A
              ELSE IF (MAT.EQ.3) THEN
C                LINEARLY VARYING DISTRIBUTED LOAD OVER
C                A PORTION OF THE ELEMENT
                 EL=ELENG
```

```
                A   = X1 / EL
                B   = X2 / EL
                D1  = (P1*EL/60.)*(A-B)
                D2  = (P2*EL/60.)*(B-A)
              QHAT(2     ) =QHAT(2     )+D2*3.*(B**2*(15.-8.*B)
     $                     +(A*B)*(10.-6.*B)+(A**2)*(5.-4.*B)
     $                     -2.*(A**3)-10.)-D1*3.*(A**2*(15.-8.
     $                     *A)+(A*B)*(10.-6.*A)+(B**2)*(5.-4.
     $                     *A)-(2.*B**3)-10.)
              QHAT(3     ) =QHAT(3     )+D2*EL*(B**2*(30.-12.*B)
     $                     +(A*B)*(20.-9.*B)+A**2*(10.-6.*B)-3.
     $                     *A**3-10.*(2.*B+A))-D1*EL*(A**2*(30.
     $                     -12.*A)+(A*B)*(20.-9.*A)+(B**2)*(10.
     $                     -6.*A)-(3.*B**3)-10.*(2.*A+B))
              QHAT(5     ) =QHAT(5     ) -D2*3.*(B**2*(15.-8.*B)
     $                     +(A*B)*(10.-6.*B)+(A**2)*(5.-4.*B)-
     $                      2.*(A**3))+D1*3.*(A**2*(15.-8.*A)+
     $                      (A*B)*(10.-6.*A)+(B**2)*(5.-4.*A)-
     $                      (2.*B**3))
              QHAT(6     ) =QHAT(6     )-D2*EL*(B**2*(12.*B-15.)
     $                     +(A*B)*(9.*B-10.)+(A**2)*(6.*B-5.)+
     $                     3.*A**3)+D1*EL*(A**2*(12.*A-15.)+
     $                     (A*B)*(9.*A-10.)+(B**2)*(6.*A-5.)+
     $                     (3.*B**3))
               ELSE IF(MAT.EQ.4) THEN
C               LINEARLY VARYING DISTRIBUTED AXIAL LOAD OVER
C               A PORTION OF THE ELEMENT.
              EL = ELENG
              QHAT(1) =QHAT(1) -P1*(X2-X1)*(1.-(X1+X2)/(2*EL)
     $              )-(P2-P1)*((X2-X1)/2)*(1.-(X1+2.*X2)/3.*EL)
              QHAT(4) =QHAT(4) -P1*(X2-X1)*((X1+X2)/(2*EL))
     $                    -(P2-P1)*((X2-X1)/2)*((X1+2.*X2)/3.*EL)
               ELSE
C                  TEMPRETURE CHANGE
                   P=AREA*EMOD*CTE*ACT
                   QHAT(1)=QHAT(1) +P
                   QHAT(4)=QHAT(4) -P
               ENDIF
              READ(5,*)  MN, MAT, X1 ,P1 ,X2 ,P2
                   GO TO 15
          ELSE
                   BACKSPACE 5
          ENDIF
       ELSE
          IF (IACT.EQ.0) THEN
               WRITE(6,60)' NO MEMBER ACTIONS'
    60         FORMAT(///T10,A)
               IACT = IACT+1000
          ELSE
               IACT= IACT+1000
```

```
   ENDIF
ENDIF
ENDIF
WRITE(19,REC = INE) (QHAT(I),I=1,6)
RETURN
END
```

```
C*****************************************************************
C*                              ASSEMF                           *
C*****************************************************************
      SUBROUTINE ASSEMF(F   ,QHAT   ,QBAR ,INE ,C1     ,C2    ,NA)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION  F(*) ,QHAT(*) ,QBAR(*)
C
C     TRANSFORM AND ASSEMBLE THE LOCAL FIXED-END FORCES,QHAT(6)
C     AND APLIED  MEMBER  JOINT  LOADS  QBAR(*)  TO PRODUCE THE
C     EQUIVALENT ELEMENT JOINT LOAD VECTOR, F(*)
C     STATEMENT FUNCTION
      FG(C1I,C2I,FLX,FLY) = C1I * FLX + C2I * FLY
      IF (NA .NE. Ø) THEN
          DO 1Ø NDOFE = 1, 6
             IF(NDOFE.EQ.1)  THEN
                   F(NDOFE) = QBAR(NDOFE)-FG(C1,
     $                        -C2,QHAT(1),QHAT(2))
             ELSE IF(NDOFE.EQ.2)  THEN
                   F(NDOFE) = QBAR(NDOFE)-FG(C2,
     $                         C1,QHAT(1),QHAT(2))
             ELSE IF(NDOFE.EQ.3)  THEN
                   F(NDOFE) =QBAR(NDOFE)-QHAT(3)
             ELSE IF(NDOFE.EQ.4)  THEN
                   F(NDOFE) = QBAR(NDOFE)-FG(C1,
     $                         -C2,QHAT(4),QHAT(5))
             ELSE IF(NDOFE.EQ.5)  THEN
                   F(NDOFE) = QBAR(NDOFE)-FG(C2,
     $                         C1,QHAT(4),QHAT(5))
             ELSE IF(NDOFE.EQ.6)  THEN
                   F(NDOFE) = QBAR(NDOFE)-QHAT(6)
             ENDIF
   1Ø     CONTINUE
      ENDIF
      RETURN
      END
```

```
C*****************************************************************
C*                              ESTIFF                           *
C*****************************************************************
      SUBROUTINE ESTIFF (INE, ESTIF, AREA, ZI    ,EMOD,
     $                   CTE,    C1,   C2, ELENG)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION    G(7) ,ESTIF(6,6) ,INDEX(6,6)
      DATA INDEX / 1,2,4,-1,-2,4,     2,3,5,-2,-3,5,
     $             4,5,6,-4,-5,7,    -1,-2,-4,1,2,-4,
     $            -2,-3,-5,2,3,-5,    4,5,7,-4,-5,6/
C
C  FUNCTION:
C     FOR ELEMENT INE,COMPUTE THE GLOBAL STIFFNESS
C     COEFFICIENTS,G(7) AND ASSEMBLE ESTIF(6,6)
C
      ALFA = EMOD*ZI/ELENG**3
      BETA = AREA*ELENG**2/ZI
      G(1)= ALFA*(BETA*C1**2+12.0*C2**2)
      G(2)= ALFA*C1*C2*(BETA-12.0)
      G(3)= ALFA*(BETA*C2**2+12.0*C1**2)
      G(4)=-ALFA*6.0*ELENG*C2
      G(5)= ALFA*6.0*ELENG*C1
      G(6)= ALFA*4.0*ELENG**2
      G(7)= ALFA*2.0*ELENG**2
C
      DO 20 I = 1,6
         DO 10 J = 1,6
            L = INDEX(I,J)
            IF(L.GT.0)  THEN
               ESTIF(I,J) = G(L)
            ELSE
               ESTIF(I,J) = -G(-L)
            ENDIF
   10    CONTINUE
   20 CONTINUE
      RETURN
      END
```

```
C*****************************************************************
C*                          ASSEMBLE                             *
C*****************************************************************
      SUBROUTINE ASSEMB( ESTIF ,GSTIF ,NDEST ,INE ,GL  ,F ,LC)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION  ESTIF(6,6) ,GSTIF(*) ,NDEST(*) ,GL(*) ,F(*)
      NFUNC(I,J) = (J*J-J)/2+I
C
C  FUNCTION:
C     ASSEBLE ELEMENT LOADS,F, INTO A GLOBAL LOAD VECTOR,GL,
C     AND ELEMENT STIFFNESS MATRIX INTO A GLOBAL STIFFNESS
C     ARRAY,GSTIF.
      DO 30 IDOFE = 1,6
          IDEST = NDEST(IDOFE)
          GL(IDEST) = GL(IDEST) + F(IDOFE)
          IF(LC.EQ.1)  THEN
              DO 40 JDOFE = 1,6
                 JDEST = NDEST(JDOFE)
                 NGASH = NFUNC(IDEST,JDEST)
                 IF(JDEST.GE.IDEST)  THEN
                     GSTIF(NGASH) = GSTIF(NGASH) +
     $                                ESTIF(IDOFE,JDOFE)
                 ENDIF
                 ENDIF
  40          CONTINUE
          ENDIF
  30  CONTINUE
      RETURN
      END
```

```
C*****************************************************************
C*                          EXAM                                 *
C*****************************************************************
      SUBROUTINE EXAM( LOCEL ,NACVA ,MFRON ,NFRON ,EQ ,GSTIF,
     $                 GL    ,KELVA ,LC    ,FFIX  ,INE )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION      LOCEL(*)   ,NACVA(*)   ,EQ(*) ,
     $               GSTIF(*)   ,GL(*)      ,FFIX(*)
C
      NFUNC(I,J) = (J*J-J)/2+I
      DO 40 IDOFE = 1,6
      NIKNO = -LOCEL(IDOFE)
      IF(NIKNO.GT.0)  THEN
         IFRON = 1
   30    IF(IFRON.LE.NFRON) THEN
            IF(NACVA(IFRON).EQ.NIKNO)  THEN
               IF(LC.EQ.1)  THEN
                  DO 10 JFRON = 1,MFRON
                     IF(IFRON.LT.JFRON)  THEN
                        NLOCA = NFUNC(IFRON,JFRON)
                     ELSE
                        NLOCA = NFUNC(JFRON,IFRON)
                     ENDIF
                     EQ(JFRON) = GSTIF(NLOCA)
                     GSTIF(NLOCA) = 0.0
   10             CONTINUE
               ENDIF
               EQRHS = GL(IFRON)
               GL(IFRON) = 0.0
               KELVA = KELVA+1
               IF(LC.EQ.1)  THEN
                  WRITE(17,REC=KELVA) (EQ(I),I=1,MFRON),EQRHS,
     $                                 IFRON,NIKNO
               ELSE
                  WRITE(18,REC=KELVA) EQRHS
                  READ(17,REC=KELVA) (EQ(I),I=1,MFRON),DUMMY,
     $                                IDUM,NIKNO
               ENDIF
               PIVOT = EQ(IFRON)
               EQ(IFRON) = 0.0
C
               IF(FFIX(NIKNO).NE.10000.)  THEN
                  IF(FFIX(NIKNO).NE.00000.)  THEN
                     DO 20 JFRON = 1,NFRON
                       GL(JFRON) = GL(JFRON)-FFIX(NIKNO)*
     $                                       EQ(JFRON)
   20                CONTINUE
                  ENDIF
               ELSE
```

```
                  CALL REDUCE( EQ      ,EQRHS   ,GL   ,GSTIF ,
     $                         NFRON   ,PIVOT   ,LC   ,INE )
               ENDIF
               EQ(IFRON) = PIVOT
               NACVA(IFRON) = 0
               IFRON=NFRON+1
            ENDIF
            IFRON=IFRON+1
            GOTO 30
         ENDIF
35       IF(NACVA(NFRON).EQ.0)  THEN
            NFRON = NFRON-1
            IF(NFRON.GT.0)  THEN
               GOTO 35
            ENDIF
         ENDIF
      ENDIF
40    CONTINUE
      RETURN
      END
```

```
C*****************************************************************
C*                          REDUCE                               *
C*****************************************************************
      SUBROUTINE REDUCE( EQ      ,EQRHS   ,GL   ,GSTIF ,
     $                   NFRON   ,PIVOT   ,LC   ,INE )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION       EQ(*)    ,GL(*)    ,GSTIF(*)
C     FUNCTION:
C     ELIMINATION OF VARIABLES UPON LAST APPEARANCE
C
      NFUNC(I,J) =(J*J-J)/2+I
      DO 20 JFRON = 1,NFRON
         GL(JFRON) = GL(JFRON)-EQ(JFRON)*EQRHS/PIVOT
         IF(LC.EQ.1)  THEN
            IF(EQ(JFRON).NE.0)  THEN
               NLOCA = NFUNC(0,JFRON)
               DO 10 LFRON = 1,JFRON
                  NGASH = LFRON+NLOCA
                  GSTIF(NGASH) = GSTIF(NGASH)-EQ(JFRON)*
     $                                    EQ(LFRON)/PIVOT
 10            CONTINUE
            ENDIF
         ENDIF
 20   CONTINUE
      RETURN
      END
```

```
C*****************************************************************
C*                      BACK SUBSTITUTION                        *
C*****************************************************************
      SUBROUTINE BKSUB(FFIX ,KELVA ,VECRV ,MFRON ,Q ,LC, EQ)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION       FFIX(*) ,VECRV(*) ,EQ(*) ,Q(*)
C
      DO 1Ø IELVA=1,KELVA
         KK=KELVA+1-IELVA
         READ(17,REC=KK) (EQ(I),I=1,MFRON),EQRHS,IFRON,NIKNO
         IF (LC.GT.1) THEN
             READ(18,REC=KK) EQRHS
         ENDIF
         PIVOT =EQ(IFRON)
         IF(FFIX(NIKNO).NE.1ØØØØ) THEN
            VECRV(IFRON) = FFIX(NIKNO)
         ELSE
            EQ(IFRON) =Ø.Ø
         ENDIF
         DO 2Ø JFRON = 1,MFRON
            EQRHS = EQRHS - VECRV(JFRON)*EQ(JFRON)
 2Ø      CONTINUE
         IF(FFIX(NIKNO).EQ.1ØØØØ) THEN
            VECRV(IFRON) = EQRHS/PIVOT
         ENDIF
         Q(NIKNO)=VECRV(IFRON)
 1Ø   CONTINUE
      RETURN
      END
```

```
C*****************************************************************
C*                              RESULT                           *
C*****************************************************************
      SUBROUTINE RESULT(Q ,MINC ,NE    ,NJ )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION        Q(*)   ,MINC(2,*)    ,P(3)
C
C     INITIALIZE THE JOINT FORCES, P, TO ZERO ; CALL
C     FORCES AND OUTPUT.
      OPEN(21,FILE='JOINT.FIL',STATUS='NEW',ACCESS='DIRECT',
     $RECL=24)
      DO 10 I = 1,3
         P(I) = 0.0
   10 CONTINUE
      DO 20 J = 1,NJ
          WRITE(21,REC=J) (P(I),I=1,3)
   20 CONTINUE
      CALL FORCES(Q    ,MINC   ,NE)
      CALL OUTPUT(Q    ,NE     ,NJ)
      RETURN
      END
```

```
C*****************************************************************
C*                              FORCES                           *
C*****************************************************************
      SUBROUTINE FORCES(Q   ,MINC   ,NE)
      IMPLICIT REAL*8  (A-H,O-Z)
      DIMENSION         Q(*)   ,MINC(2,*)
C
C     FOR EACH ELEMENT I, CALL ELEMF AND JOINTF.
C
      DO 10 I=1, NE
         READ(16,REC=I) AREA, ZI, EMOD,   CTE, C1, C2, ELENG
         CALL ELEMF ( Q,AREA, ZI, EMOD, ELENG, C1, C2, MINC,I)
         CALL JOINTF(                          C1, C2, MINC,I)
   10 CONTINUE
      RETURN
      END
```

```
C*****************************************************************
C*                              ELEMF                            *
C*****************************************************************
      SUBROUTINE ELEMF(Q, AREA, ZI, EMOD, ELENG, C1, C2, MINC,I)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION  D(6) ,MINC(2,*) ,Q(*) ,QHAT(6)
C
C     COMPUTE THE LOCAL FORCES IN ELEMENT I, QHAT(6) :
C     DETERMINE THE GLOBAL ELEMENT DISPLACEMENTS, D(6),
C     FROM THE JOINT DISPLACEMENT VECTOR, Q, COMPUTE THE
C     LOCAL END AT THE A-END OF THE ELEMENT BY EQS.
C     2.34, 2.36 & 3.59 ; USE EQUILIBRIUM TO COMPUTE
C     THE LOCAL FORCES AT THE B-END OF THE ELEMENT
C     AND EQ. 3.95 TO COMPUTE THE ACTUAL ELEMENT FORCES.
C
      ALFA=EMOD*ZI/ELENG**3
      BETA=AREA*ELENG**2/ZI
C
C     GLOBAL ELEMENT DISPLACEMENTS:
C     ACTUAL DISPLACEMENTS
      JA = IABS(MINC(1,I))
      JB = IABS(MINC(2,I))
      DO 10 L=1,3
          JALOC = (JA-1)*3+L
          JBLOC = (JB-1)*3+L
          D(L)  = Q(JALOC)
          D(L+3)= Q(JBLOC)
   10 CONTINUE
C     RELATIVE DISPLACEMENTS
      D14=D(1)-D(4)
      D25=D(2)-D(5)
      D36=D(3)+D(6)
C
C     FORCES AT A-END DUE TO END DISPLACEMENTS
      F1=ALFA*BETA*(C1*D14+C2*D25)
      F2=6.0*ALFA*(2.0*C1*D25-2.0*C2*D14+ELENG*D36)
      F3=2.0*ALFA*ELENG*(3.0*C1*D25-3.0*C2*D14+ELENG*(D(3)+D36))
      READ(19,REC=I) (QHAT(J),J=1,6)
C     ACTUAL ELEMENT-END FORCES
      QHAT(1)=QHAT(1)+F1
      QHAT(2)=QHAT(2)+F2
      QHAT(3)=QHAT(3)+F3
      QHAT(4)=QHAT(4)-F1
      QHAT(5)=QHAT(5)-F2
      QHAT(6)=QHAT(6)+ELENG*F2-F3
      WRITE(19,REC=I) (QHAT(J),J=1,6)
      RETURN
      END
```

```
C*****************************************************************
C*                              JOINTF                           *
C*****************************************************************
      SUBROUTINE JOINTF(C1    ,C2    ,MINC   ,I)
      IMPLICIT REAL*8  (A-H,O-Z)
      DIMENSION        QHAT(6)   ,MINC(2,*) ,P1(3) ,P2(3)
C
C     TRANSFORM THE LOCAL FORCES OF ELEMENT I, QHAT(6), TO
C     GLOBAL FORCES AND ASSIGN THEM TO THE JOINT FORCES , P,
C     BY EQS. 2.29, 2.34, 2.37, AND MINC.
C
C     STATEMENT FUNCTION
      FG(C1I,C2I,FLX,FLY)=C1I*FLX+C2I*FLY
C
      J=IABS(MINC(1,I))
      K=IABS(MINC(2,I))
      READ(21,REC=J) (P1(N),N=1,3)
      READ(21,REC=K) (P2(N),N=1,3)
      READ(19,REC=I) (QHAT(N),N=1,6)
      P1(1)=P1(1)+FG(C1,-C2,QHAT(1),QHAT(2))
      P1(2)=P1(2)+FG(C2, C1,QHAT(1),QHAT(2))
      P1(3)=P1(3)+QHAT(3)
C
      P2(1)=P2(1)+FG(C1,-C2,QHAT(4),QHAT(5))
      P2(2)=P2(2)+FG(C2, C1,QHAT(4),QHAT(5))
      P2(3)=P2(3)+QHAT(6)
      WRITE(21,REC=J) (P1(N),N=1,3)
      WRITE(21,REC=K) (P2(N),N=1,3)
      RETURN
      END
```

```
C*****************************************************************
C*                              OUTPUT                           *
C*****************************************************************
      SUBROUTINE OUTPUT(Q ,NE ,NJ)
      IMPLICIT REAL*8  (A-H,O-Z)
      DIMENSION        QHAT(6) ,P1(3) ,P2(3) ,Q(*) ,U(3)
C
C     USE JOINT DISPLACEMENT VECTOR, Q, TO PRINT JOINT
C     DISPLACEMENTS (INCLUDING JOINT CONSTRAINTS) ;
C     PRINT LOCL ELEMENT FORCES, QHA(6,NE) ; PRINT
C     JOINT FORCES, P(3,NJ)
C
      WRITE(6,10) 'LOCAL ELEMENT FORCES','A-END','B-END',
     $'ELEMENT','DIRECT-1','DIRECT-2','DIRECT-3','DIRECT-4',
     $'DIRECT-5','DIRECT-6'
   10 FORMAT(///T10,A/T33,A,T65,A/T1,'+',T18,30('_'),T51,
     $30('_')/T10,A,T18,A,5(3X,A))
      DO 30 I=1,NE
         READ(19,REC=I) (QHAT(L),L=1,6)
         WRITE(6,20) I,(QHAT(L),L=1,6)
   20    FORMAT(T10,I3,T13,6(1X,F10.3))
   30 CONTINUE
C
      WRITE(6,40) 'JOINT DISPLACEMENTS (GLOBAL)','JOINT',
     $         'DIRECTION-1','DIRECTION-2','DIRECTION-3'
   40 FORMAT(///T10,A/T10,A,T17,A,T30,A,T43,A)
      DO 70 J=1,NJ
         DO 50 L=1,3
            K=(J-1)*3+L
            U(L)=Q(K)
   50    CONTINUE
            WRITE(6,60) J,(U(K), K=1,3)
   60       FORMAT(T11,I3,T18,F10.5,T31,F10.5,T44,F10.5)
   70 CONTINUE
C
      WRITE(6,40) 'JOINT FORCES (GLOBAL)','JOINT','DIRECTION-1',
     $         'DIRECTION-2','DIRECTION-3'
      DO 100 J=1,NJ
         READ(21,REC=J) (P1(L),L=1,3)
         WRITE(6,90) J,(P1(L),L=1,3)
   90    FORMAT(T11,I3,T18,F10.3,T31,F10.3,T44,F10.3)
  100 CONTINUE
      RETURN
      END
```

The vita has been removed from the scanned document