This is to certify that the

thesis entitled

COUPLING OF THE FINITE ELEMENT
AND BOUNDARY ELEMENT METHODS
FOR THE SOLUTION OF PLANE PROBLEMS
OF LINEAR ELASTICITY

presented by

Jose Jamie Garcia

has been accepted towards fulfillment
of the requirements for

M.S. degree in Mechanics

Nicholas J Altiero
Major professor

Date 12/7/84

COUPLING OF THE FINITE ELEMENT AND BOUNDARY ELEMENT METHODS FOR THE SOLUTION OF PLANE PROBLEMS OF LINEAR ELASTICITY

By

Jose Jaime Garcia

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Department of Metallurgy, Mechanics and Materials Science

1984

ABSTRACT

COUPLING OF THE FINITE ELEMENT
AND BOUNDARY ELEMENT METHODS
FOR THE SOLUTION OF PLANE
PROBLEMS OF
LINEAR ELASTICITY

By

Jose Jaime Garcia A.

A hybrid finite element-boundary element formulation for the solution of elastostatics plane problems is presented. Only isotropic materials are considered. A finite element program, FEM, which uses triangular and quadrilateral isoparametric elements, is initially developed. Next, a direct boundary element program, BEM, is created using a piecewise straight boundary discretization. Constant tractions and linear displacements are assumed on each boundary element. This model allows traction discontinuities at the nodes and is compatible with the elements used in FEM. Finally, a mixed formulation is obtained in the program FEMBOU, which uses standard finite elements and multi-node elements or "superelements", whose symmetric stiffness matrices, created using the boundary integral equations, are incorporated into a standard finite element program, similar to FEM. Results of the three programs are compared for two examples.

ACKNOWLEDGEMENTS

The author wishes to express his gratitude and deep appreciation to his advisor, Professor Nicholas J. Altiero, for his able guidance and sincere encouragement during the course of this thesis research, and for his spending many hours with the author in the preparation of this thesis. Special thanks are due to General Dynamics Land Systems Division, Warren, Michigan (Contract No. DEY-600484, Print. Investigator: N. Altiero) for its support of this research.

The author also wishes to thank The Universidad del Valle, Cali, Colombia and The Fulbright Commission for their support of his studies in the United States during the last year. Finally, the author wishes to acknowledge his parents for their continuous encouragement and moral support in the past several years.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

The classical mixed boundary value problem of linear elastostatics, shown in Figure I.1, consists of an elastic body, R, loaded by specified tractions, t_i, on part B_t of the boundary and specified displacements, u_i, on the remainder of the boundary B_u. Body forces are neglected here. The displacement and stress fields everywhere in R, subject to the boundary conditions, are sought. Exact solutions have been obtained for many problems in which simple geometry is involved, such as rectangular, circular, elliptical bodies with particular boundary conditions. Nevertheless, it is quite difficult to obtain the exact solution for a problem in which either the body shape is more complicated or the load is more general. Under these circumstances, numerical techniques have been developed to approximate the solution of any general elastostatics problem. The most notable of these procedures are the Finite Element Method and the Boundary Element Method, which have become practical only after the development of the high-speed digital computer.

The Finite Element Method [1,2,3],* first used for solving structural problems in the 1960's, has become a powerful engineering tool which can be applied to a wide range of problems such as fluid flow, electromagnetic fields, etc. The method consists of dividing the continuum into a set of imaginary discrete elements or "finite elements", which are defined to interact with the neighboring elements only at

* Brackets refer to references listed in BIBLIOGRAPHY

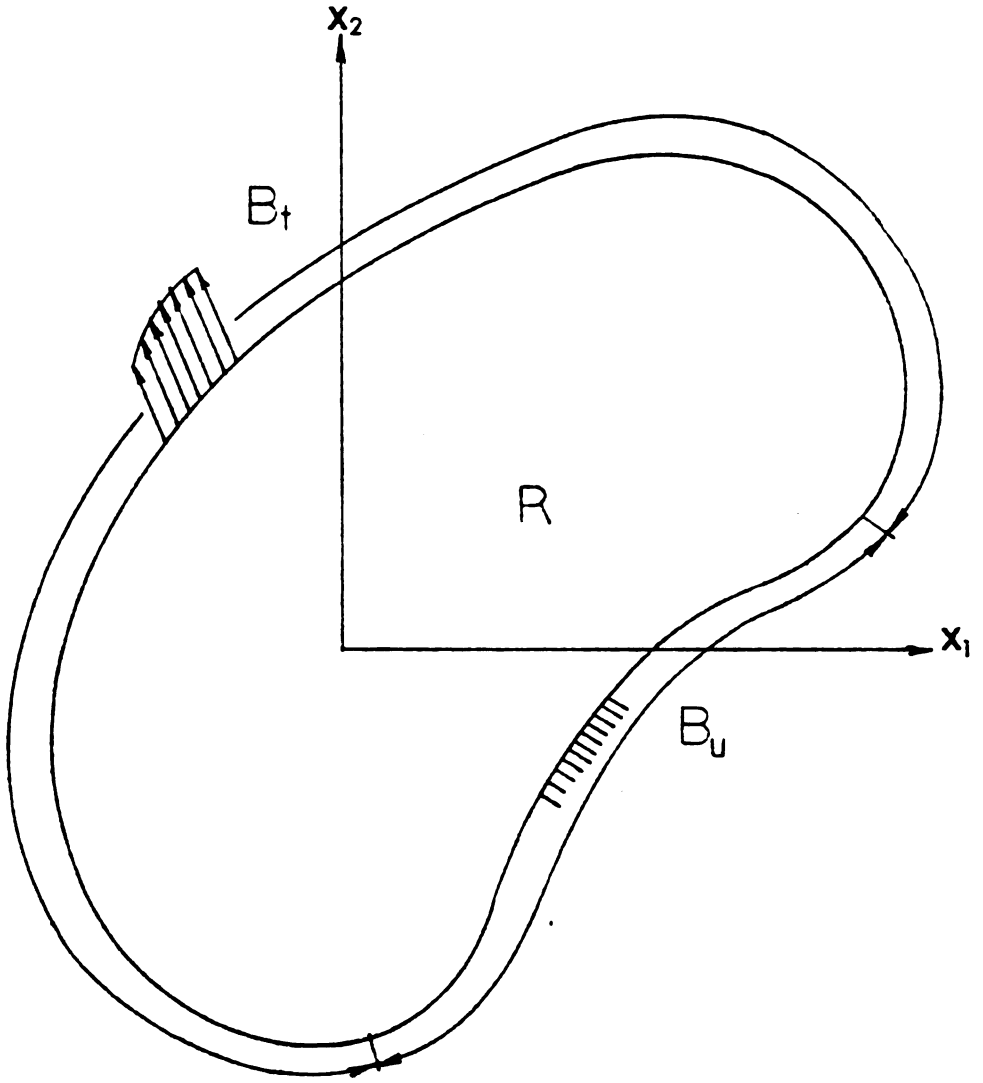


Figure I.1. Mixed boundary value problem

specified points on their boundaries, called "nodes". Governing field equations are approximated within the elements by prescribed interpolating functions or "shape functions". Initial engineering intuition was that such a model should produce results similar to those of the real problem, a fact which has since been amply proven by mathematicians.

In the case of a three dimensional problem, it is necessary to handle a large amount of data for specification of elemental and nodal information of the discretized model. This is a major drawback of the method, especially when a large number of small elements is required to accurately predict high stress gradients.

The Boundary Element Method [4,5], introduced for elasticity problems in the late 1960's and developed through the 1970's, solves governing integral equations by dividing the boundary of the body into a set of small segments or "boundary elements", on which certain properties (tractions, displacements, etc) are approximated by interpolating functions or "shape functions". In the case of the "direct" method for elastostatics, Somigliana's integral equation is discretized. At least one exact singular solution must be known in order to implement the approximate equations. The dimensional discretization is one order less than that for the Finite Element Method, this being an important advantage in particular when solving three dimensional problems. Moreover, field equations are satisfied exactly within the body, allowing accurate calculation of field properties, even if there are high gradients involved. Furthermore, the Boundary Element Method can be implemented to handle problems over infinite or semi-infinite domains.

Due to the fact that boundary data is discretized, problems arise for the method when solving cases in which the ratio of boundary area to

domain volume becomes large, i.e. when most of the domain lies in the proximity of the boundary.

In the hope that the difficulties of the two methods can be avoided, a mixed Finite Element - Boundary Element formulation has often been suggested and actually implemented using various approaches, [3,4,6,7,8,9,10].

In this dissertation, programs based on the Finite Element and Boundary Element methods are implemented, with the goal of using them to produce a third "combined" formulation, which is subsequently presented. Additionally, results of the three programs are compared for some example problems.

The Finite Element Program, FEM, described in CHAPTER II, follows standard procedures for saving the global stiffness matrix, solving the equations, etc., as recommended by various Finite Element authors, [1,2,3,10]. Numerical integration is used to create elemental stiffness matrices. The extension to the three dimensional problem can be easily accomplished.

The Boundary Element program, BEM, described in CHAPTER III, is based on the direct approach. The boundary discretization consists of straight segments in which displacements vary linearly and tractions are constant. This model allows traction discontinuities and is compatible with the finite elements used in FEM.

The hybrid program, FEMBOU, described in CHAPTER IV, consists of the program, FEM, plus a subroutine, BOUN, which creates the stiffness matrix of a "superelement", in which the boundary integral equations are valid.

In the final chapter of the work, results of the three programs are presented and compared with regard to accuracy and computational speed.

Some conclusions can be drawn from these results but no definitive statement can be made as to which is the "best" method. This work is just a beginning toward the goal of implementation of the methods to the general three dimensional anisotropic case.

CHAPTER II

FINITE ELEMENT MODELLING

II.1 DESCRIPTION OF THE FINITE ELEMENT METHOD

The displacement approach, or direct formulation, of the Finite Element Method, consists of dividing the body into imaginary "finite elements" as shown for a plane problem in Figure II.1. Each element is connected to neighboring elements at specified points called nodes.

The components u_1, u_2 of the displacement at a point within an element e is given approximately by

$$u_1(x) = \sum_{i=1}^{n} N_i(x) u_1^{(i)}, \quad u_2(x) = \sum_{i=1}^{n} N_i(x) u_2^{(i)} \qquad \text{(II.1)}$$

where $u_1^{(i)}$, $u_2^{(i)}$ are the components of the displacement of node i, $N_i(x)$ is a specified function corresponding to that node, or "shape function" of the node i, and n is the number of element nodes. There are as many shape functions as nodes in the element. These shape functions must satisfy certain requirements, i.e., they are equal to 1 at the corresponding node and are equal to 0 at all other nodes of the element. Also, they must be able to reproduce exactly a state of constant stress [2].

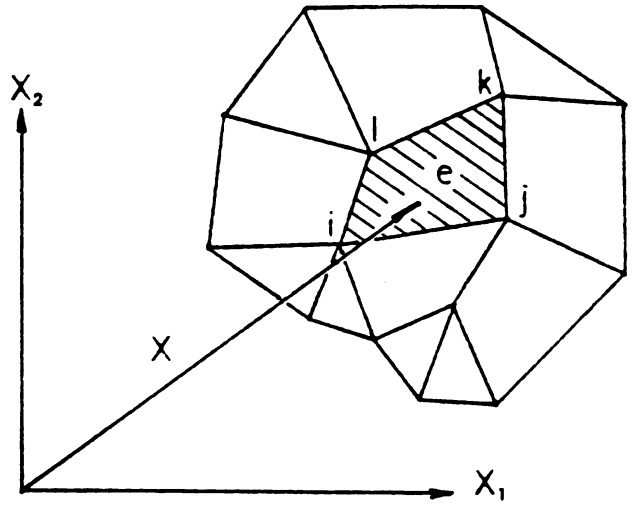


Figure II.1. Domain discretization for Finite Element Analysis

If the nodal displacements are known, the stress and strain can be calculated in the element. For a plane problem, the components of strain are

$$\{\varepsilon\}=\begin{Bmatrix}\varepsilon_{11}\\ \varepsilon_{22}\\ 2\varepsilon_{12}\end{Bmatrix}=\begin{Bmatrix}\partial u_1/\partial x_1\\ \partial u_2/\partial x_2\\ \partial u_1/\partial x_2+\partial u_2/\partial x_1\end{Bmatrix}=\begin{bmatrix}\partial N_1/\partial x_1 & 0 & \cdots\cdots & \partial N_n/\partial x_1 & 0\\ 0 & \partial N_1/\partial x_2 & \cdots\cdots & 0 & \partial N_n/\partial x_2\\ \partial N_1/\partial x_2 & \partial N_1/\partial x_1 & \cdots & \partial N_n/\partial x_2 & \partial N_n/\partial x_1\end{bmatrix}\begin{Bmatrix}u_1^{(1)}\\ u_2^{(1)}\\ \vdots\\ u_1^{(n)}\\ u_2^{(n)}\end{Bmatrix}$$

or

$$\{\varepsilon\}=[B_1\ B_2\ \ldots\ B_n]\{u\}^{(e)} \qquad \text{(II.2)}$$

where

$$[B_i]=\begin{bmatrix}\partial N_i/\partial x_1 & 0\\ 0 & \partial N_i/\partial x_2\\ \partial N_i/\partial x_2 & \partial N_i/\partial x_1\end{bmatrix}$$

and $\{u\}^{(e)}$ is the vector of nodal coordinates of the element.

With the elastic constants of the material, Young's Modulus E and Poisson's Ratio υ, the components of stress are

$$\{\sigma\}=\begin{Bmatrix}\sigma_{11}\\ \sigma_{22}\\ \sigma_{12}\end{Bmatrix}=\frac{E}{1-\upsilon^2}\begin{bmatrix}1 & \upsilon & 0\\ \upsilon & 1 & 0\\ 0 & 0 & \frac{(1-\upsilon)}{2}\end{bmatrix}\begin{Bmatrix}\varepsilon_{11}\\ \varepsilon_{22}\\ 2\varepsilon_{12}\end{Bmatrix}=[D]\{\varepsilon\}=[D][B]\{u\}^{(e)} \qquad \text{(II.3)}$$

in the case of plane stress.

If the potential energy of the element,

$$\Pi^{(e)}=\tfrac{1}{2}\int_{Ve}\{\sigma\}^{T}\{\varepsilon\}dv-\{F\}^{(e)T}\{u\}^{(e)},$$

or

$$\Pi^{(e)}=\tfrac{1}{2}\{u\}^{(e)T}\int_{Ve}[B]^{T}[D][B]dv\{u\}^{(e)}-\{F\}^{(e)T}\{u\}^{(e)} \qquad \text{(II.4)}$$

is minimized, a relationship between the nodal forces $\{F\}^{(e)}$ and the nodal displacements $\{u\}^{(e)}$ is obtained. This relationship is

$$[K]^{(e)}\{u\}^{(e)}=\{F\}^{(e)},$$

where the matrix $[K]^{(e)}$ is the stiffness matrix of the element e,i.e.

$$[K]^{(e)} = \int_{Ve} [B]^T [D][B] dv \qquad \text{(II.5)}$$

The global stiffness matrix which relates the global displacement vector {u} to the global force vector {F}, is also obtained by minimizing the total potential energy of the body. The final system of equations can be written as,

$$[K]\{u\} = \{F\} \qquad \text{(II.6)}$$

where {u} is a vector containing all nodal displacements, {F} is a vector containing externally applied nodal forces, and [K] is symmetric and banded. The entries of [K] are composed of contributions of the elemental stiffness matrices. The final system of equations can be solved for the unknown displacements, and subsequently the state of stress and strain can be calculated at all points of the body using eqs. (II.3) and (II.2), respectively.

The general algorithm of a finite element program is as follows:

- Read the data of the discretized model.
- Form the global stiffness matrix from the contributions of all elemental stiffness matrices.
- Modify the system of equations for specified displacements.
- Calculate displacements and stresses at prescribed points in the body.

II.2 ISOPARAMETRIC QUADRILATERAL ELEMENTS

In the case of an isoparametric element, the relation between the local and the global coordinate system is given by,

$$x_1 = \sum_{i=1}^{n} N_i(\xi,\eta) x_1^{(i)} \quad , \quad x_2 = \sum_{i=1}^{n} N_i(\xi,\eta) x_2^{(i)} \qquad \text{(II.7)}$$

where $N_i(\xi,\eta)$ is the shape function of node i. In the case of a four-node quadrilateral, these can be represented by the equation,

$$N_i(\xi,\eta) = \tfrac{1}{4}(1+\xi\xi_i)(1+\eta\eta_i) \qquad \text{(II.8)}$$

where ξ_i and η_i are the local coordinates of node i, as shown in Figure II.2. For example, the local coordinates of node 2 are $\xi_2 = 1$ and $\eta_2 = -1$ so that

$$N_2(\xi,\eta) = \tfrac{1}{4}(1+\xi)(1-\eta)$$

Values of the displacements in the element are related to those at the nodes by the same shape functions (thus the description "isoparametric") so that

$$u_1(x) = \sum_{i=1}^{n} N_i(\xi,\eta) u_1^{(i)} \quad , \quad u_2(x) = \sum_{i=1}^{n} N_i(\xi,\eta) u_2^{(i)} \qquad \text{(II.9)}$$

where n is the number of nodes in the element.

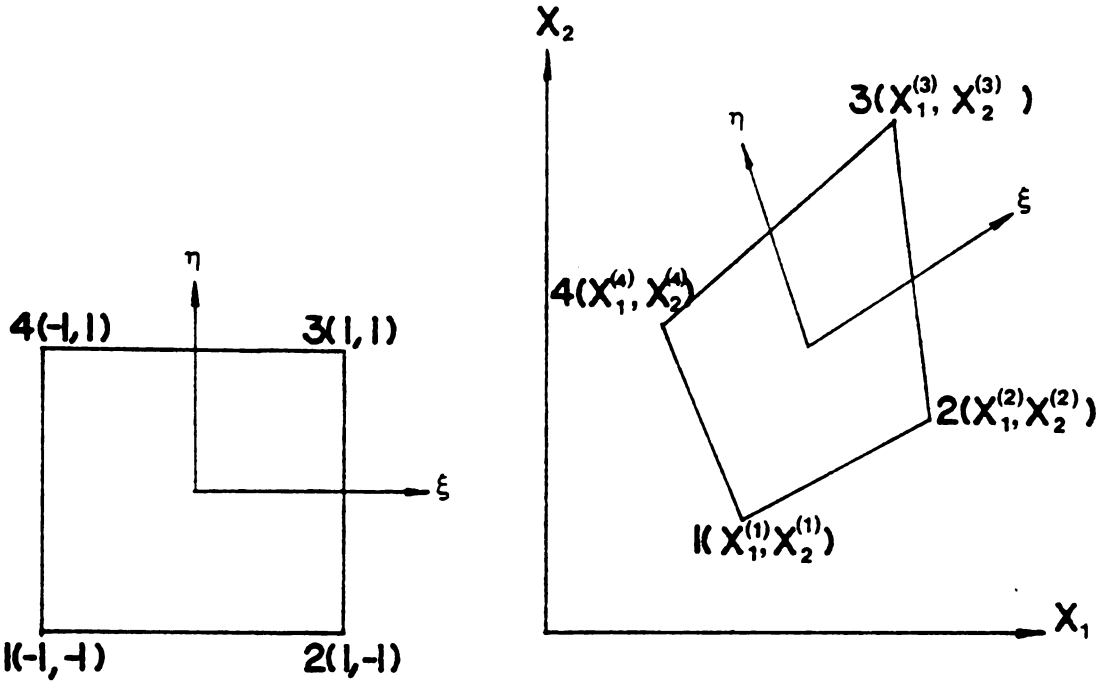


Figure II.2. Local and global coordinates for a four-node isoparametric element

Since the shape functions are in terms of the local coordinates (ξ, η), a transformation is necessary to calculate the derivatives of the shape functions with respect to the global coordinates x_1 and x_2,

$$\begin{Bmatrix} \partial N_i/\partial x_1 \\ \partial N_i/\partial x_2 \end{Bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} (\partial N_i/\partial \xi) x_1^{(i)} & \sum_{i=1}^{n} (\partial N_i/\partial \xi) x_2^{(i)} \\ \sum_{i=1}^{n} (\partial N_i/\partial \eta) x_1^{(i)} & \sum_{i=1}^{n} (\partial N_i/\partial \eta) x_2^{(i)} \end{bmatrix}^{-1} \begin{Bmatrix} \partial N_i/\partial \xi \\ \partial N_i/\partial \eta \end{Bmatrix}$$

$$= [J]^{-1} \begin{Bmatrix} \partial N_i/\partial \xi \\ \partial N_i/\partial \eta \end{Bmatrix} \qquad \text{(II.10)}$$

where [J] is the Jacobian matrix.

As is explained by Gupta in [11], the most efficient way to calculate the stiffness of the element is to evaluate first the (2x2) submatrix,

$$\begin{bmatrix} \int_{Ve} (\partial N_i/\partial x_1)(\partial N_j/\partial x_1) dv & \int_{Ve} (\partial N_j/\partial x_1)(\partial N_i/\partial x_2) dv \\ \int_{Ve} (\partial N_i/\partial x_1)(\partial N_j/\partial x_2) dv & \int_{Ve} (\partial N_j/\partial x_2)(\partial N_i/\partial x_2) dv \end{bmatrix} = \begin{bmatrix} W11 & W12 \\ W21 & W22 \end{bmatrix} \qquad \text{(II.11)}$$

and then to rearrange it into another 2x2 submatrix

$$E/(1-\upsilon^2) \begin{bmatrix} W11+W22(1-\upsilon)/2 & \upsilon W12+W21(1-\upsilon)/2 \\ \upsilon W21+W12(1-\upsilon)/2 & W22+W11(1-\upsilon)/2 \end{bmatrix} \qquad \text{(II.12)}$$

located in the positions (2i-1, 2j-1), (2i-1, 2j), (2i, 2j-1), (2i,2j) of $[K]^{(e)}$.

All of the integrals are evaluated using numerical integration and Gauss points.

II.3 PLANE STRESS FINITE ELEMENT PROGRAM

FEM is a finite element program that can be used to solve elastostatics problems in two dimensions. Three-node triangular and four-node quadrilateral elements may be used in the program. A flow chart of FEM is shown in Fig. II.3.

Subroutine RED1 reads and prints the data related to material properties, geometry and boundary conditions. Subroutine PROFIL determines the dimensions and the addresses for the vectors in which portions of the global stiffness matrix are to be saved. PFORM creates the entries of the stiffness matrix by calling ELEMENT, which calculates elemental stiffness matrices, and ADDSTIF, which adds these elemental matrices into the global one. Subroutine SHAPE, called by ELEMENT, calculates the global derivatives of the shape functions at the points of integration, and PGAUSS computes the Gauss weights and points for the numerical integration. One, four, or nine Gauss points may be used to perform the numerical integration. The global stiffness matrix formed is then factored into lower and upper triangular matrices by the first part of the subroutine SOLVE(1). Several load cases can be solved for the same geometry within the loop from I=1 to NLO(number of load cases). MODIFY reads and prints non-zero specified displacements. SOLVE(2) solves the equations for the values of nodal displacements by back and forward substitution using the triangular matrices already calculated in

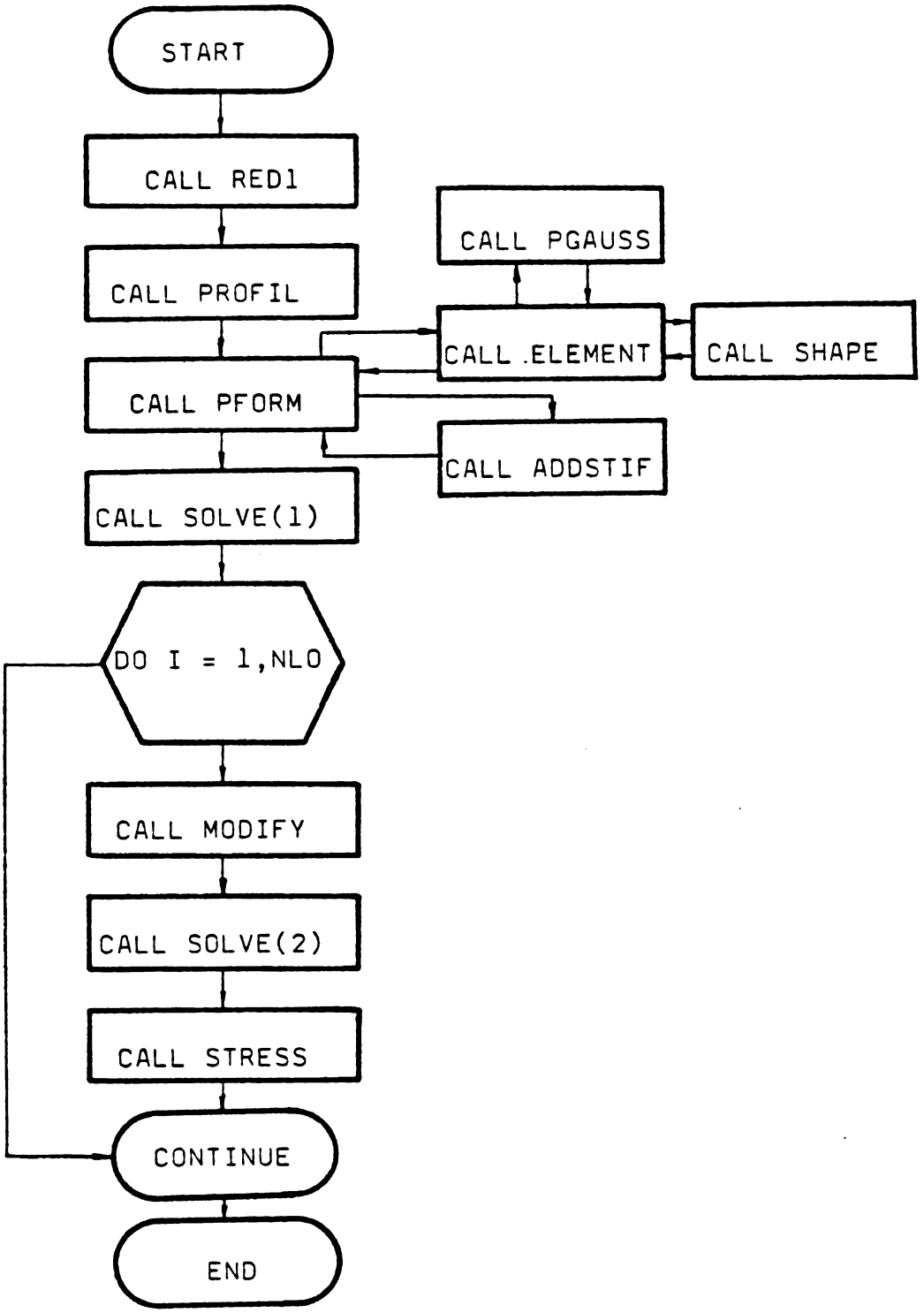


Figure II.3. Flowchart of Program FEM

SOLVE(1) and the load vector created in MODIFY. With the values of nodal displacements known, stresses and reactions are calculated and printed by STRESS. In the following sections detailed explanations of the subroutines are presented.

II.3.1 SUBROUTINE RED1

This subroutine reads and prints the input data. A list of the order and format in which the variables and arrays must be submitted to the program follows:

TITLE - The title of the problem. This must be written on one line in columns 1 through 80.

NMAT - The number of materials of which the body is composed. Enter in free format.

NEL - The number of elements used. Enter in free format.

NNO - The total number of nodes in the model. Enter in free format.

NDOF - The number of degrees of freedom of the problem. This parameter is equal to 2 for plane problems. Enter in free format.

NLO - The number of load cases to be solved for the same geometry. Enter in free format.

E(I),PR(I)- Material properties (E is Young's Modulus and PR is Poisson's Ratio). Enter for I=1, NMAT, one set of properties per line. Enter E(I) in columns 1 through 20 in E format, and PR(I) in columns 21 through 40 in F format.

MP(I), NEN(I), (NN(I,J), J=1, NEN(I)) - Element properties.

MP(I) is the material identification of the element (MP = 1,2,..., or NMAT), NEN(I) is the number of nodes of the element, and (NN(I,J),J=1,NEN(I)) are the number of the nodes which form the element (listed counter-clockwise). Enter for I=1, NEL, one set of element properties per line. These properties are read in free format.

X(I,1), NTBC(I,1), X (I,2), NTBC(I,2) - Node properties.

X(I,1) is the coordinate x_1 of node I, NTBC(I,1) is the type of boundary condition in the x_1 direction at node I, X(I,2) is the coordinate x_2 of node I, and NTBC(I,2) is the type of boundary condition in the x_2 direction at node I. Enter for I=1,NNO, one set of node properties per line. Enter X(I,1) in columns 1 through 15 in F format, NTBC (I,1) in columns 16 through 20 in I format, X(I,2) in columns 21 through 35 in F format and NTBC(I,2) in columns 36 to 40 in I format.

The boundary condition code for NTBC(I,K) is as follows,

NTBC(I,K)= -1 if specified zero displacement at node I in direction K.

NTBC(I,K)=0 if specified force at node I in direction K.

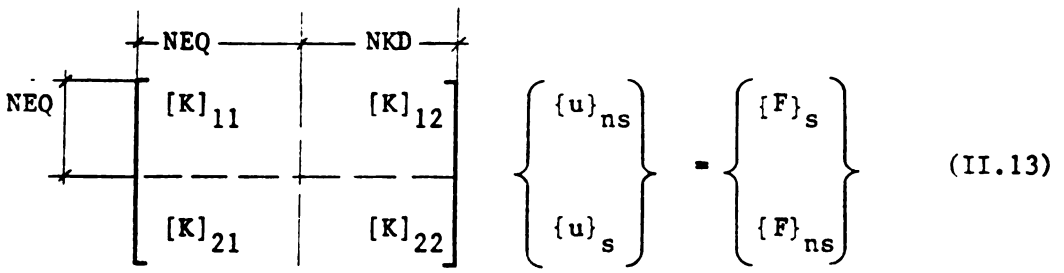
NTBC(I,K)=1 if specified non-zero displacement at node I in direction K.

The default value for NTBC(I,K) is 0.

Subroutine RED1 also prints the input data after it is read.

II.3.2 SUBROUTINE PROFIL

The global system of equations for a Finite Element program, given by eq. (II.6), can be rearranged as



$$\begin{bmatrix} [K]_{11} & [K]_{12} \\ [K]_{21} & [K]_{22} \end{bmatrix} \begin{Bmatrix} \{u\}_{ns} \\ \{u\}_{s} \end{Bmatrix} = \begin{Bmatrix} \{F\}_{s} \\ \{F\}_{ns} \end{Bmatrix} \qquad \text{(II.13)}$$

where $\{u\}_{ns}$ is a vector of unknown displacements, $\{u\}_{s}$ is a vector of specified displacements (zero and non-zero), $\{F\}_{s}$ is a vector of known external forces, $\{F\}_{ns}$ is a vector of unknown external forces or reactions, NEQ is the number of equations to be solved and NKD is the number of specified displacements. The number of equations is calculated as

NEQ = (No. of nodes)x(No. of degrees of freedom)- NKD

and the equations to be solved can be written as

$$[K]_{11}\{u\}_{ns}=\{F\}_{s}-[K]_{12}\{u\}_{s}=\{F\}_{s}^{*}. \qquad \text{(II.14)}$$

If the specified displacements are all zero, $\{F\}_{s}^{*} = \{F\}_{s}$ and it is not necessary to save $[K]_{12}$. It is noted that the submatrix $[K]_{11}$ is symmetric and banded.

The objective of the subroutine PROFIL is to calculate the dimensions of two vectors, A(J) and B(J), in which parts of $[K]_{11}$ and $[K]_{12}$ respectively are to be saved by columns, and to set two arrays, JDIAG(I) and IDIAG(I), that serve to identify positions of the elements of $[K]_{11}$ and $[K]_{12}$ in the vectors A(J) and B(J). Space is reserved in A(J) for all column entries in $[K]_{11}$ from the first non-zero entry to the diagonal. With this organization, considered by Taylor in [3], less space is required for saving the matrix. On the other hand, some additional time must be spent in order to set the arrays JDIAG(I) and IDIAG(I).

II.3.2.1 FIRST PART OF PROFIL

For each specified force condition, NTBC(I,K)=0, an equation is counted and the number of that equation is assigned to NTBC(I,K). For each specified non-zero displacement, NTBC(I,K)=1, the specified non-zero displacement is counted and the negative of that number is assigned to NTBC(I,K). For each specified zero displacement, NTBC(I,K)=-1, NTBC(I,K) is set equal to zero. For the problem of Figure II.4, the values of NTBC(I,K) before and after PROFIL are shown in Table II.1. In this example there are 2 specified non-zero displacements and 3 specified zero displacements.

II.3.2.2 SECOND PART OF PROFIL

For the same example of Figure II.4, the form of the complete stiffness matrix is shown in Figure II.5(a), where the positions above the upper profile line are zero. The form of the submatrix $[K]_{11}$ is shown in Figure II.5(b) and the positions marked as x are the ones to be saved in A(J) as is shown in the same figure.

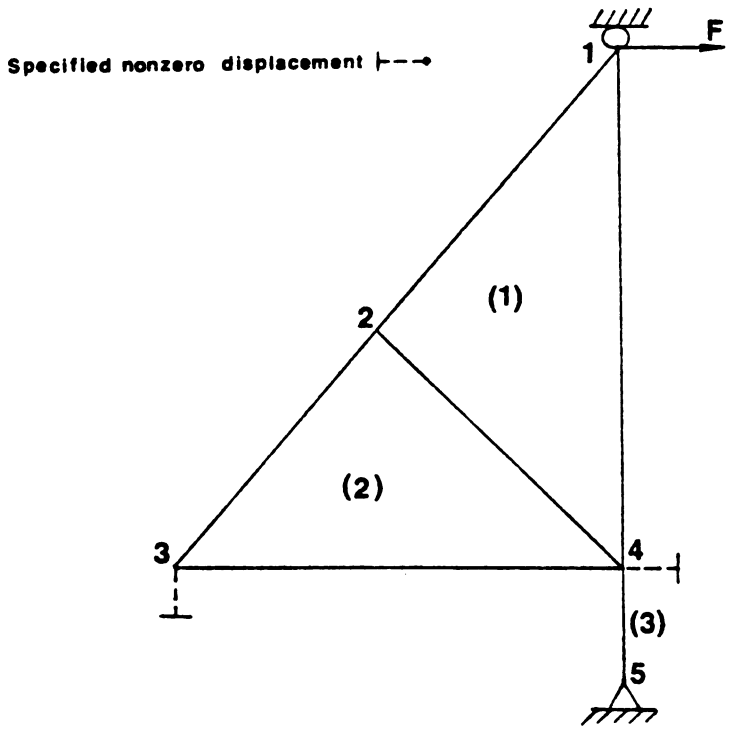
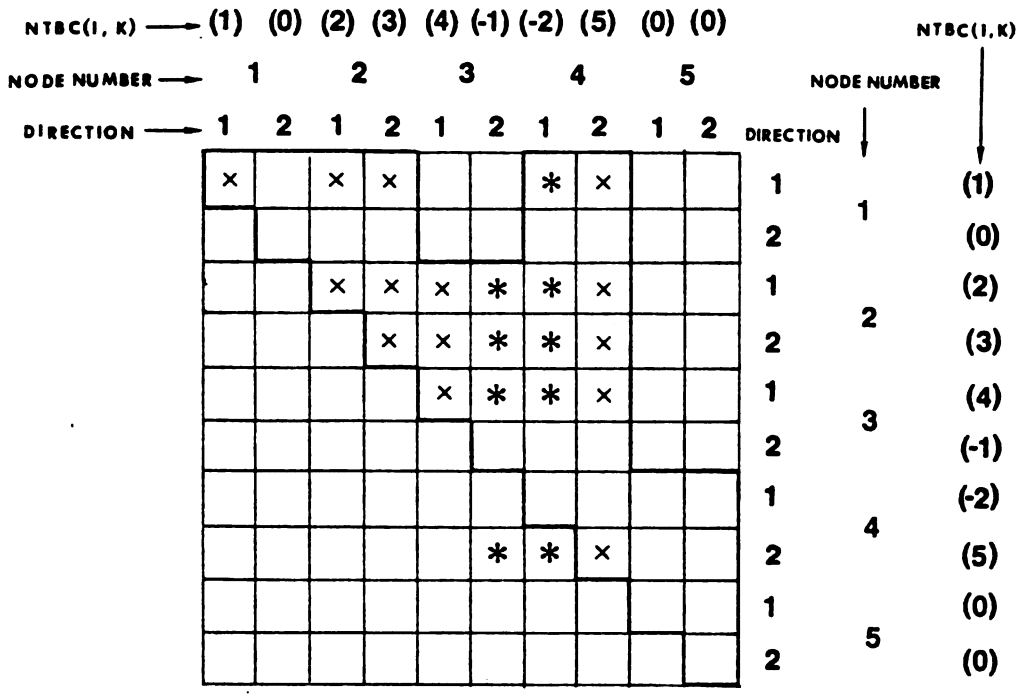


Figure II.4. Example for PROFIL

Table II.1 Array NTBC(I,K) before and after PROFIL

Node No. I	Degree of Freedom, K	NTBC(I,K) Before PROFIL	NTBC(I,K) After PROFIL
1	1	0	1
	2	-1	0
2	1	0	2
	2	0	3
3	1	0	4
	2	1	-1
4	1	1	-2
	2	0	5
5	1	-1	0
	2	-1	0



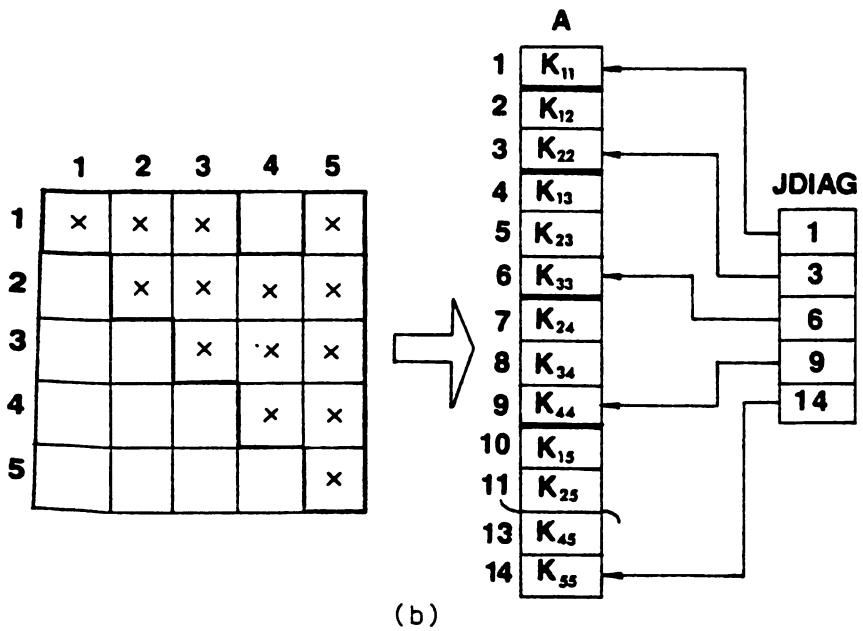


Figure II.5. Matrices [K], $[K]_{11}$ and arrays A(J) and JDIAG(J) for the example of figure II.4

The array JDIAG(I) relates the entries of A(J) to those of $[K]_{11}$. There are as many components in JDIAG(I) as there are equations. Each entry JDIAG(I) indicates the location in A(J) of the diagonal element of column I.

The number of non-zero positions in a column I of $[K]_{11}$ is equal to the maximum difference between the equation number I and the equation numbers corresponding to the degrees of freedom of the nodes adjacent to I, plus 1, i.e. the number of elements in column 4 of $[K]_{11}$ is equal to $\max\{|4-3|, |4-2|\} + 1 = 2+1 = 3$.

The algorithm to set the array JDIAG(I) is as follows.

- Do a loop on the elements.
- Do a loop on the nodes of the element.
- Do a loop on the degrees of freedom.
- Find the maximum difference between the equation I of that degree of freedom and the equation numbers of the other degrees of freedom of the same element. Assign that value to JDIAG(I).
- When the same equation number is found in another element, assign to JDIAG(I) the maximum value between the current maximum difference and the value already saved in JDIAG(I).
- After finishing these loops, the entry JDIAG(I) is equal to the height of column I minus 1.
- To obtain the final JDIAG(I) array, another loop is made to add the heights of all the columns.

In the array B(J) are saved the columns of [K] corresponding to the specified non-zero displacements. For the example of Figure II.4, these are marked as asterisks in Figure II.5. The number of entries in each column is equal to the difference between the highest and the lowest equation number of the degrees of freedom with entries in that column, plus 1, i.e. the number of entries in column No. 6 is (5-2)+1=4. There

are two entries in IDIAG(I) for each specified non-zero displacement I. The first, IDIAG(2I-1), indicates the equation number that corresponds to the first element of column I, and the second indicates the position in the array B(J) of the last element of column I. In Figure II.6, are shown the arrays B(J) and IDIAG(I) for the example of Figure II.4.

The algorithm to calculate IDIAG(I) is as follows.

- Do a loop on the elements.
- Do a loop on the nodes of the element.
- Do a loop on the degrees of freedom.
- For each negative NTBC(I,K), find the highest and the lowest values of the equation numbers of the other positive NTBC(I,J) corresponding to the other degrees of freedom of the same element. Save these values in IDIAG(2I-1) and IDIAG(2I).
- Add the heights of the columns to obtain the final array.

II.3.3 SUBROUTINE PFORM

This subroutine fills the arrays A(J) and B(J) in which the submatrices $[K]_{11}$ and $[K]_{12}$ are saved.

The flow chart for PFORM is shown in Figure II.7. In the first part, PFORM calculates the entries of the matrix [D], which relates stress to strain. These entries are saved in the arrays E(I), PR(I) and G(I).

The second part of PFORM consists of a loop on the elements. For each element I, the material properties are localized in D(1), D(2), D(3), the coordinates of the nodes in XE(I,L) and the boundary condition types at the nodes in NBC (I,L). Also the local stiffness matrix S(I,L) is initialized. With the arrays D(I) and XE(I,L), the stiffness of the element is calculated by the subroutine ELEMENT. With the arrays

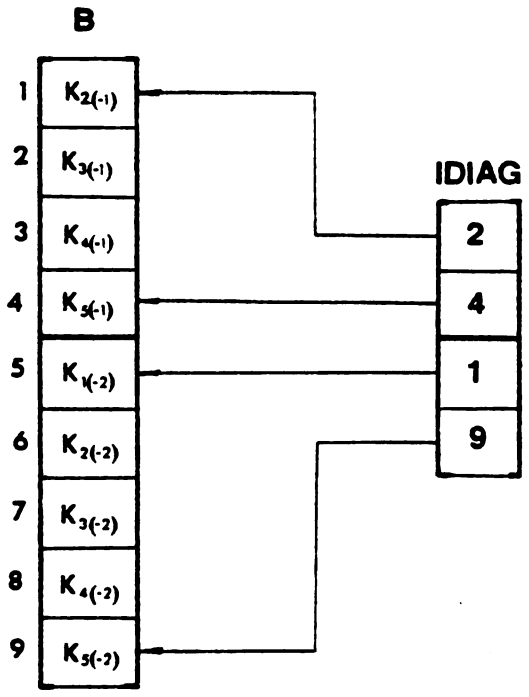


Figure II.6. Arrays B(J) and IDIAG(J) for example of Figure II.4

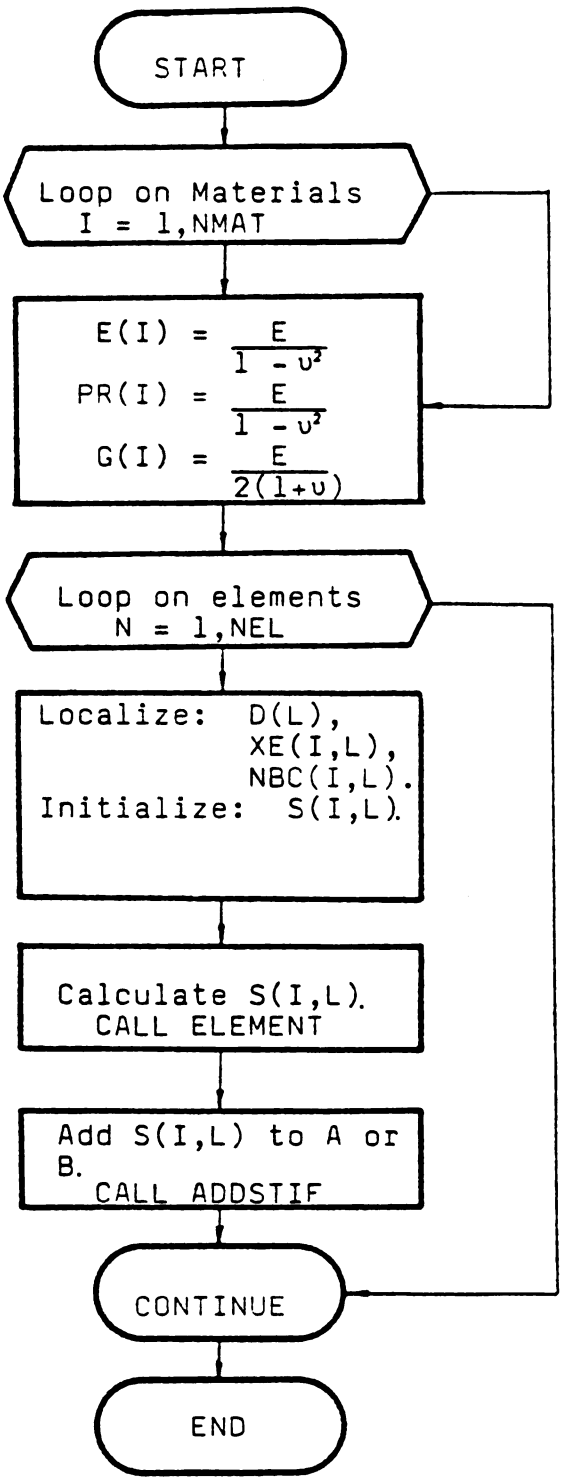


Figure II.7. Flowchart of PFORM

NBC(I,L), JDIAG(I), IDIAG(I), the entries of the local stiffness matrix are located in A(J) or B(J) by the subroutine ADDSTIF.

II.3.3.1 SUBROUTINE ELEMENT

This subroutine calculates the stiffness matrix of three-node triangular and four-node quadrilateral elements by numerical integration. The flow chart of ELEMENT is shown in Figure II.8.

In the first part, ELEMENT calculates the entries W11, W12, W21, W22 of eq. (II.11). Each of these entries is approximately equal to

$$W12=\int_{Ve}(\partial N_j/\partial x_1)(\partial N_i/\partial x_2)dv=\int_{-1}^{1}\int_{-1}^{1}(\partial N_j/\partial x_1)(\partial N_i/\partial x_2)\det|J|\,d\xi\,d\eta$$

$$=\sum_{k=1}^{N^2} W_{kk}\det|J|(\partial N_j/\partial x_1)(\partial N_i/\partial x_2)(\xi_k,\eta_k) \qquad (II.15)$$

where N is the number of points of integration in each direction, W_{kk} are the Gauss weights and ξ_k, η_k are the local coordinates of the Gauss points. The subroutine PGAUSS calculates W_{kk}, ξ_k, η_k, depending on the value of N. The subroutine SHAPE calculates the derivatives of the shape functions at the points of integration. The summation from 1 to N^2 is performed in ELEMENT.

In the second part of ELEMENT, the entries W11, W12, W21, W22 are rearranged and multiplied by the material constants to obtain the components of eq. (II.12).

Subroutine SHAPE returns the values of the global derivatives $\partial N_i/\partial x_1$, $\partial N_i/\partial x_2$ in the array SHP(I,K), where I indicates the shape function number and K the derivative,i.e., K=1→$\partial N_i/\partial x_1$, K=2→$\partial N_i/\partial x_2$, K=3→$N_i$. The sequence of operations in SHAPE is essentially that of equation (II.10). First the local derivatives are formed, then the Jacobian matrix [J] and its inverse $[J]^{-1}$ are calculated, and finally the multiplication of $[J]^{-1}$ by local derivatives is performed. When it is

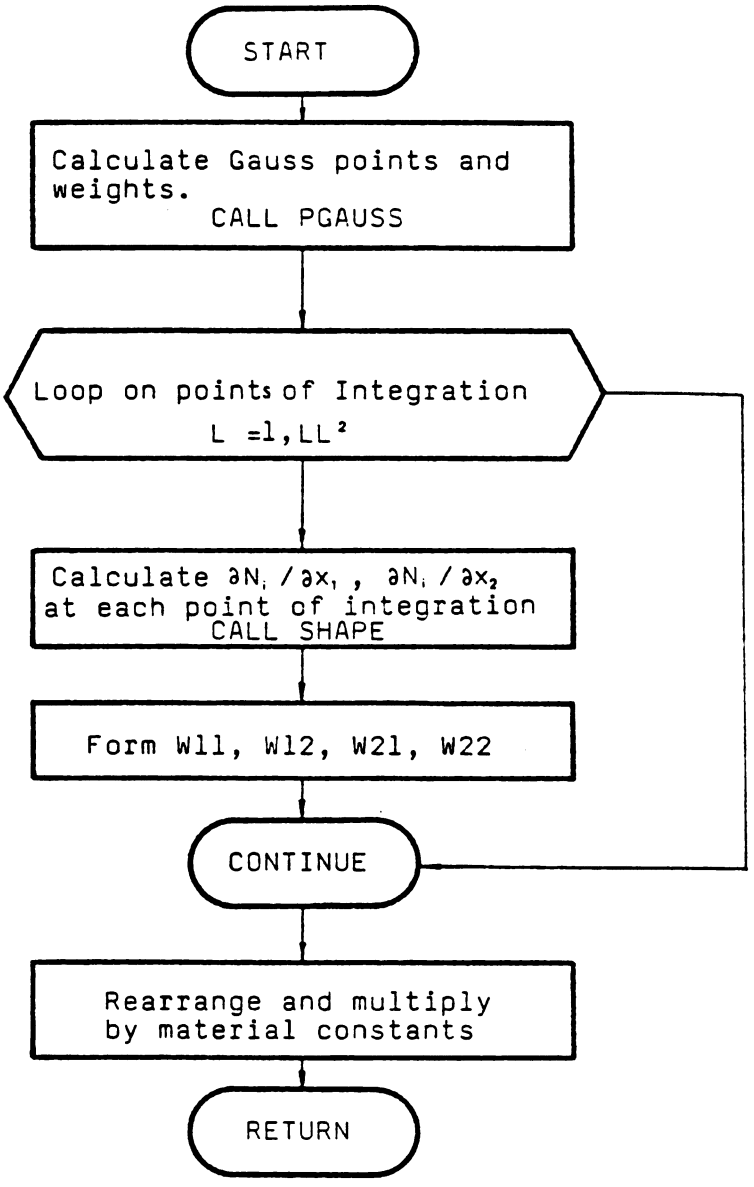


Figure II.8. Flowchart of ELEMENT

required to find the stiffness matrix of a triangular element, the shape functions N_3 and N_4 are added, and the same procedure is followed in all other steps.

II.3.3.2 SUBROUTINE ADDSTIF

This subroutine stores the values of elemental stiffness matrices in either A(J) or B(J), depending on the boundary condition NBC(I,K) at each degree of freedom.

The algorithm for ADDSTIF is,

- Do a loop on the nodes of the element, I=1, NEN(N).
- Do a loop on the degrees of freedom, K=1, NDOF.
- For each degree of freedom:
  a) If NBC(I,K) is zero, continue to the other degrees of freedom.
  b) If NBC(I,K) is negative, store the column(NDOF*(2*I-1)+K) of S(L,M) is B(J), except for the entries with an NBC(L,M) value also negative or zero.
  c) If NBC(I,K) is positive, store the first part of the column in A(J), except for those entries with an NBC(L,M) value negative or zero. Only the first part of the column (up to the diagonal term) is stored since the matrix is symmetric.

The way each entry of [K] is located in A(I) is illustrated in Figure II.9. For a given address (M,J) in [K], the number (JDIAG(J)-J) is calculated. This number gives the positions in A(I) in which the first entry of the column would be located if it were different from zero. The proper position is obtained by adding the row number M.

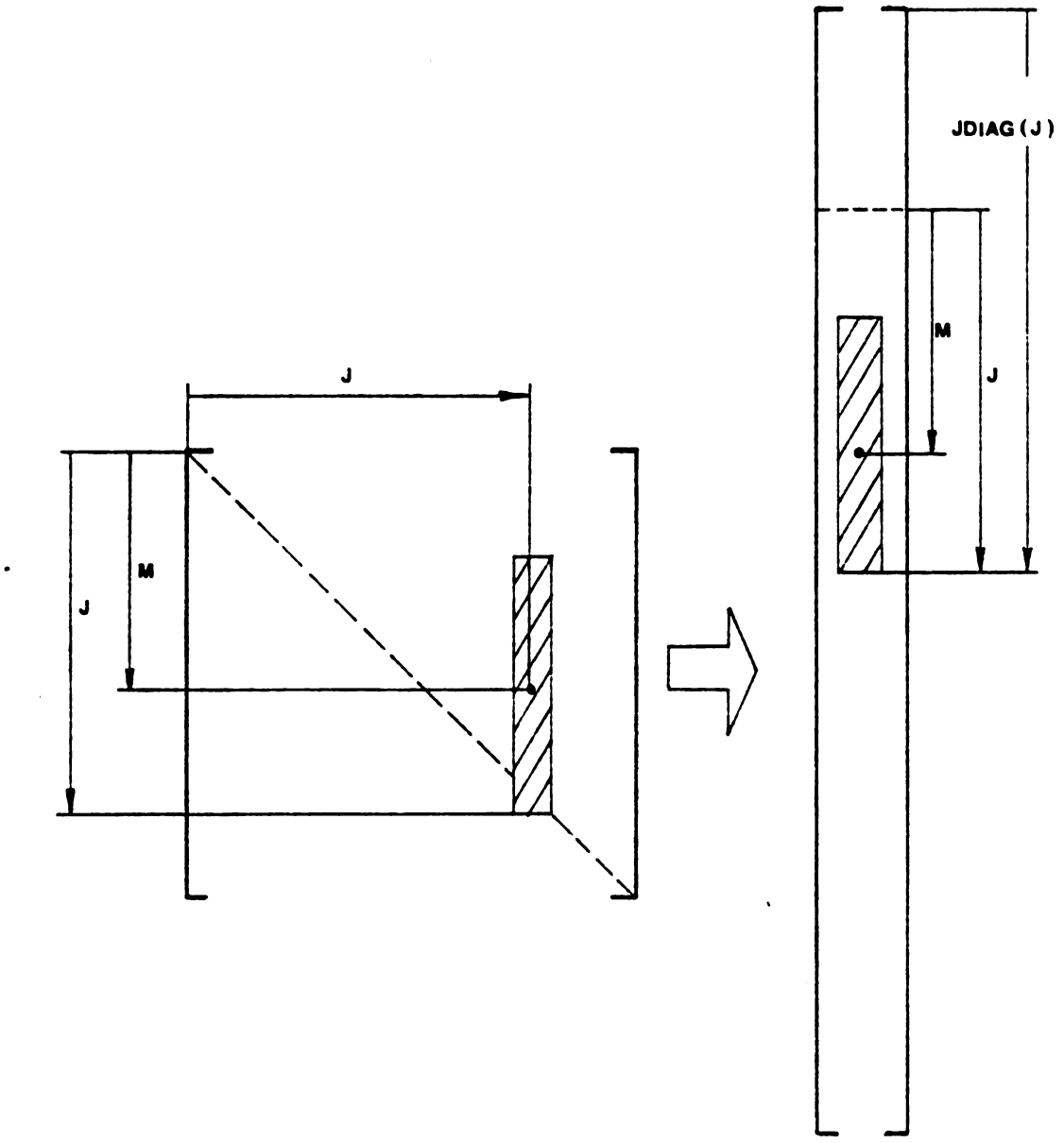


Figure II.9. Location of elements of [K] in A(I)

The location procedure in B(I) is illustrated in Figure II.10. For each NBC(L,N) negative, the position of the first element of the column J is located in B(I), this position being given by the number IDIAG(2*(J-1)+1). Next, a number of positions equal to the difference between the equation number M, and the equation number corresponding to the first non-zero element in the column IDIAG(2*J-1) must be added to set the correct location for the entry.

II.3.4 SUBROUTINE SOLVE

This subroutine uses a variation of the Gauss elimination method called Crout algorithm. This algorithm is thoroughly discussed by R.L. Taylor in [3]. In this section, the important facts of the method and the subroutine are summarized.

The first part of SOLVE, which works when the logical arguments AFAC and BACK are true and false respectively, factors the stiffness matrix $[K]_{11}$ stored in A(I) into the form

$$[K]_{11} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ l_{21} & 1 & \cdots & 0 \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & 1\ 0 \\ l_{n1} & l_{n2} & \cdots & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \\ 0 & & & 0\ u_{nn} \end{bmatrix} = [L][U]$$

The procedure which calculates the entries of [L] and [U] consists of performing the matrix multiplication and equating the respective terms. Following this procedure, it is found that

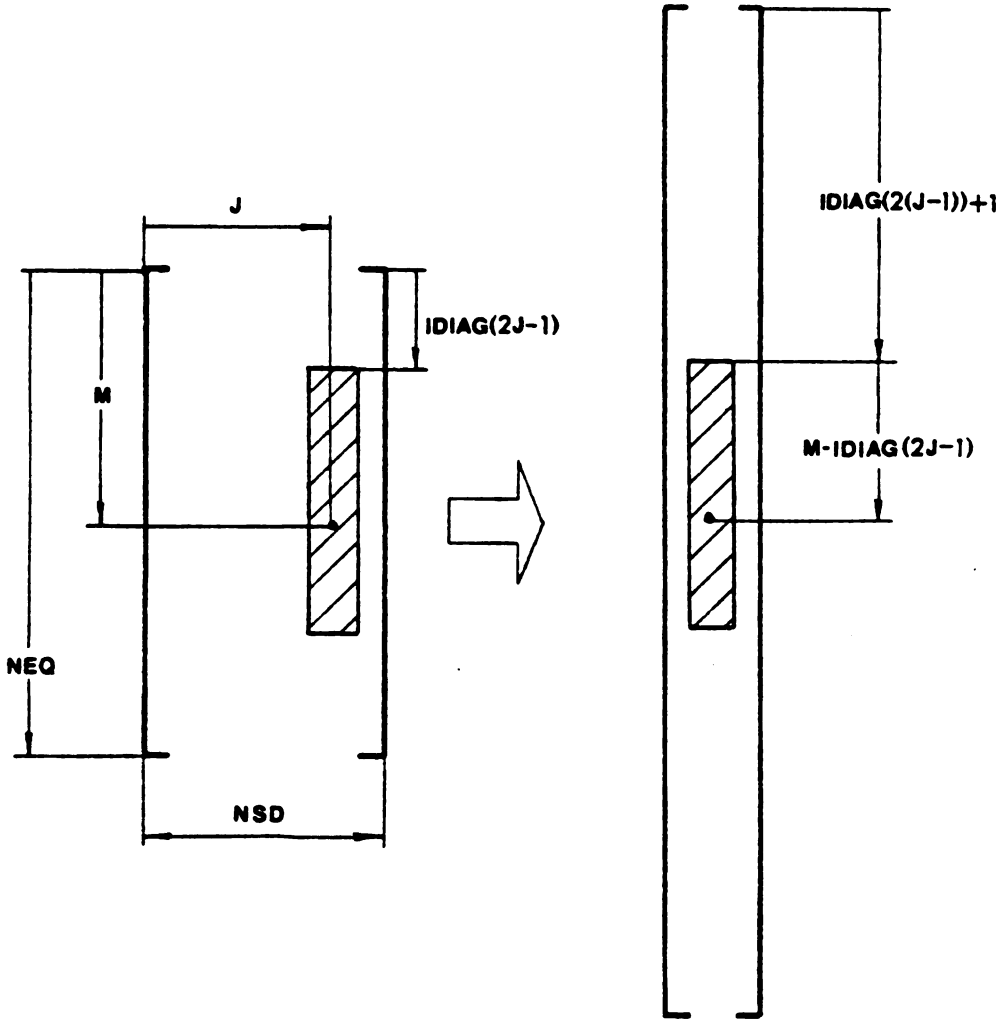


Figure II.10. Location of elements of $[K]_{12}$ in B(I)

$$u_{1n}=k_{1n}$$

$$l_{n1}=k_{n1}/u_{11}$$

$$u_{ij}=k_{ij}-\sum_{m=1}^{i-1} l_{im}u_{mj},\quad i\neq 1$$

$$l_{ij}=\frac{k_{ij}-\sum_{m=1}^{j-1} l_{im}u_{mj}}{ujj}\quad (i\neq j, j\neq 1)$$

$$l_{ij}=u_{ji}/u_{jj},\ \text{if } [K]_{11} \text{ is symmetric.}$$

In the subroutine, the sequence in which the entries are calculated and saved is shown in the example of Figure II.11.

It can be shown that both [L] and [U] have the same profile as $[K]_{11}$. If $k_{1n}=0$, $u_{1n}=0$; if $k_{2n}=0$, $u_{2n}=-l_{21}u_{1n}=0$ also; and, in general, if $k_{in}=0$, $u_{in}=0$ since all the elements u_{mn} of the same column, above u_{in} are 0.

Since $[K]_{11}$ is stored in A(I) by columns, addresses must be found using the array JDIAG(I). Also, since only the profile part of $[K]_{11}$ is saved, some attention must be paid to the performance of the operations, i.e. the summation

$$\sum_{m=1}^{j-1} l_{im}u_{mj}$$

is on the elements of column i and j so that, to do this operation, it is necessary to know which is the shortest column. This is done with the function MINO, which finds the minor value between two integers.

$$\begin{bmatrix} k_{11} & k_{12} & k_{13} \\ & k_{22} & k_{23} \\ & & k_{33} \end{bmatrix} \xrightarrow{u_{11}=k_{11}} \begin{bmatrix} u_{11} & k_{12} & k_{13} \\ & k_{22} & k_{23} \\ & & k_{33} \end{bmatrix} \xrightarrow{u_{12}=k_{12}} \begin{bmatrix} u_{11} & u_{12} & k_{13} \\ & k_{22} & k_{23} \\ & & k_{33} \end{bmatrix}$$

$$\xrightarrow[u_{22}=k_{22}-l_{21}u_{12}]{l_{21}=u_{12}\div u_{11}} \begin{bmatrix} u_{11} & l_{21} & k_{13} \\ & u_{22} & k_{23} \\ & & k_{33} \end{bmatrix} \xrightarrow[u_{23}=k_{23}-l_{21}u_{13}]{u_{13}=k_{13}} \begin{bmatrix} u_{11} & l_{21} & u_{13} \\ & u_{22} & u_{23} \\ & & k_{33} \end{bmatrix}$$

$$\xrightarrow[u_{33}=k_{33}-(l_{31}u_{13}+l_{32}u_{23})]{\substack{l_{31}=u_{13}\div u_{11} \\ l_{32}=u_{23}\div u_{22}}} \begin{bmatrix} u_{11} & l_{21} & l_{31} \\ & u_{22} & l_{32} \\ & & u_{33} \end{bmatrix}$$

Figure II.11. Illustration of the first part of SOLVE

A summary of the algorithm for the first part of SOLVE is as follows,

- Do a loop on the columns.
- If the height of the column equals 1, continue with other columns.
- If the height of the column equals 2, calculate the diagonal element.
- If the height is greater than two, calculate

$$u_{ij}=k_{ij}-\sum_{m=1}^{i-1} l_{im}u_{mj}$$

from J=2 through the diagonal.

- On the diagonal, calculate

$$l_{ji}=u_{ij}/u_{ii} \quad , \quad u_{jj}=k_{jj}-\sum_{m=1}^{j-1} l_{jm}u_{mj}$$

A flow chart of this algorithm is shown in Fig. II.12.

The second part of SOLVE, which works when the arguments AFAC and BACK are false and true respectively, performs two operations. It solves the systems $[L]\{y\}=\{F\}^*$ and $[U]\{x\}=\{y\}$ by forward and backsubstitution respectively. In these systems, $\{F\}^*$ is the force vector formed in MODIFY, $\{x\}$ is the vector of unknown displacements, and $\{y\}$ is an intermediate vector.

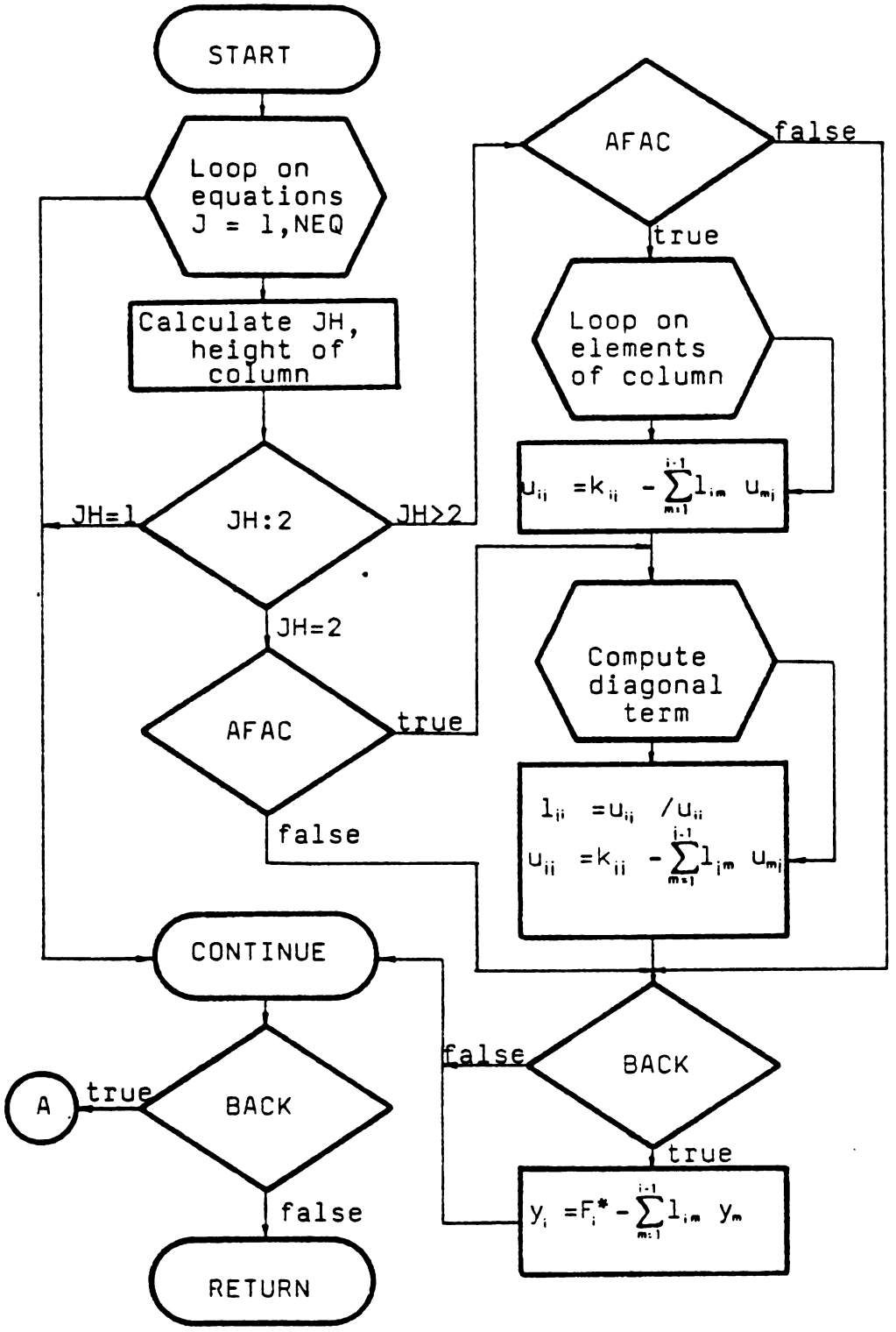


Figure II.12. Flowchart for first part of SOLVE

For a 3x3 matrix the first operation is as follows,

$$\begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix} \begin{Bmatrix} y_1 \\ y_2 \\ y_3 \end{Bmatrix} = \begin{Bmatrix} F_1^* \\ F_2^* \\ F_3^* \end{Bmatrix}, \qquad \begin{aligned} y_1 &= F_1^* \\ y_2 &= F_2^* - l_{21} y_1 \\ y_3 &= F_3^* - (l_{31} y_1 + l_{32} y_2). \end{aligned}$$

In general

$$y_i = F_i^* - \sum_{m=1}^{i-1} l_{im} y_m$$

The elements of {y} are saved in the same vector {F} .

For the same 3x3 matrix, the back substitution is

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} y_1 \\ y_2 \\ y_3 \end{Bmatrix}, \qquad \begin{aligned} x_3 &= y_3/u_{33} \\ x_2 &= (y_2 - u_{23} x_3)/u_{22} \\ x_1 &= (y_1 - (u_{12} x_2 + u_{13} x_3))/u_{11}. \end{aligned}$$

In general

$$x_i = \frac{y_i - \sum_{m=NEQ}^{i+1} u_{im} x_m}{u_{ii}} = y_i/u_{ii} - \sum_{m=NEQ}^{i+1} (u_{im}/u_{ii}) x_m$$

or

$$x_i = v_i/u_{ii} - \sum_{m=NEQ}^{i+1} l_{mi}x_m$$

Note that the values of x_i are in terms of u_{ii} and $l_{mi}(m\neq i)$, so that it is not necessary to calculate u_{mi} for $m\neq i$. A flow chart of this procedure is shown in Fig. II.13.

II.3.5 SUBROUTINE MODIFY

This subroutine has two parts. The first part reads and prints specified non-zero loads or displacements, and the second part modifies the load vector if there are specified non-zero displacements.

In the first part the subroutine reads (in free format) the node number, the direction and the value of each non-zero force or displacement. The loads are stored in the first NEQ positions of F(I), and the displacements in the remaining NSD positions of F(I). No ordering is necessary to specify this data. The subroutine stops reading when a zero value is given for the node number.

The second part of MODIFY performs the operation

$$\{F\}_s^* = \{F\}_s - [K]_{12}\{u\}_s \qquad \text{(II.16)}$$

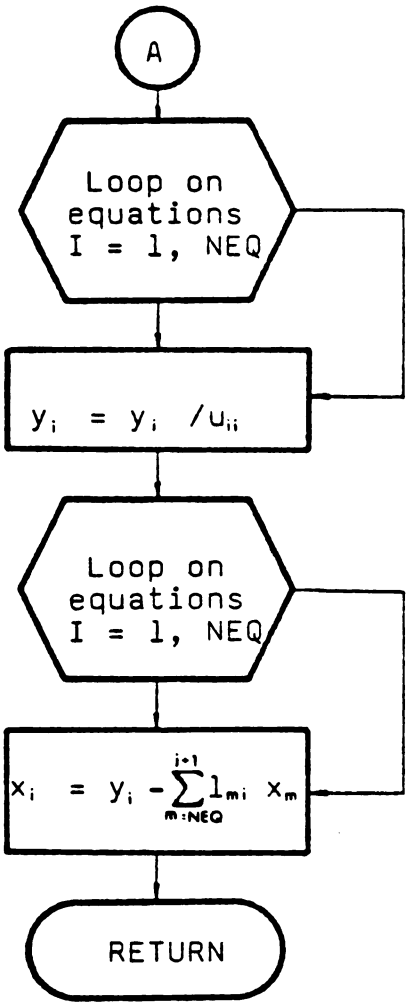


Figure II.13. Flowchart for second part of SOLVE

where the dimensions of the matrix $[K]_{12}$, stored in the array B(J), are NEQxNSD. The algorithm to do this modification is as follows.

- Do a loop on the equations.
- Do a loop on the specified non-zero displacements.
- Locate each element in B(J), and perform the product.

The array IDIAG(I) is used to locate positions in B(J).

II.3.6 SUBROUTINE STRESS

This subroutine calculates and prints the stresses in each element and the reactions at the points of specified displacement.

The algorithm of the subroutine is as follows.

- Do a loop on the elements.
- For each element, localize D(K),XE(I,K).
- Call SHAPE to calculate the global derivatives of the shape functions.
- Compute stresses following eq. (II.3).

In order to organize the reactions, changes are made in some positions of the array NTBC(I,K). A negative value is assigned at those positions where NTBC(I,K) is equal to zero (corresponding to a specified zero displacement) starting from -NSD-1. Thus, for each negative NTBC(I,K), a reaction is computed. This change in NTBC(I,K) is actually made in PFORM, and it permits organization of the reactions in the last part of the array F(I).

Each reaction is obtained by calculating and adding elemental nodal forces at each location in which NTBC(I,K) is negative. This procedure requires knowledge of the stiffness matrices of those elements adjacent to the reactions. These matrices can be saved in disk storage during the execution of PFORM, or can be calculated again, as is done in the program. Although this procedure requires a larger number of operations, considerable storage is saved in central memory since only a reduced part of the global stiffness matrix needs to be saved. This is an important feature when treating large problems.

II.3.7 EXAMPLE OF INPUT DATA

For the example of Figure II.14, the input data is as follows,

```
(TITLE, 1-80)
     EXAMPLE OF INPUT DATA, FEM
(NMAT,NEL,NNO,NDOF,NLO,Free Format)
     1    4    9    2    1
(E(1),PR(1),E20.12,F20.12)
2.5E00                              0.25
(NMAT(I),NEN(I),(NN(I,K),K=1,NEN(I)),Free Format)
     1    4    1    4    5    2
     1    4    2    5    6    3
     1    4    4    7    8    5
     1    4    5    8    9    6

(XE(I,1),NTBC(I,1),XE(I,2),NTBC(I,2),F15.5,I5,F15.5,I5)
     0.0                 -1   0.0            -1
     0.0                 -1   1.0
     0.0                 -1   2.0
     2.0                      0.0
     2.0                      1.0
     2.0                      2.0
```

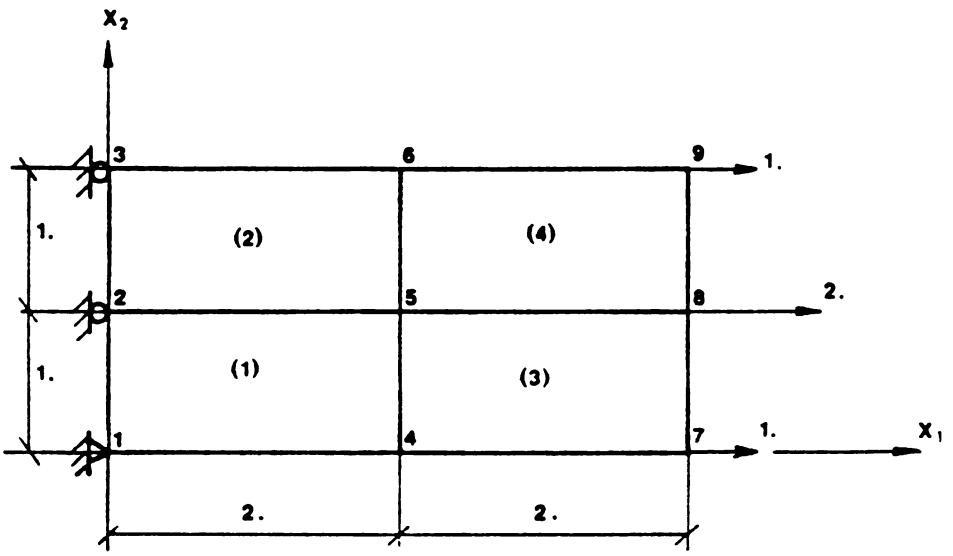


Figure II.14. Example problem to demonstrate input data for FEM

```
4.0                         0.0
4.0                         1.0
4.0                         2.0

(LL, Free Format)

3

(N,I,F(2*(N-1)+I, Free Format)

7    1    1.0
8    1    2.0
9    1    1.0
0    0    0.0

(R, Free Format)

0.5

(NE1,NE2, Free Format)

1    1
3    4
0    0
```

Note that stresses are to be calculated in elements 1,3 and 4 at four points defined by the local coordinate R=0.5. For the element 1 those points are located in (0.5,0.25), (1.5, 0.25), (1.5, 0.75) and (0.5, 0.75).

CHAPTER III

BOUNDARY ELEMENT MODELLING

III.1 DESCRIPTION OF THE BOUNDARY ELEMENT METHOD

For the plane boundary-value problem of linear elasticity illustrated in Figure III.1, the displacement at a point x on B is related to the displacements and tractions at all other points on B by Somigliana's identity, i.e.

$$\alpha_{ij}(x)u_j(x)+\int_B (UC)_{i.j}(x,\bar{x})u_j(\bar{x})d\bar{s}=\int_B (UR)_{i.j}(x,\bar{x})t_j(\bar{x})d\bar{s} \qquad \text{(III.1)}$$

where the integral on the left hand side is interpreted in the Cauchy principal-value sense. The function $(UC)_{i.j}(x,\bar{x})$ is the displacement, $u_i(x)$, due to a unit displacement discontinuity, $C_j(\bar{x})$, applied in the infinite elastic plane and $(UR)_{i.j}(x,\bar{x})$ is the displacement, $u_i(x)$ due to a unit force, $R_j(\bar{x})$, applied in the infinite elastic plane.
The coefficients $\alpha_{ij}(x)$ are defined as follows for plane stress:

$$\alpha_{ij}(x)= \begin{cases} \omega/2\pi+(1+\upsilon)(\sin 2(\alpha+\omega)-\sin 2\alpha)/8\pi, & i=j \\ (1+\upsilon)(\sin^2(\alpha+\omega)-\sin^2\alpha)/4\pi, & i\neq j \end{cases} \qquad \text{(III.2)}$$

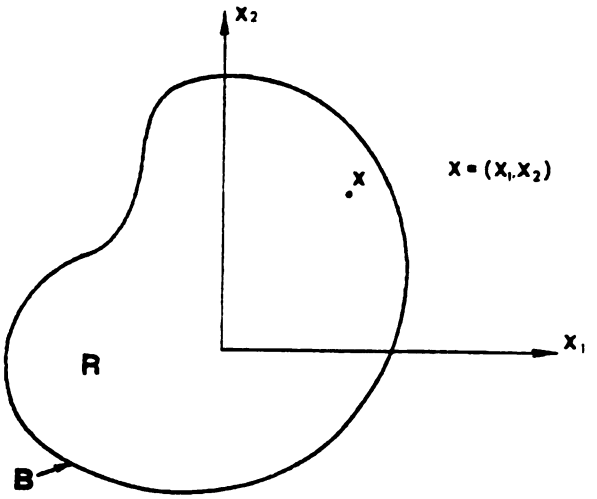


Figure III.1. Plane boundary value problem

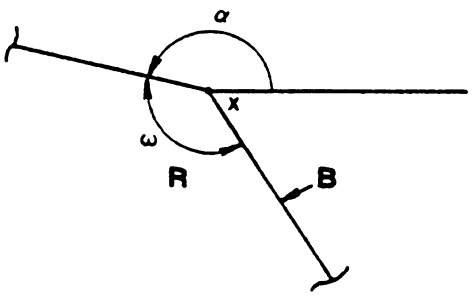


Figure III.2. Angles α and ω of eq. (III.2)

where α and ω are shown in Figure III.2. For a smooth boundary at x, $\omega=\pi$ and $\alpha_{ij}=\frac{1}{2}\,\delta_{ij}$.

At a point x in R, the displacements and stresses can be calculated from the equations

$$u_i(x)=\int_B (UR)_{i.j}(x,\bar{x})\,t_j(\bar{x})\,d\bar{s}-\int_B (UC)_{i.j}(x,\bar{x})\,u_j(\bar{x})\,d\bar{s}$$

$$\sigma_{ik}(x)=\int_B (\sigma R)_{ik.j}(x,\bar{x})\,t_j(\bar{x})\,d\bar{s}-\int_B (\sigma C)_{ik.j}(x,\bar{x})\,u_j(\bar{x})\,d\bar{s} \qquad \text{(III.3)}$$

where the influence functions $(\sigma R)_{ik.j}(x,\bar{x})$ and $(\sigma C)_{ik.j}(x,\bar{x})$ give the stress component $\sigma_{ik}(x)$ due to a unit force, $R_j(\bar{x})$, and a unit displacement discontinuity, $C_j(\bar{x})$, respectively, applied in the infinite plane.

At each point x on B and in each direction, either $u_j(x)$ or $t_j(x)$ is known. Therefore, eq. (III.1) can be used to solve for the unknown values of $u_j(x)$ and $t_j(x)$, thus giving complete boundary information. The displacements and stresses at any internal point can then be determined by integration using eq. (III.3).

It can be shown that, for plane stress the influence funtions of eq. (III.1) and (III.3) are given by

$$(UR)_{i.k}=[-(3-\upsilon)\delta_{ik}\log\rho+(1+\upsilon)q_iq_k]/(8\pi G)$$

$$(UC)_{1.1}=[2(1+\upsilon)(\bar{n}_1q_1^3-\bar{n}_2q_2^3)+(1-\upsilon)\bar{n}_1q_1+(3+\upsilon)\bar{n}_2q_2]/(4\pi\rho)$$

$$(UC)_{1.2}=[2(1+\upsilon)(-\bar{n}_2q_1^3-\bar{n}_1q_2^3)+(1+3\upsilon)\bar{n}_2q_1+(3+\upsilon)\bar{n}_1q_2]/(4\pi\rho)$$

$$(UC)_{2.1}=[2(1+\upsilon)(-\bar{n}_2 q_1^3-\bar{n}_1 q_2^3)+(3+\upsilon)\bar{n}_2 q_1+(1+3\upsilon)\bar{n}_1 q_2]/(4\pi\rho)$$

$$(UC)_{2.2}=[2(1+\upsilon)(-\bar{n}_1 q_1^3+\bar{n}_2 q_2^3)+(1-\upsilon)\bar{n}_2 q_2+(3+\upsilon)\bar{n}_1 q_1]/(4\pi\rho)$$

$$(\sigma R)_{11.1}=[-2(1+\upsilon)q_1^3-(1-\upsilon)q_1]/(4\pi\rho)$$

$$(\sigma R)_{12.1}=[2(1+\upsilon)q_2^3-(3+\upsilon)q_2]/(4\pi\rho)$$

$$(\sigma R)_{22.1}=[2(1+\upsilon)q_1^3-(1+3\upsilon)q_1]/(4\pi\rho)$$

$$(\sigma R)_{11.2}=[2(1+\upsilon)q_2^3-(1+3\upsilon)q_2]/(4\pi\rho)$$

$$(\sigma R)_{12.2}=[2(1+\upsilon)q_1^3-(3+\upsilon)q_1]/(4\pi\rho)$$

$$(\sigma R)_{22.2}=[-2(1+\upsilon)q_2^3-(1-\upsilon)q_2]/(4\pi\rho)$$

$$(\sigma C)_{11.1}=G(1+\upsilon)[(1+4q_1^2-8q_1^4)\bar{n}_1+2q_1q_2(1-4q_1^2)\bar{n}_2]/(2\pi\rho^2)$$

$$(\sigma C)_{12.1}=G(1+\upsilon)[(1-8q_1^2q_2^2)\bar{n}_2+2q_1q_2(1-4q_1^2)\bar{n}_1]/(2\pi\rho^2)$$

$$(\sigma C)_{22.1}=G(1+\upsilon)[(1-8q_1^2q_2^2)\bar{n}_1+2q_1q_2(1-4q_2^2)\bar{n}_2]/(2\pi\rho^2)$$

$$(\sigma C)_{11.2}=(\sigma C)_{12.1}$$

$$(\sigma C)_{12.2}=(\sigma C)_{22.1}$$

$$(\sigma C)_{22.2}=G(1+\upsilon)[(1+4q_2^2-8q_2^4)\bar{n}_2+2q_1q_2(1-4q_2^2)\bar{n}_1]/(2\pi\rho^2) \qquad \text{(III.4)}$$

where $$\rho=[(x_1-\bar{x}_1)^2+(x_2-\bar{x}_2)^2]^{\frac{1}{2}}$$

$$q_1=(x_1-\bar{x}_1)/\rho \qquad q_2=(x_1-\bar{x}_2)/\rho \qquad \text{(III.5)}$$

and $\bar{n}_1,\bar{n}_2$ are the components of the normal unit vector at a point on the boundary.

III.2 PIECEWISE LINEAR ELEMENTS

Eq. (III.1) can be solved numerically if the boundary B is approximated by N straight segments, as shown in Figure III.3.

For this model, eq. (III.1) can be written as

$$\alpha_{ij}(x^{(n)})u_j(x^{(n)}) + \sum_{m=1}^{N} \int_m (UC)_{i.j}(x^{(n)},\bar{x})u_j(\bar{x})d\bar{s}$$

$$= \sum_{m=1}^{N} \int_m (UR)_{i.j}(x^{(n)},\bar{x})t_j(\bar{x})d\bar{s} \qquad \text{(III.6)}$$

The displacements and tractions in each segment m can be approximated using shape functions so that

$$u_j(\bar{x}) = u_j^{(m-1)}N_1(\xi) + u_j^{(m)}N_2(\xi)$$

$$t_j(\bar{x}) = t_j^{(m)} \qquad \text{(III.7)}$$

where

$$N_1(\xi) = \tfrac{1}{2}(1-\xi) \quad , \quad N_2(\xi) = \tfrac{1}{2}(1+\xi)$$

$$\bar{x} = N_1(\xi)x^{(m-1)} + N_2(\xi)x^{(m)}$$

$$d\bar{s} = \tfrac{1}{2}(s_m - s_{m-1})d\xi = \tfrac{1}{2}\Delta s_m d\xi \qquad \text{(III.8)}$$

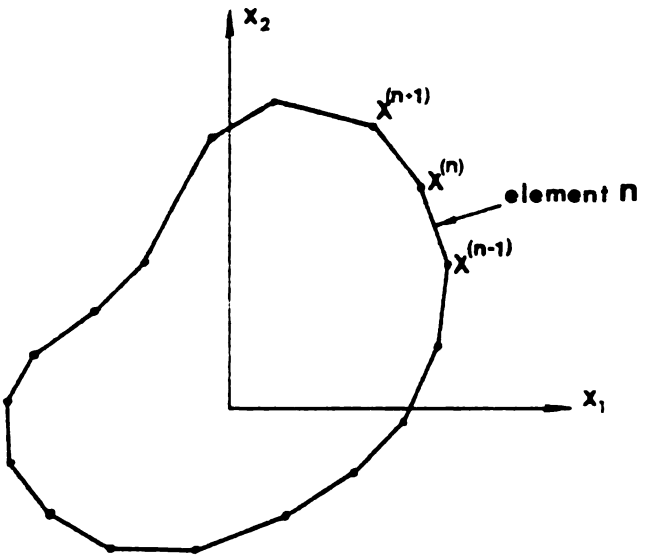


Figure III.3. Boundary discretization

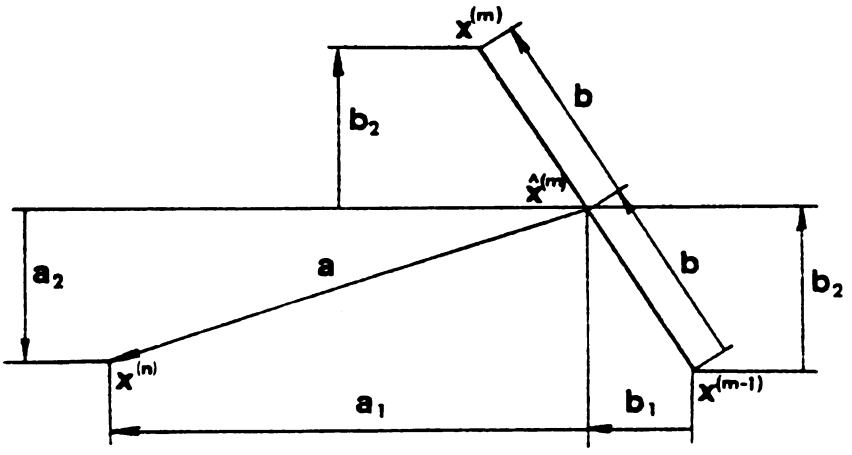


Figure III.4. Definition of a_i, b_i, $\hat{x}^{(m)}$

where ξ is a local coordinate for the segment m with value -1 at node m-1, value 0 at the center of the segment, and value 1 at node m.

Note that the order of differenciation of $t_j(x)$ in the interval is less that that of $u_j(x)$. This model allows discontinuities of $t_j(x)$ on the boundary and it is consistent with the Finite Element triangular element, which has linear displacements and constant tractions on its sides.

If eqs. (III.7) and (III.8) are substituted into eq. (III.6) the following is obtained

$$2\alpha_{ij}(x^{(n)})u_j^{(n)} + \sum_{m=1}^{N} \Delta s_m [\int_m (UC)_{i.j}(x^{(n)},\xi)N_1(\xi)d\xi\, u_j^{(m-1)}$$

$$+\int_m (UC)_{i.j}(x^{(n)},\xi)N_2(\xi)d\xi\, u_j^{(m)}] = \sum_{m=1}^{N} \int_m (UR)_{i.j}(x^{(n)},\xi)d\xi \Delta s_m t_j^{(m)}$$

or

$$2\alpha_{ij}^{(n)}u_j^{(n)} + \sum_{m=1}^{N} [A_{i.j}^{(m.n)}u_j^{(m-1)} + B_{i.j}^{(m.n)}u_j^{(m)}] = [1/G] \sum_{m=1}^{N} C_{i.j}^{(m.n)}\hat{F}_j^{(m)}$$

$$n=1,.....,N \qquad \text{(III.9)}$$

where

$$\alpha_{ij}^{(n)} = \alpha_{ij}(x^{(n)})$$

$$u_j^{(n)} = u_j(x^{(n)})$$

$$A_{ij}^{(m.n)} = \Delta s_m \int_m (UC)_{i.j}(x^{(n)},\xi)N_1(\xi)d\xi$$

$$B_{ij}^{(m.n)} = \Delta s_m \int_m (UC)_{i.j}(x^{(n)},\xi) N_2(\xi) d\xi$$

$$C_{ij}^{(m.n)} = G \int_m (UR)_{i.j}(x^{(n)},\xi) d\xi$$

$$\hat{F}_j^{(m)} = \Delta s_m t_j^{(m)} \qquad \text{(III.10)}$$

Note that the functions $(UC)_{i.j}$, $(UR)_{i.j}$ are singular when the point of coordinate x approaches the node n. This is the case when the element m ends or begins with the node n, i.e. when m=n, m=n+1 (n=m,m-1). The resulting singular integrals for these positions must be calculated analytically. A summary of those calculations is shown in Table III.1, where b_i is defined in Figure (III.4). Note that the terms $A_{1.2}^{(n+1.n)}$ and $B_{1.2}^{(n.n)}$, which contribute to the coefficient of $u_2^{(n)}$, produce a net result

$$A_{1.2}^{(n+1.n)} + B_{1.2}^{(n.n)} = (1-\upsilon)\log(\Delta s_n/\Delta s_{n+1})/(2\pi)$$

The same applies for $A_{2.1}^{(n+1.n)} + B_{2.1}^{(n.n)}$.

The nonsingular integrals are evaluated by numerical integration using the Gauss quadrature formula. If it is defined that

$$\hat{x}^{(m)} = \tfrac{1}{2}[x^{(m)} + x^{(m-1)}]$$

$$a_i = x_i^{(n)} - \hat{x}_i^{(n)} \qquad b_i = x_i^{(m)} - \hat{x}_i^{(m)}$$

$$R = a_i a_i - 2a_i b_i \xi + b_i b_i \xi^2 \qquad \text{(III.11)}$$

Table III.1 Singular Integrals

n:node number m:segment number

Condition	Coefficients	Value	
n=m-1,m	$C_{i.j}^{(m.n)}$	$[(3-\upsilon)(1-\log(2b))\delta_{ij}+(1+\upsilon)b_i b_j/b^2]/(4\pi)$	1
n=m-1,m	$A_{1.1}^{(m.n)}=B_{1.1}^{(m.n)}$ $A_{2.2}^{(m.n)}=B_{2.2}^{(m.n)}$	0	2
n=m	$A_{1.2}^{(m.n)}=-A_{2.1}^{(m.n)}$	$(1-\upsilon)/(2\pi)$	3
n=m-1	$B_{1.2}^{(m.n)}=-B_{2.1}^{(m.n)}$	$-(1-\upsilon)/(2\pi)$	4
n=m-1	$A_{1.2}^{(m.n)}=-A_{2.1}^{(m.n)}$	$(1-\upsilon)[2+2\lim_{\varepsilon\to 0}[\log(\varepsilon/\Delta s_m)]]/(4\pi)$	5
n=m	$B_{1.2}^{(m.n)}=-B_{2.1}^{(m.n)}$	$(1-\upsilon)[-2-2\lim_{\varepsilon\to 0}[\log(\varepsilon/\Delta s_m)]]/(4\pi)$	6
Net contribution of 5 and 6	$A_{1.2}^{(n+1.n)}+B_{1.2}^{(n.n)}$ $A_{2.1}^{(n+1.n)}+B_{2.1}^{(n.n)}$	$(1-\upsilon)\log(\Delta s_n/\Delta s_{n+1})/(2\pi)$	7

as shown in Figure III.4, the nonsingular integrals can be calculated for $m \neq n, n+1$

$$\left.\begin{matrix} A_{1.1}^{(m.n)} \\ B_{1.1}^{(m.n)} \end{matrix}\right\} = [2b_2(1+\upsilon)\int_{-1}^{1}(1\mp\xi)[(a_1-b_1\xi)^3/R^2]d\xi + 2b_1(1+\upsilon)\int_{-1}^{1}(1\mp\xi)[(a_2-b_2\xi)^3/R^2]d\xi$$

$$+b_2(1-\upsilon)\int_{-1}^{1}(1\mp\xi)[(a_1-b_1\xi)/R]d\xi - b_1(3+\upsilon)\int_{-1}^{1}(1\mp\xi)[(a_2-b_2\xi)/R]d\xi]/(4\pi)$$

$$\left.\begin{matrix} A_{1.2}^{(m.n)} \\ B_{1.2}^{(m.n)} \end{matrix}\right\} = [2b_1(1+\upsilon)\int_{-1}^{1}(1\mp\xi)[(a_1-b_1\xi)^3/R^2]d\xi - 2b_2(1+\upsilon)\int_{-1}^{1}(1\mp\xi)[(a_2-b_2\xi)^3/R^2]d\xi$$

$$-b_1(1+3\upsilon)\int_{-1}^{1}(1\mp\xi)[(a_1-b_1\xi)/R]d\xi + b_2(3+\upsilon)\int_{-1}^{1}(1\mp\xi)[(a_2-b_2\xi)/R]d\xi]/(4\pi)$$

$$\left.\begin{matrix} A_{2.1}^{(m.n)} \\ B_{2.1}^{(m.n)} \end{matrix}\right\} = [2b_1(1+\upsilon)\int_{-1}^{1}(1\mp\xi)[(a_1-b_1\xi)^3/R^2]d\xi - 2b_2(1+\upsilon)\int_{-1}^{1}(1\mp\xi)[(a_2-b_2\xi)^3/R^2]d\xi$$

$$-b_1(3+\upsilon)\int_{-1}^{1}(1\mp\xi)[(a_1-b_1\xi)/R]d\xi + b_2(1+3\upsilon)\int_{-1}^{1}(1\mp\xi)[(a_2-b_2\xi)/R]d\xi]/(4\pi)$$

$$\left.\begin{matrix} A_{2.2}^{(m.n)} \\ B_{2.2}^{(m.n)} \end{matrix}\right\} = [-2b_2(1+\upsilon)\int_{-1}^{1}(1\mp\xi)[(a_1-b_1\xi)^3/R^2]d\xi - 2b_1(1+\upsilon)\int_{-1}^{1}(1\mp\xi)[(a_2-b_2\xi)^3/R^2]d\xi$$

$$+b_2(3+\upsilon)\int_{-1}^{1}(1\mp\xi)[(a_1-b_1\xi)/R]d\xi - b_1(1-\upsilon)\int_{-1}^{1}(1\mp\xi)[(a_2-b_2\xi)/R]d\xi]/(4\pi)$$

$$C_{1.1}^{(m.n)} = [-\tfrac{1}{2}(3-\upsilon)\int_{-1}^{1}\log(R)d\xi + (1+\upsilon)\int_{-1}^{1}[(a_1-b_1\xi)^2/R]d\xi]/(8\pi)$$

$$C_{1.2}^{(m.n)} = C_{2.1}^{(m.n)} = [(1+\upsilon)\int_{-1}^{1}[(a_1-b_1\xi)(a_2-b_2\xi)/R]d\xi]/(8\pi)$$

$$C_{2.2}^{(m.n)} = [-\tfrac{1}{2}(3-\upsilon)\int_{-1}^{1}\log(R)d\xi + (1+\upsilon)\int_{-1}^{1}[(a_2-b_2\xi)^2/R]d\xi]/(8\pi) \qquad \text{(III.12)}$$

E

T

resul

probl

nodal

relat

$\hat{F}$,

inter

nodal

node,

wher

Eq. (III.9) can be also written in matrix form as

$$[UC]\{Gu\}=[UR]\{\hat{F}\} \tag{III.13}$$

This is a system of equations relating nodal displacements to resultant segment forces. In order to solve a well-posed elasticity problem, it is necessary to re-pose this system of equations in terms of nodal forces. Therefore, a transformation

$$\{F\}=[\Gamma]\{\hat{F}\} \tag{III.14}$$

relating the vector of nodal forces $\{F\}$ to the vector of segment forces $\{\hat{F}\}$, is introduced into the system (III.13). The simplest physical interpretation of the transformation is to replace the segment forces by nodal forces equal to the average of the segment forces adjacent to each node, or

$$F_i^{(n)}=\tfrac{1}{2}[\hat{F}_i^{(n)}+\hat{F}_i^{(n+1)}] \tag{III.15}$$

The form of $[\Gamma]$ for this transformation is

$$[\Gamma]=\tfrac{1}{2}\begin{bmatrix} I & I & 0 & 0..0 \\ 0 & I & I & 0..0 \\ 0 & 0 & I & I..0 \\ 0 & 0 & 0 & I..0 \\ \cdots & \cdots & \cdots & \cdots \\ I & 0 & 0 & 0..I \end{bmatrix} \tag{III.16}$$

where I is a 2x2 identity matrix.

For an odd number of nodes, the inverse of $[\Gamma]$ is

$$[\Gamma]^{-1}=\begin{bmatrix} I & -I & I & -I\ldots I \\ I & I & -I & I\ldots -I \\ -I & I & I & -I\ldots I \\ I & -I & I & I\ldots -I \\ \ldots & \ldots & \ldots & \ldots \\ -I & I & -I & I \quad I \end{bmatrix} \tag{III.17}$$

and eq. (III.13) becomes

$$[UC]\{Gu\}=[UR][\Gamma]^{-1}\{F\}. \tag{III.18}$$

It should be noted that, for an even number of nodes, $[\Gamma]$ has no inverse.

A small perturbation is introduced into the system of equations (III.18) in order to enforce the equilibrium conditions

$$\sum_{i=1}^{N} F_1^{(i)}=0 \qquad \sum_{i=1}^{N} F_2^{(i)}=0 \qquad \sum_{i=1}^{N} (x_1^{(i)}F_2^{(i)}-x_2^{(i)}F_1^{(i)})=0$$

or in matrix form

$$[Q]\{F\}=0 \tag{III.19}$$

where

$$[Q]=\begin{bmatrix} 1 & 0 & 1 & 0 & \ldots. & 1 & 0 \\ 0 & 1 & 0 & 1 & \ldots. & 0 & 1 \\ -x_2^{(1)} & x_1^{(1)} & -x_2^{(2)} & x_1^{(2)} & \ldots. & -x_2^{(N)} & x_1^{(N)} \end{bmatrix}.$$

Coupling eqs. (III.18) and (III.19):

$$\begin{bmatrix} UC \\ 0 \end{bmatrix}\{Gu\}=\begin{bmatrix} UR^* & Q^T \\ Q & 0 \end{bmatrix}\begin{Bmatrix} \{F\} \\ \{\lambda\} \end{Bmatrix} \tag{III.20}$$

where $[UR^*] = [UR][\Gamma]^{-1}$.

In partioned form

$$\begin{aligned} [UC]\{Gu\}&=[UR^*]\{F\}+[Q]^T\{\lambda\} \\ 0&=[Q]\{F\}. \end{aligned}$$

The vector $\{\lambda\}_{(3 \times 1)}$ contains three operators which introduce a perturbation into the system. Note that the higher the number of nodes in the model, the smaller the components of $\{\lambda\}$.

After solving eq. (III.20), stresses and displacements can be calculated anywhere in the body using a discretized form of eq. (III.3) which can be written as

$$\begin{Bmatrix} u_1(x) \\ u_2(x) \end{Bmatrix} = [UR^{(u)}]^*_{(2,2n)}\{F\} - [UC^{(u)}]_{(2,2n)}\{u\}$$

$$\begin{Bmatrix} \sigma_{11}(x) \\ \sigma_{22}(x) \\ \sigma_{12}(x) \end{Bmatrix} = [UR^{(s)}]^*_{(3,2n)}\{F\} - [UC^{(s)}]_{(3,2n)}\{u\} \qquad \text{(III.21)}$$

where

$$[UR^{(u)}]^* = [UR^{(u)}][\Gamma]^{-1}, \quad [UR^{(s)}]^* = [UR^{(s)}][\Gamma]^{-1}.$$

All the entries of the matrices as $[UR^{(u)}]$, $[UC^{(u)}]$, $[UR^{(s)}]$, $[UC^{(s)}]$ are calculated by numerical integration. Eq. (III.12) are used to compute $[UR^{(u)}]$ and $[UC^{(u)}]$. For the entries of $[UR^{(s)}]$ and $[UC^{(s)}]$ the following equations apply

$$(UR^{(s)})_{1,(2n-1)} = [-2(1+\upsilon)\int_{-1}^{1} [(a_1-b_1\xi)^3/R^2]d\xi - (1-\upsilon)\int_{-1}^{1} [(a_1-b_1\xi)/R]d\xi]/(8\pi)$$

$$(UR^{(s)})_{1,(2n)} = [2(1+\upsilon)\int_{-1}^{1} [(a_2-b_2\xi)^3/R^2]d\xi - (1+3\upsilon)\int_{-1}^{1} [(a_2-b_2\xi)/R]d\xi]/(8\pi)$$

$$(UR^{(s)})_{2,(2n-1)}=[+2(1+\upsilon)\int_{-1}^{1}[(a_1-b_1\xi)^3/R^2]d\xi-(1+3\upsilon)\int_{-1}^{1}[(a_1-b_1\xi)/R]d\xi]/(8\pi)$$

$$(UR^{(s)})_{2,(2n)}=[-2(1+\upsilon)\int_{-1}^{1}[(a_2-b_2\xi)^3/R^2]d\xi-(1-\upsilon)\int_{-1}^{1}[(a_2-b_2\xi)/R]d\xi]/(8\pi)$$

$$(UR^{(s)})_{3,(2n-1)}=[2(1+\upsilon)\int_{-1}^{1}[(a_2-b_2\xi)^3/R^2]d\xi-(3+\upsilon)\int_{-1}^{1}[(a_2-b_2\xi)/R]d\xi]/(8\pi)$$

$$(UR^{(s)})_{3,(2n)}=[2(1+\upsilon)\int_{-1}^{1}[(a_1-b_1\xi)^3/R^2]d\xi-(3+\upsilon)\int_{-1}^{1}[(a_1-b_1\xi)/R]d\xi]/(8\pi)$$

$$\left.\begin{aligned}(UC^{(s)})_{1,(2n-3)}&=G(1+\upsilon)[2b_2[\int_{-1}^{1}[(1\mp\xi)/R]d\xi+4\int_{-1}^{1}(1\mp\xi)[(a_1-b_1\xi)^2/R^2]d\xi\\&\qquad -8\int_{-1}^{1}(1\mp\xi)[(a_1-b_1\xi)^4/R^3]d\xi]\\(UC^{(s)})_{1,(2n-1)}&\quad -4b_1[\int_{-1}^{1}(1\mp\xi)[(a_1-b_1\xi)(a_2-b_2\xi)/R^2][1-4(a_1-b_1\xi)^2/R]d\xi]]/(8\pi)\end{aligned}\right\}$$

$$\left.\begin{aligned}(UC^{(s)})_{1,(2n-2)}&=G(1+\upsilon)[-2b_1\int_{-1}^{1}[(1\mp\xi)/R][1-8(a_1-b_1\xi)^2(a_2-b_2\xi)^2/R^2]d\xi\\(UC^{(s)})_{1,(2n)}&\quad +4b_2\int_{-1}^{1}(1\mp\xi)[(a_1-b_1\xi)(a_2-b_2\xi)/R^2][1-4(a_1-b_1\xi)^2/R]d\xi]/(8\pi)\end{aligned}\right\}$$

$$\left.\begin{aligned}(UC^{(s)})_{2,(2n-3)}&=G(1+\upsilon)[+2b_2\int_{-1}^{1}[(1\mp\xi)/R][1-8(a_1-b_1\xi)^2(a_2-b_2\xi)^2/R^2]d\xi\\(UC^{(s)})_{2,(2n-1)}&\quad -4b_1\int_{-1}^{1}(1\mp\xi)[(a_1-b_1\xi)(a_2-b_2\xi)/R^2][1-4(a_2-b_2\xi)^2/R]d\xi]/(8\pi)\end{aligned}\right\}$$

$$\left.\begin{aligned}(UC^{(s)})_{1,(2n-2)}&=G(1+\upsilon)[-2b_1[\int_{-1}^{1}[(1\mp\xi)/R]d\xi+4\int_{-1}^{1}(1\mp\xi)[(a_2-b_2\xi)^2/R^2]d\xi\\&\qquad -8\int_{-1}^{1}(1\mp\xi)[(a_2-b_2\xi)^4/R^3]d\xi]\\(UC^{(s)})_{1,(2n)}&\quad +4b_2[\int_{-1}^{1}(1\mp\xi)[(a_1-b_1\xi)(a_2-b_2\xi)/R^2][1-4(a_2-b_2\xi)^2/R]d\xi]]/(8\pi)\end{aligned}\right\}$$

$$(UC^{(s)})_{3,(2n-3)}=(UC^{(s)})_{1,(2n-2)} \qquad (UC^{(s)})_{3,(2n-1)}=(UC^{(s)})_{1,(2n)}$$

$$(UC^{(s)})_{3,(2n-2)}=(UC^{(s)})_{2,(2n-3)} \qquad (UC^{(s)})_{3,(2n)}=(UC^{(s)})_{2,(2n-1)} \qquad \text{(III.22)}$$

where a_i, b_i and R are defined by eq. (III.11).

III.3 PLANE STRESS BOUNDARY ELEMENT PROGRAM

The boundary element program described in this section can solve plane stress elastostatics problems using linear interpolation for the displacements and constant segment tractions.

The flow chart for the program is shown in Figure (III.5). In the first part, the data of the problem is read. Then the weights and points for the numerical integration of the non-singular integrals are assigned to the arrays W(L) and R(L). Next, the augmented matrices [UR] and [UC] are initialized. In part 4 of the program, the entries of the matrices [UC] and [UR] are calculated. In part 5, the operation [UR] $[\Gamma]^{-1}$ is performed, and in part 6, the system of equations is rearranged in order to have all the unknowns on the same side of the system of equations. Subroutine GAUSS solves the equations, and the unknown are printed in part 8 of the program. Finally, stresses and displacements are calculated for the field points.

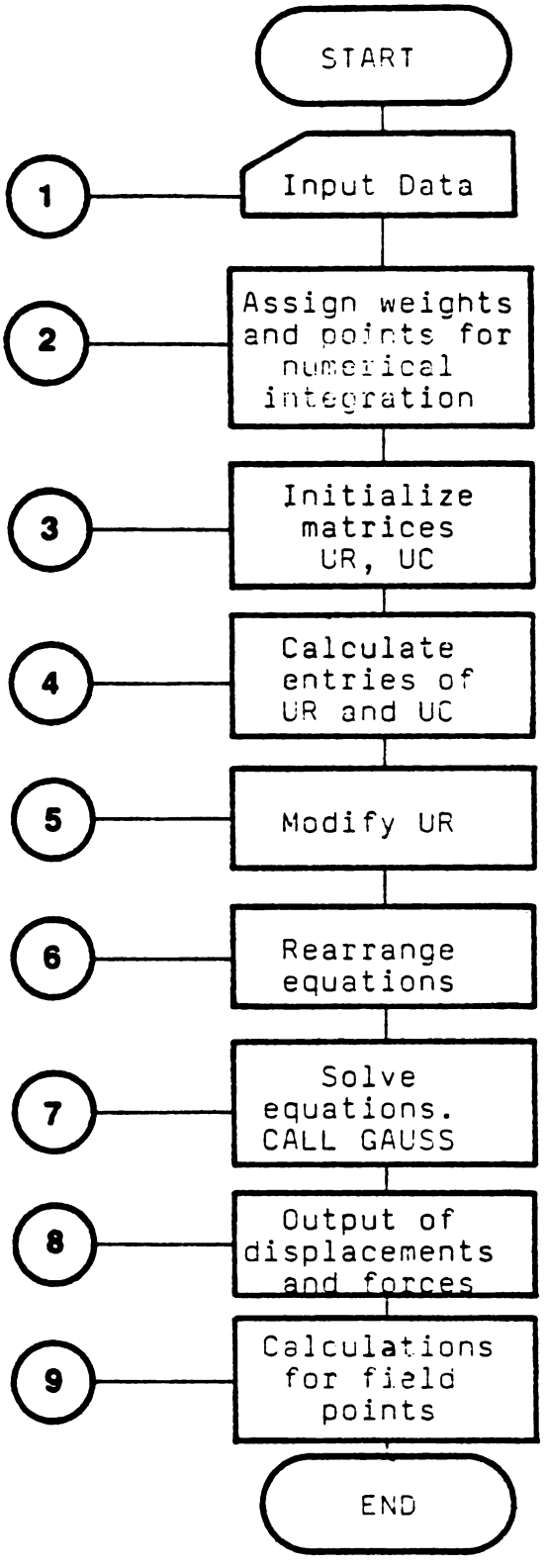


Figure III.5. Flowchart of program BEM

III.3.1 INPUTTING THE PROBLEM DATA.

The order and format in which the variables and arrays must be submitted to the program are as follows:

TITLE -The title of the problem. This must be written on one line in columns 1 through 80.

N -The number of nodes on the boundary. Must be odd number.
Enter in free format.

E, PR -Material properties where E is Young's modulus and PR is Poisson's ratio. Enter in free format.

X(I), Y(I) -Coordinates of nodes I through N, entered counter-clockwise in free format.

J, K -Nodes, J, and corresponding directions, K, at which displacements are specified (zero and non-zero).
Enter in free format and end by inputting 0,0.

J,K,F(2*J+K-2) -Nodes, J, and corresponding directions, K, at which non-zero forces are specified, and specified values F.
Enter in free format and end by inputting 0,0,0..

LL -No. of points used for numerical integration. LL maybe 3,4,5 or 6.

I,XF,YF -Coordinates of each internal point at which displacements and stresses are to be computed. Enter in free format, one point per line. Always begin line with I=1. Enter I=integer other than one to stop.

III.3.2 CALCULATION OF COEFFICIENT MATRICES.

A more detailed flow chart for part 4 of the program is shown in Figure III.6. The order of the loops was chosen in order to avoid repeating operations.

For each value of J, associated with the element J, parts of the (2x2) singular submatrices, shown in Figure III.7, are calculated. Submatrices 1 and 3 correspond to the cases 3 and 4 respectively of Table III.1 Submatrix 2 contains contributions from α_{ij} of eq. (III.2) and from the net contributions of cases 5 and 6 of Tables III.1, i.e. case 7 of that table. Submatrices 4 and 5 correspond to case 1 of Table III.1.

In the other loop, I, all of the nonsingular integrals are calculated by numerical integration. The positions in the matrices are illustrated in Figure III.8. The functions for the numerical integration are calculated using eq. (III.12).

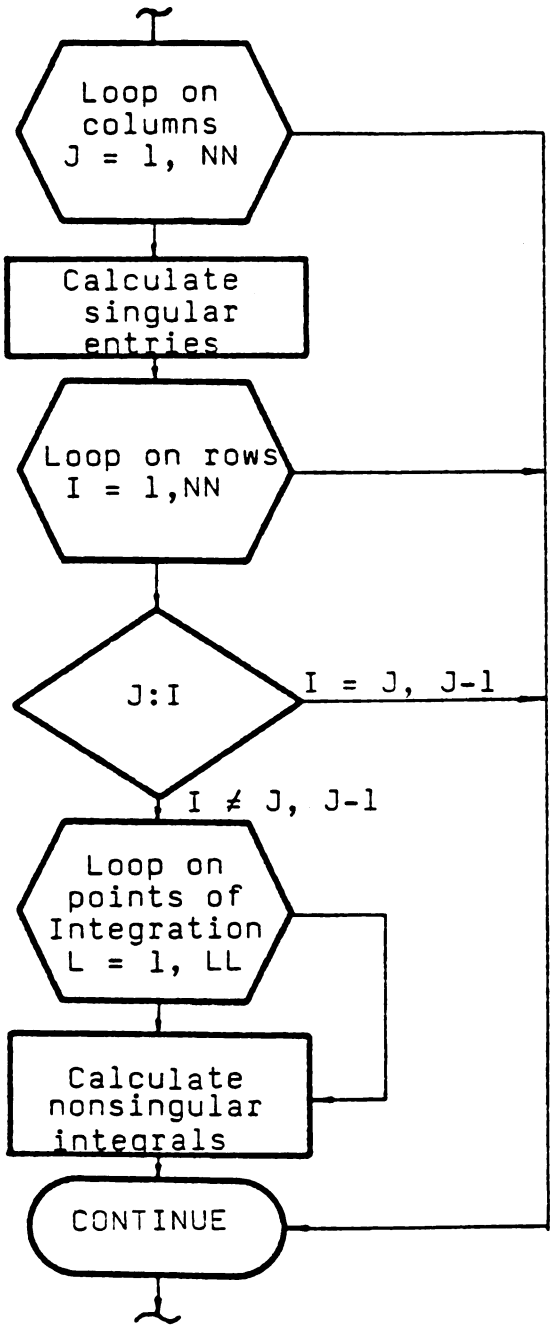


Figure III.6. Flowchart of part 4 of the program

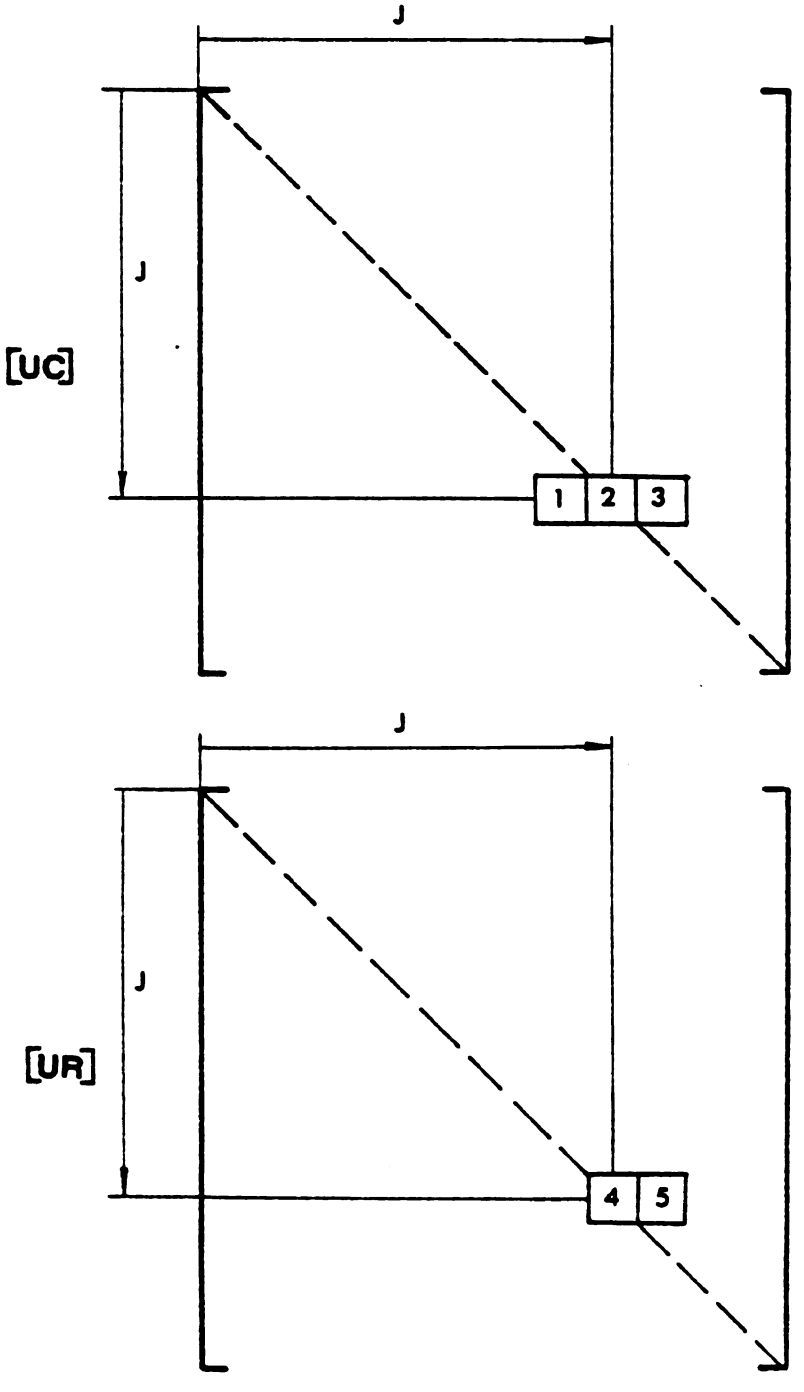


Figure III.7. Singular contributions in [UC] and [UR]

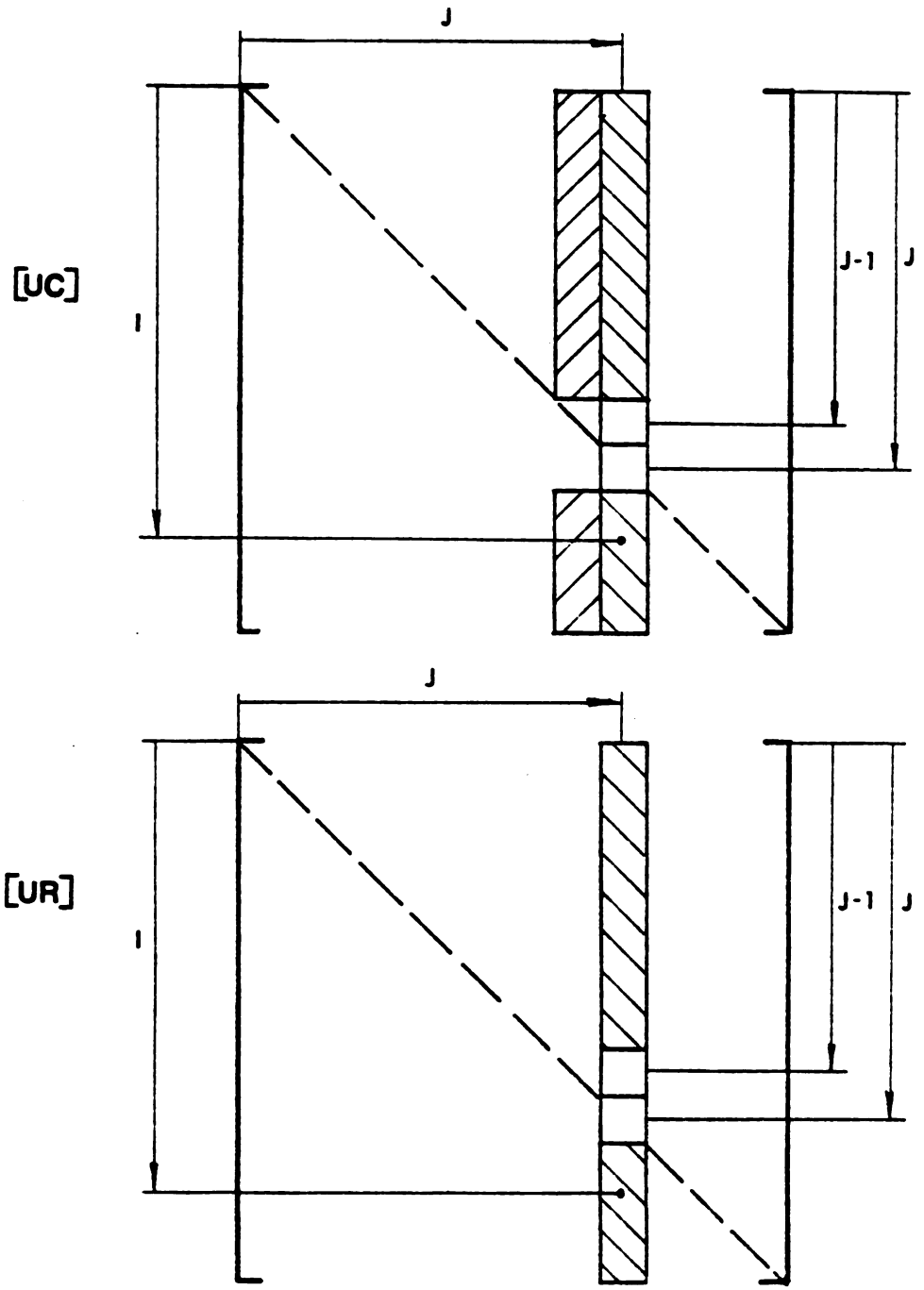


Figure III.8. Non-singular contributions in [UC] and [UR]

III 3.3 TRANSFORMATION OF COEFFICIENT MATRIX UR

This part of the program performs the operation

$$[UR]^{*}=[UR][\Gamma]^{-1}$$

in which $[\Gamma]^{-1}$ is defined in eq.(III.17)

The most efficient way to obtain the matrix $[UR]^{*}$ is not to create $[\Gamma]^{-1}$ and postmultiply by [UR], but to modify [UR] itself.

The algorithm for this modification is better understood by examining the example of a 6 x 6 matrix:

$$\begin{bmatrix} [UR_{11}]^{*} & [UR_{12}]^{*} & [UR_{13}]^{*} \\ [UR_{21}]^{*} & [UR_{22}]^{*} & [UR_{23}]^{*} \\ [UR_{31}]^{*} & [UR_{32}]^{*} & [UR_{33}]^{*} \end{bmatrix} = \begin{bmatrix} UR_{11} & UR_{12} & UR_{13} \\ UR_{21} & UR_{22} & UR_{23} \\ UR_{31} & UR_{32} & UR_{33} \end{bmatrix} \begin{bmatrix} I & -I & I \\ I & I & -I \\ -I & I & I \end{bmatrix},$$

in which the entries $[UR_{21}]^{*}$ are

$$[UR_{21}]^{*}= [UR_{21}] + [UR_{22}] - [UR_{23}]$$

$$[UR_{22}]^{*}=-[UR_{21}] + [UR_{22}] + [UR_{23}]=-[[UR_{21}] + [UR_{22}] - [UR_{23}]]+2[UR_{22}]$$

$$=-[UR_{21}]^{*}+2[UR_{22}]$$

$$[UR_{23}]^{*}=-[UR_{22}]^{*}+2[UR_{23}]$$

In general, it can be shown that

$$[UR_{k1}]^{*}=[UR_{k1}]+[UR_{k2}]-[UR_{k3}]+[UR_{k4}]-....+[UR_{kn}] \qquad \text{(III.23)}$$

and

$$[UR_{ki}]^{*}=-[UR_{k(i-1)}]^{*}+2[UR_{ki}] \ , \ i=2,...,N. \qquad \text{(III.24)}$$

The flow chart for this modification is illustrated in Figure III.9. In the first loop, the first two columns of $[UR]^*$ are obtained using eq. (III.23). In the second loop, the other entries of $[UR]^*$ are generated using eq. (III.24).

III.3.4 CONSOLIDATION OF KNOWN AND UNKNOWNS BOUNDARY VALUES

This part of the program consists of a reorganization of the system of eqs. (III.20) so that all unknowns go to the left hand side and all knowns go to the right hand side of the equations. This procedure is illustrated with the flow chart of Figure (III.10). The array NTBC (K,I) which specifies the boundary condition at node K, directions I, is used to decide when to interchange columns, i.e. if NTBC(K,I,)=0, there is a specified force, and no interchange is necessary.

The final vector of knowns is multiplied by the matrix of coefficients to produce the right-hand side of the final system of equations.

The final system of equations is then solved by the subroutine GAUSS, which uses the well known GAUSS elimination procedure.

III.3.5 CALCULATION OF STRESSES AND DISPLACEMENTS AT INTERNAL POINTS

In this part of the program, eq. (III.21) are evaluated. Initially the matrices $[UR^{(u)}]$, $[UC^{(u)}]$, $[UR^{(s)}]$, $[UC^{(s)}]$, defined by eqs. (III.12) and (III.22), are calculated by numerical integration using the Gauss quadrature formula. These matrices are saved in the first five rows of the arrays [UC] and [UR]. Next $[UR^{(u)}]$ and $[UR^{(s)}]$ are modified to produce $[UR^{(u)}]^*$ and $[UR^{(s)}]^*$, respectively. Finally, the matrices are

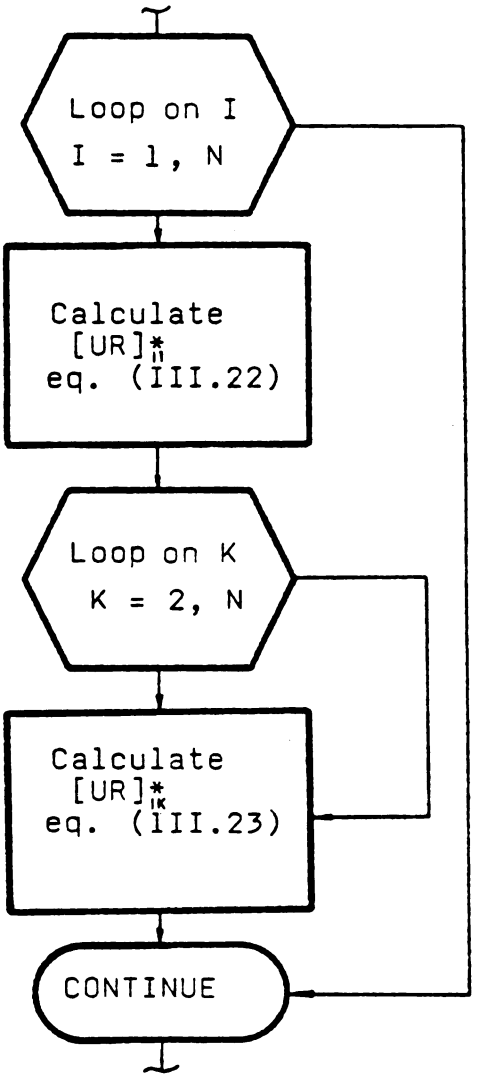


Figure III.9. Flowchart for the modification of [UR]

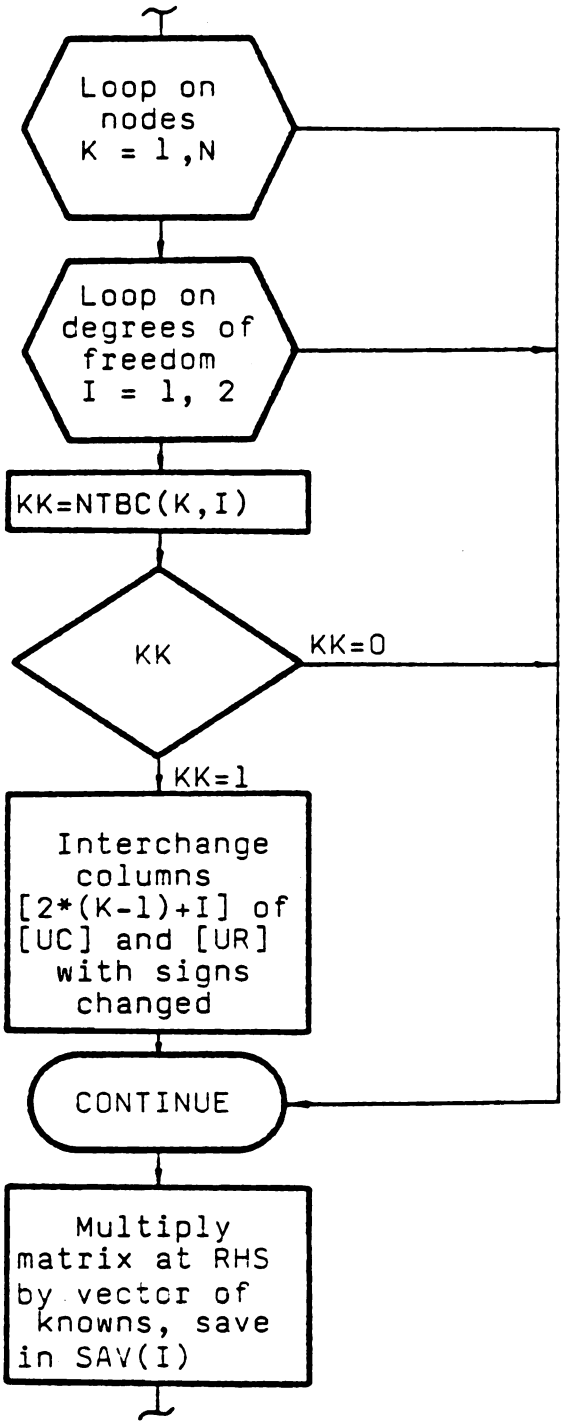


Figure III.10. Flowchart for rearranging equations into final form

multiplied by the vectors $\{F\}$ and $\{u\}$ to produce displacements and stresses at each chosen field point.

III.3.6 EXAMPLE OF INPUT DATA

For the example of Figure III.11, the input data is as follows

```
(TITLE, 1-80)
     EXAMPLE OF INPUT DATA - BEM
(N, Free format)
7
(E, PR, Free format)
2.5E00                  .25
(X(I), Y(I), Free format)
0.0                     0.0
2.0                     0.0
4.0                     0.0
4.0                     2.0
2.0                     2.0
0.0                     2.0
0.0                     1.0

(J,K, Free format)
1    1
1    2
7    1
6    1
0    0

(J,K,F(2*J+K-2) Free format)
3    1    1.0
4    1    1.0
0    0    0.0
```

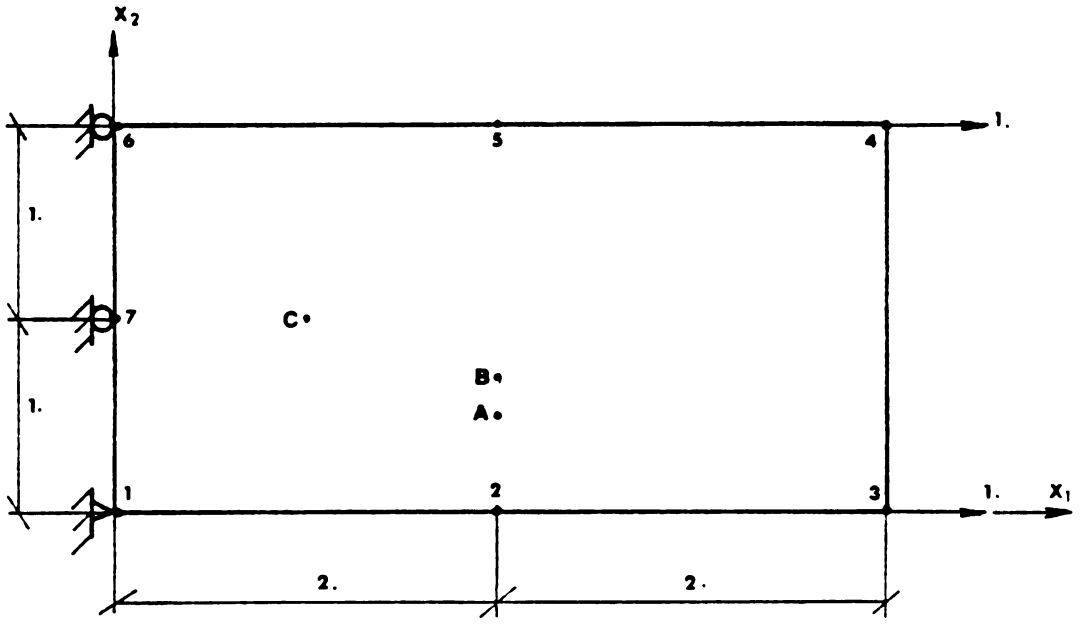


Figure III.11. Example problem to demonstrate input data for BEM

```
(LL, Free format)
5

(I,XF,YF, Free format)
1    2.0  0.5
1    2.0  0.7
1    1.0  1.0
0    0.0  0.0
```

Displacements and stresses are calculated at points A,B and C of Figure III.11.

CHAPTER IV

COUPLING THE FINITE ELEMENT AND BOUNDARY ELEMENT METHODS

IV.1 PROCEDURE

The procedure outlined here for combining the two methods has been suggested by several authors, [3,4,6,7,8,9]. The method consists of dividing the body as shown in Figure IV.1. Elements defined by more than four nodes, such as the shaded element, are treated as "superelements", whose symmetric stiffness matrices are obtained by using the boundary integral equations. These matrices are incorporated into the global stiffness matrix for the entire body and the equations are solved as was done in the program FEM described in CHAPTER II. The field equations are satisfied exactly in each superelement. Compatibility with the neighboring finite elements is guaranteed since the shape functions for the superelement boundaries have the same form as those for the isoparametric quadrilateral and triangular elements. Equilibrium is satisfied in an average sense.

The procedure used to obtain a symmetric stiffness matrix for a superelement begins with solution of the system (III.20) for the set of displacement vectors in which only one of the components is 1 and all others are 0. The solution vectors can be arranged in a matrix [E] so that

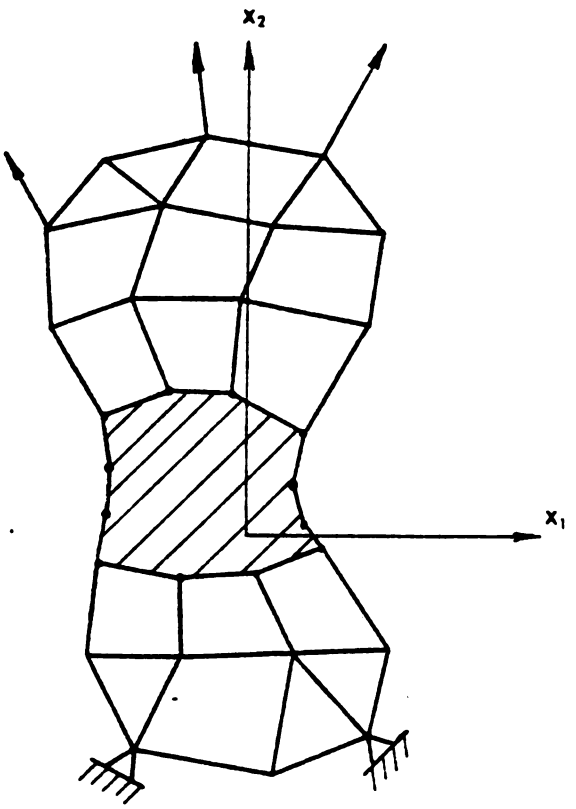


Figure IV. 1. Combined discretization

$$[E]\{u\}^{(se)}=\{F\}^{(se)} \quad , \tag{IV.1}$$

where $\{u\}^{(se)}$ and $\{F\}^{(se)}$ are the vectors of nodal displacements and nodal forces, respectively, for the superelement. The same result may be reached if the inverse of the augmented matrix [UR]* is multiplied by [UC], but this procedure was shown to be less efficient for the problems examined.

Note that the entry E_{ij} should be equal to E_{ji}, but this condition is, in general, only attained exactly as the number of nodes goes to infinity. However, since the boundary element model has exactly the same interpolating functions on the elements as the finite element triangular element, the matrix [E] of eq. (IV.1) is exactly equal to the stiffness matrix for that triangular element.

It is important to note that the rows and columns of [E] each sum to zero, since equilibrium equations were enforced in the superelement.

The necessary symmetric matrix of the superelement is obtained by minimizing the functional

$$\Pi^{(se)}=\tfrac{1}{2}\int_V\{\varepsilon\}^T\{\sigma\}dv-\{u\}^{(se)T}\{F\}^{(se)} \tag{IV.2}$$

which can be written for the discretized model as

$$\Pi^{(se)}=\tfrac{1}{2}\{u\}^{(se)T}[E]\{u\}^{(se)}-\{u\}^{(se)T}\{F\}^{(se)} \quad . \tag{IV.3}$$

The minimization produces the system

$$[K^{(se)}]\{u\}^{(se)}=\{F\}^{(se)} \tag{IV.4}$$

where

$$[K^{(se)}]=\tfrac{1}{2}\{[E]+[E]^T\}. \qquad (IV.5)$$

As the number of nodes in the superelement increases, [E] approaches [K].

IV.2 PLANE STRESS HYBRID PROGRAM

IV.2.1 GENERAL DESCRIPTION

This combined program is the same finite element program FEM described in CHAPTER II, with the addition of the subroutine BOUN, which creates the stiffness matrix of the superelement and calculates stresses and displacements at the field points within the superelement.

A few variations are also introduced into the subroutines PFORM and STRESS, allowing the program to call BOUN when required, see Figure IV.2. The format and order for the input data is exactly the same as was described in CHAPTER IV, with the addition of the coordinates of the field points in the superelement as described in section III.3.1. An example of the input data for a problem is given in section IV.2.4.

IV.2.2 SUBROUTINE BOUN

The flow chart for BOUN is shown in Figure IV.3. The first part of the subroutine, which works when the argument IK is 1, calculates the stiffness matrix of the superelement. The second part of BOUN computes the stresses and displacements at the field points when the argument IK is 2. Steps 1,2 and 3 of BOUN are the same as those already explained for the program BEM in CHAPTER III.

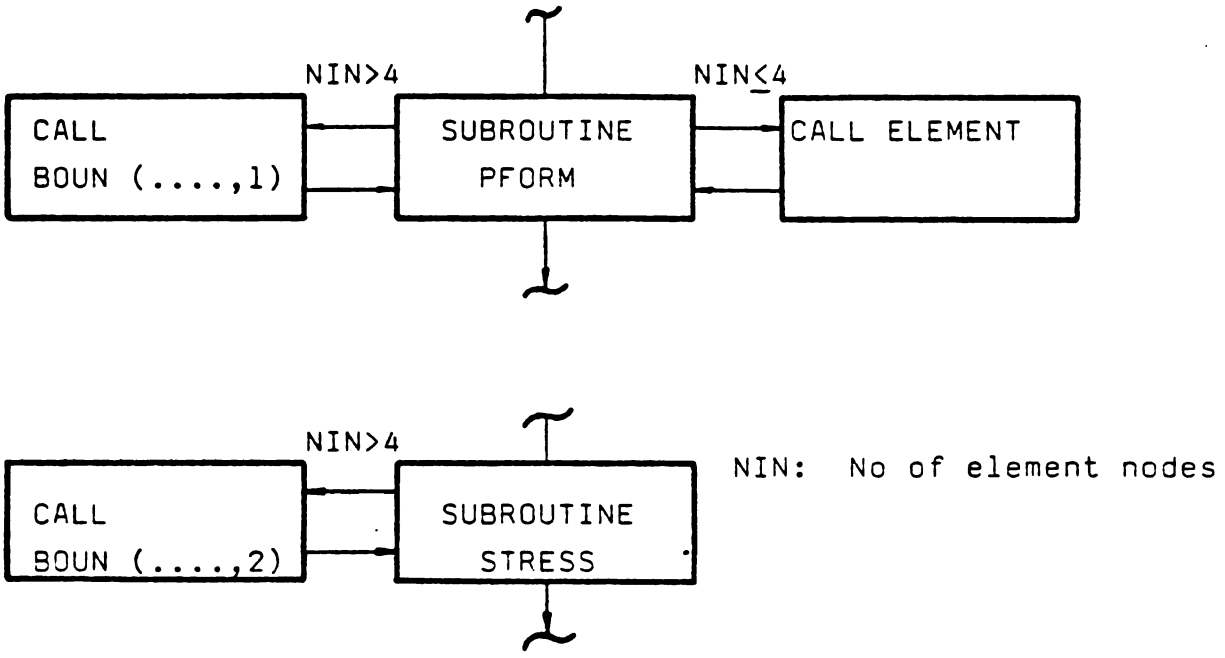


Figure IV.2. Interaction between BOUN and FEMBOU

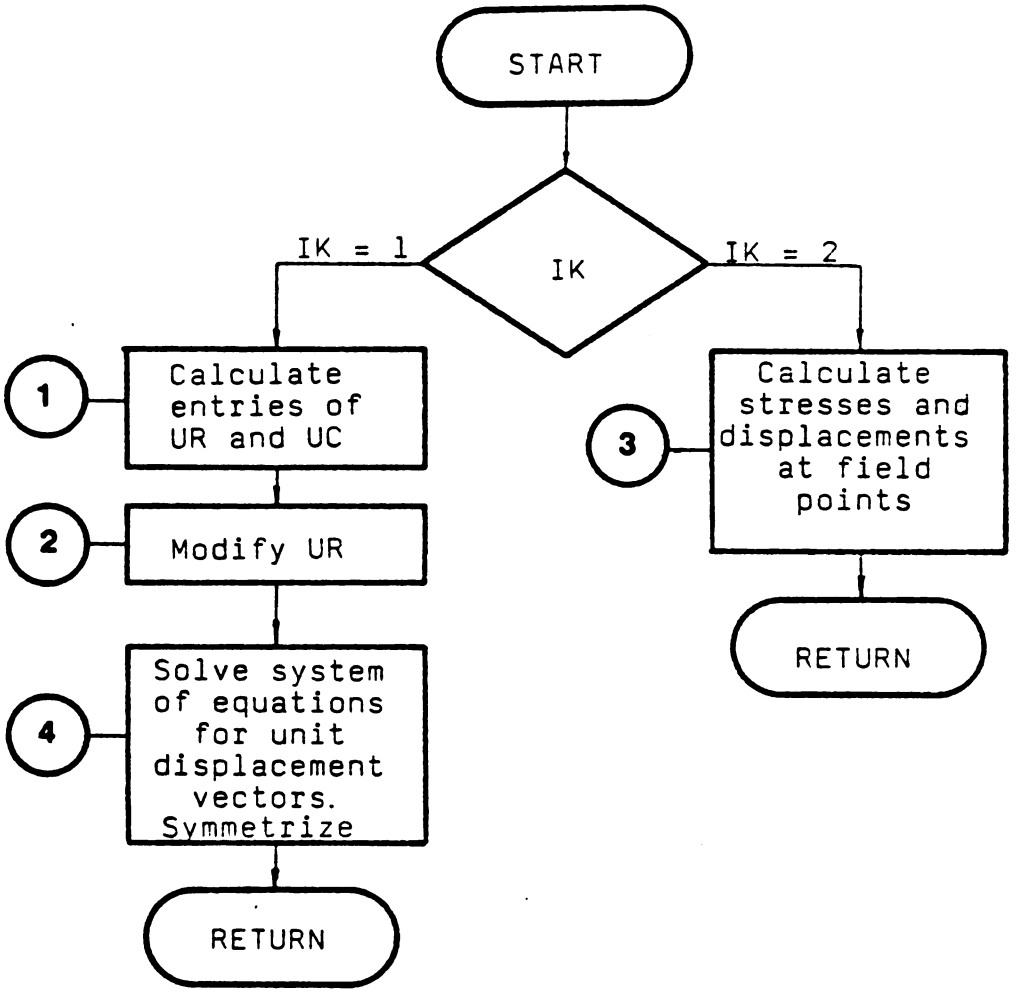


Figure IV.3. Flowchart for BOUN

In part 4 of the subroutine, the system of equations

$$\begin{bmatrix} UC \\ 0 \end{bmatrix}_{(2N+3,2N)} = \begin{bmatrix} UR^* & Q^T \\ Q & 0 \end{bmatrix}_{(2N+3,2N+3)} \begin{Bmatrix} \{F\} \\ \{\lambda\} \end{Bmatrix} \qquad \text{(IV.6)}$$

is solved for the set of unit prescribed displacements of the form

$$\{u\} = \begin{Bmatrix} 0 \\ 0 \\ \cdot \\ \cdot \\ 0 \\ 1 \\ 0 \\ \cdot \\ 0 \end{Bmatrix}$$

This is the same as solving the system

$$\begin{bmatrix} UR^* & Q^T \\ Q & 0 \end{bmatrix} \{x\} = \{b\}$$

for vectors $\{b\}$ equal to each of the colums of $\begin{bmatrix} UC \\ 0 \end{bmatrix}$. The method for doing this is to perform the factorization

$$\begin{bmatrix} UR^* & Q^T \\ Q & 0 \end{bmatrix} = [L][U] \quad ,$$

where [L] and [U] are lower and upper triangular matrices respectively, and then to solve by back and forward substitution for each column of $\begin{bmatrix} UC \\ 0 \end{bmatrix}$. An additional array IPIV(I) contains row permutations if any are required. Each solution vector is saved in the same matrix [UC] so that, when the procedure is completed, the first 2N rows of this matrix are equal to the matrix [E] of eq. (IV.1). A more detailed explanation of this method can be found in [12].

IV.2.3 INPUT PROCEDURE

The input data procedure is almost the same as that for the FEM program, although few variations are introduced as follows.

The number of points of integration is fixed into the program as 3 for the finite elements and 6 for the boundary elements, therefore these numbers do not need to be given to the program.

The points in which stresses are to be calculated within the finite elements are defined by the local coordinate R as in the FEM program. The intervals of finite elements in which the stresses are to be calculated are given in ascending order, ending with zeros. If no stress calculation is required into the finite elements, two zeros must be input. The field points within each superelement must be given after the interval of elements corresponding to that particular superelement, ending with zeros as in the BEM program. If no stress calculation is to be made into the finite elements, the field points within superelements must be given in order from the first to the last superelement in the problem, each list ending with zeros as in the BEM program. An aclaratory example is presented in the next section.

IV.2.4 EXAMPLE OF INPUT DATA

The input data for the problem of Figure IV.4 is as follows.

(TITLE, 1-80)

EXAMPLE OF INPUT DATA, FEMBOU

(NMAT,NEL,NNO,NDOF,NLO, Free format)

1 5 12 2 1

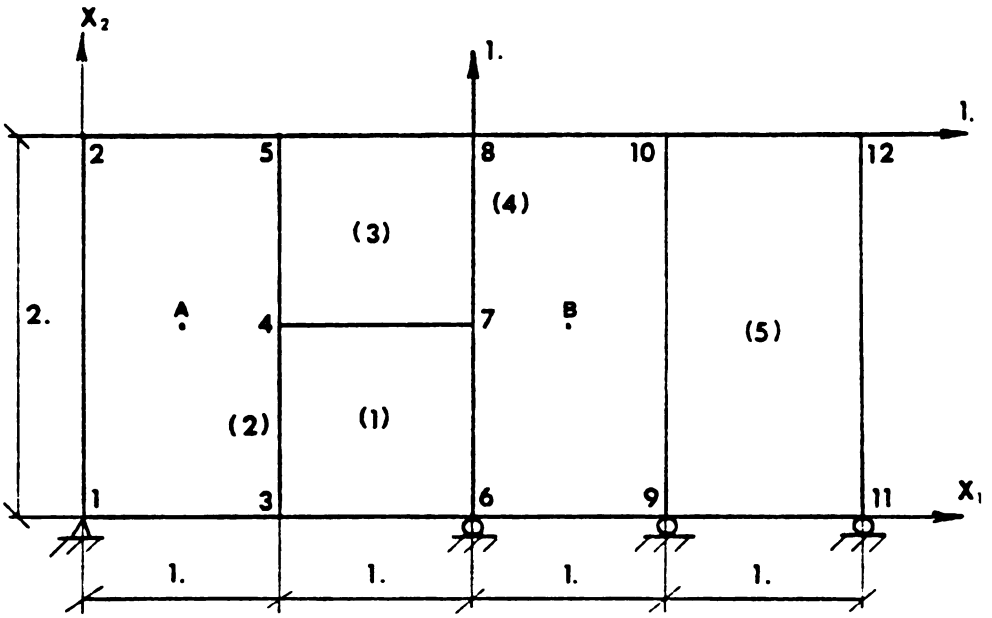


Figure IV.4. Example problem to demonstrate input data for FEM80U

```
(E(1),PR(1),E20.12,F20.12)

2.5E00                0.25

(MP(I),NEN(I),(NN(I,J),J=1,NEN(I)), Free format)

1    4    3    6    7    4
1    5    1    3    4    5    2
1    4    4    7    8    5
1    5    6    9    10   8    7
1    4    9    11   12   10

(XE(I,1),NTBC(I,1),XE(I,2),NTBC(I,2),F15.5,I5,F15.5,I5)
0.0                  -1 0.0                -1
0.0                     2.0
1.0                     0.0
1.0                     1.0
1.0                     2.0
2.0                     0.0                -1
2.0                     1.0
2.0                     2.0
3.0                     0.0                -1
3.0                     2.0
4.0                     0.0                -1
4.0                     2.0

(N,I,F(2*(N-1)+I), Free format)
12   1    1.0
 8   2    1.0
 0   0    0.0
```

Three different alternatives for calculation of stresses are as follows.

```
(R, Free format)
0.5

(NE1, NE2, Free format)
1    4
0    0
(I, XF, YF, Free format)
1    0.5  1.0
0    0.0  0.0
1    2.5  1.0
0    0.0  0.0
```

Stresses are to be calculated into the finite elements 1 and 3, at four points defined by the local coordinate R=0.5; and at points A and B of superelements 2 and 4 respectively.

```
(R, Free format)
0.5
```

```
(NE1, NE2, Free format)
0    0

(I, XF, YF, Free format)
0    0.0  0.0
1    2.5  1.0
0    0.0  0.0
```

Stresses are to be calculated at point B of superelement 4.

```
(R, Free format)
0.5

(NE1, NE2, Free format)
(I, XF, YF, Free format)
1    2
1    0.5  1.0
0    0.0  0.0
4    5
1    2.5  1.0
0    0.0  0.0
0    0
```

Stresses are to be calculated into the finite elements 1 and 5, at four points defined by the local coordinate R=0.5; and at points A and B of superelements 2 and 5 respectively.

CHAPTER V

EXAMPLES AND DISCUSSION

V.1 EXAMPLES

In first example, the cantilever beam shown in Figure V.1 was modelled using 13 and 19 nodes on the boundary, as illustrated in Figures V.2 and V.3. The CPU time on a PRIME 750 minicomputer and the computed vertical deflection of the point A are shown in Table V.1. The vertical deflection is compared to the exact solution for the same problem with the true parabolic load distribution on its end, i.e. 14.8 taken from [13]. Normal stresses, σ_{11}, on the section a-a are shown in Figure V.3 for the 19-node discretization. Note that the results of σ_{11} given by BEM and FEMBOU are not reliable near the boundary.

In the second example, half of the plate of Figure V.5 was modelled using 49 and 55 nodes on the boundary for the ratios R/w=0.5 and R/w=0.2 respectively, see Figures V.6, V.7, V.8, V.9, V.10 and V.11. Results for the normal stress, σ_{22}, on section a-a are plotted in Figures V.12 and V.13, as well as the exact solution for the problem in which the notch is a perfect semicircle, as taken from [14,15]. Note that the ratio of perimeter to area is 1.46 1/UL for the plate with R/w=0.5, and 0.53 1/UL for the plate with R/w=0.2, where UL is a length unit.

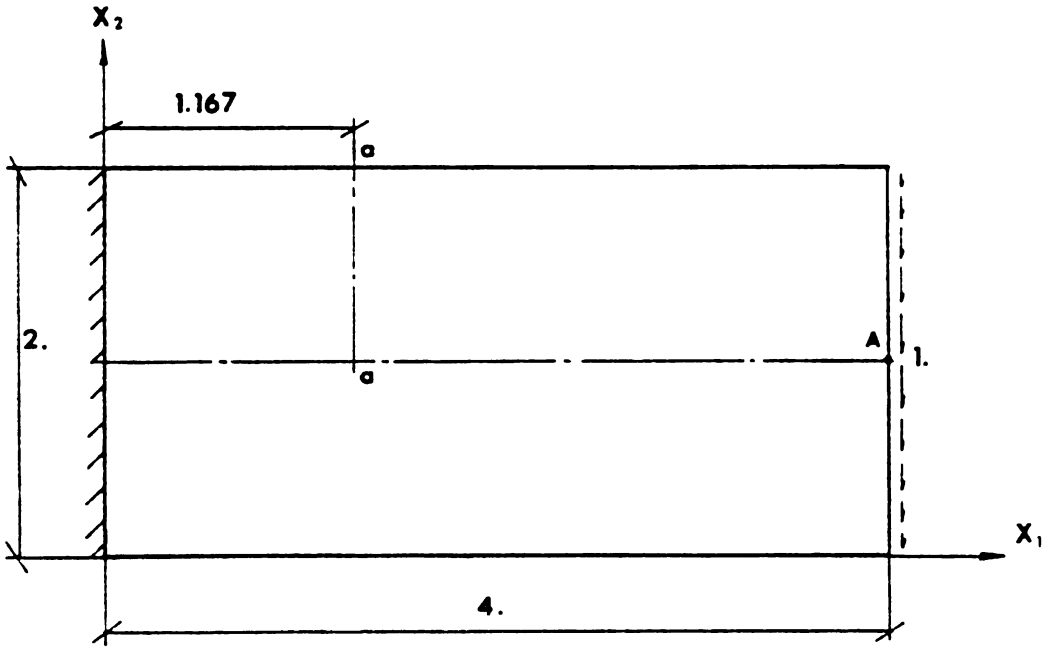


Figure V.1. Cantilever beam

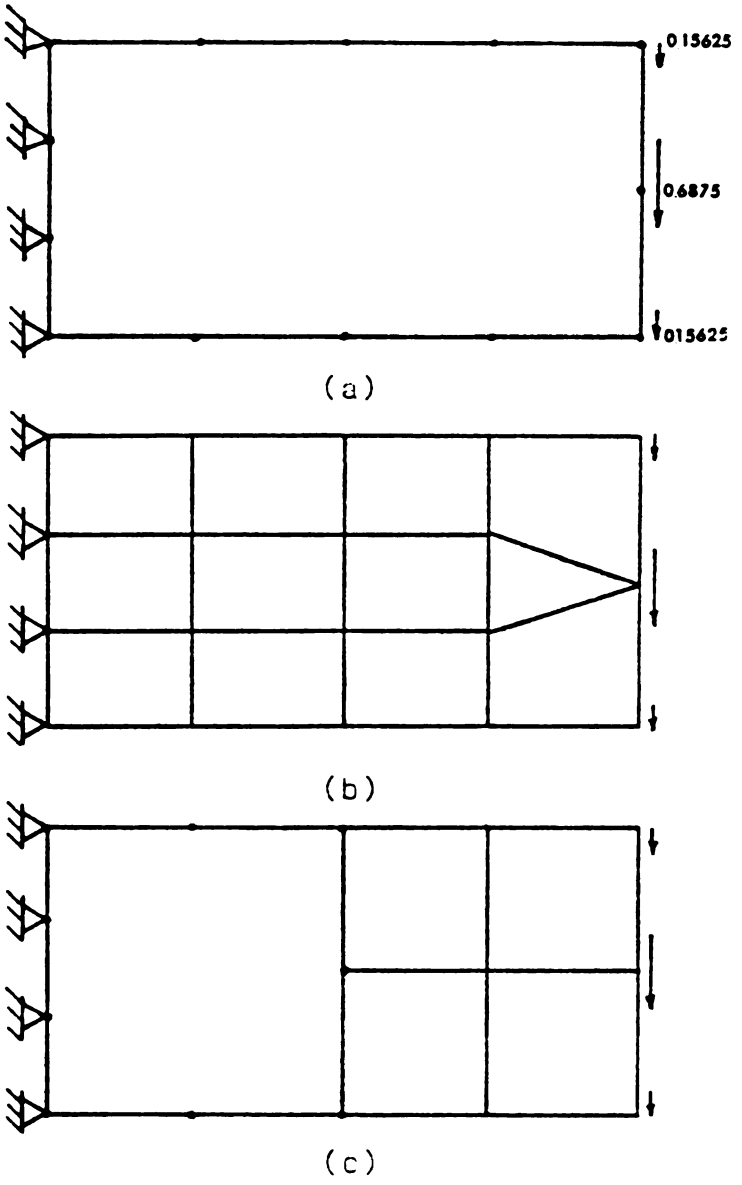


Figure V.2. 13 node boundary discretization of cantilever beam

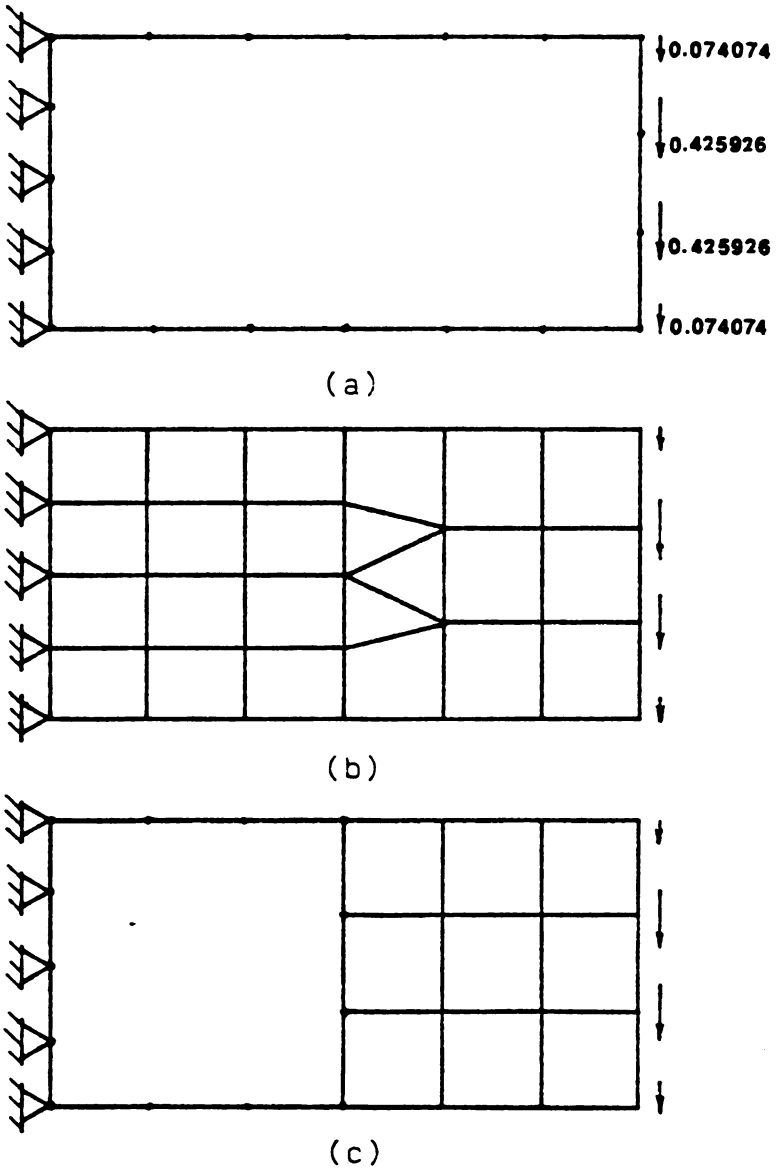


Figure V.3. 19 node boundary discretization of cantilever beam

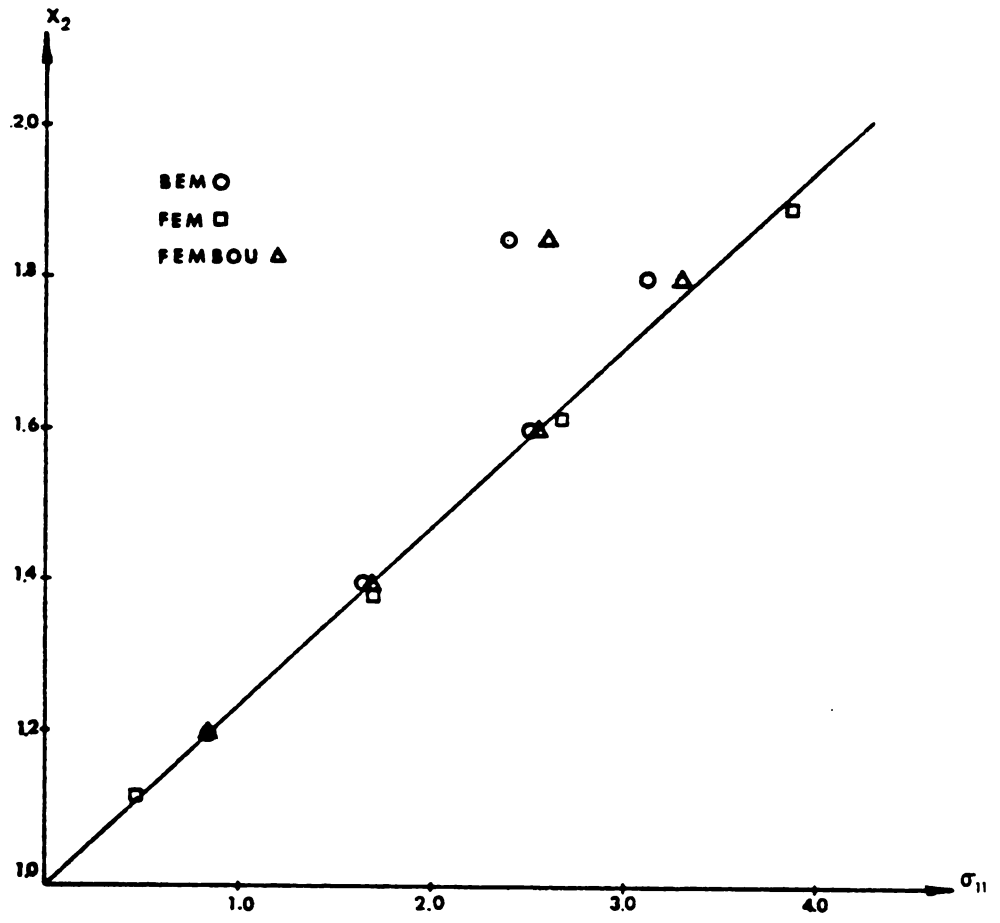


Figure V.4. Stress distribution on section a-a, located 2.833 from free end

Table V.1. Vertical deflection of A and CPU time for cantilever beam problem.

Model	Program	Deflection of A	%difference(14.8)	CPU Time(sec)
13	BEM	-13.829	6.57	2.57
	FEM	-13.358	9.74	3.52
	FEMBOU	-14.194	4.09	3.41
19	BEM	-14.230	3.85	4.52
	FEM	-14.149	4.39	6.13
	FEMBOU	-14.292	3.43	5.81

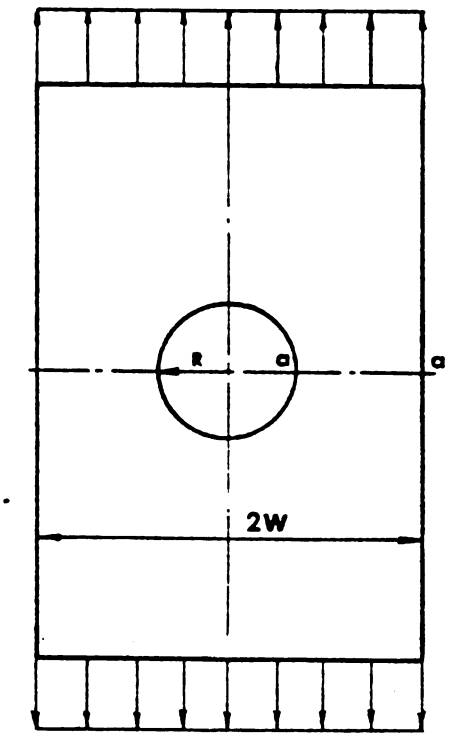


Figure V.5. Example No. 2

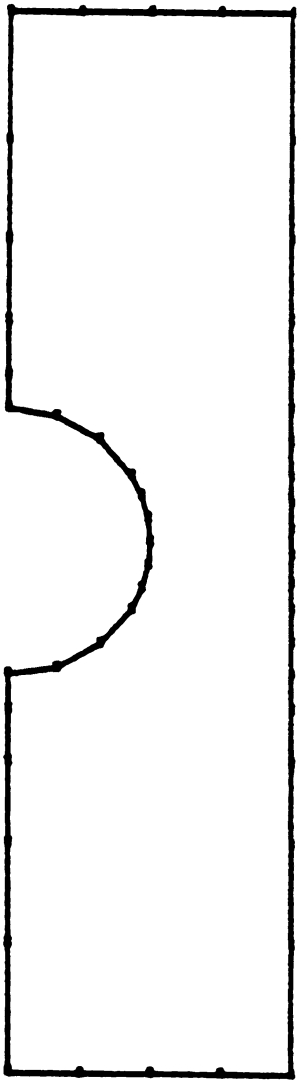


Figure V.6. BEM model for R/W = 0.5

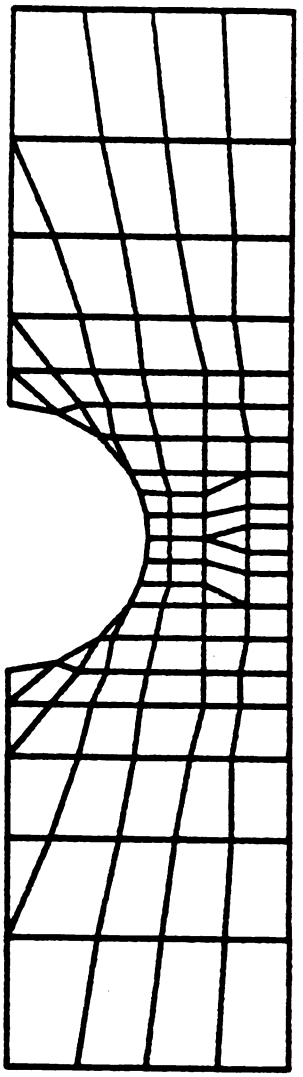


Figure V.7. FEM model for R/W = 0.5

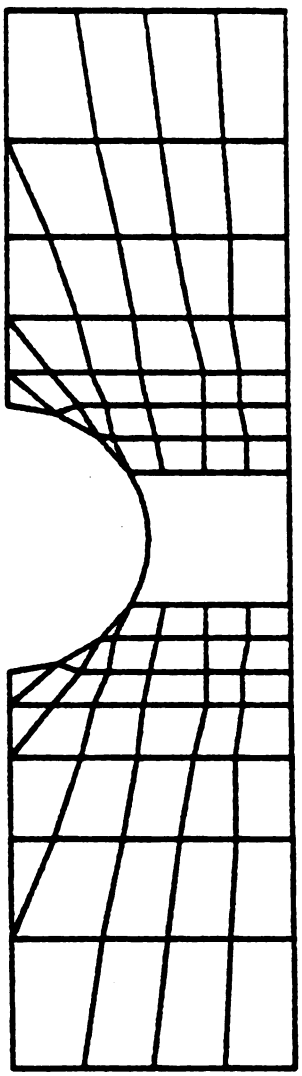


Figure V.8. FEMBOU model for R/W = 0.5

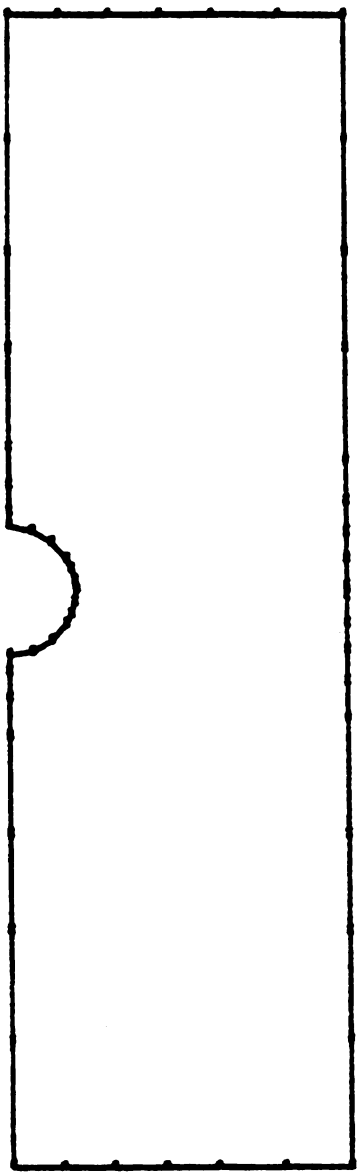


Figure V.9. BEM model for R/W = 0.2

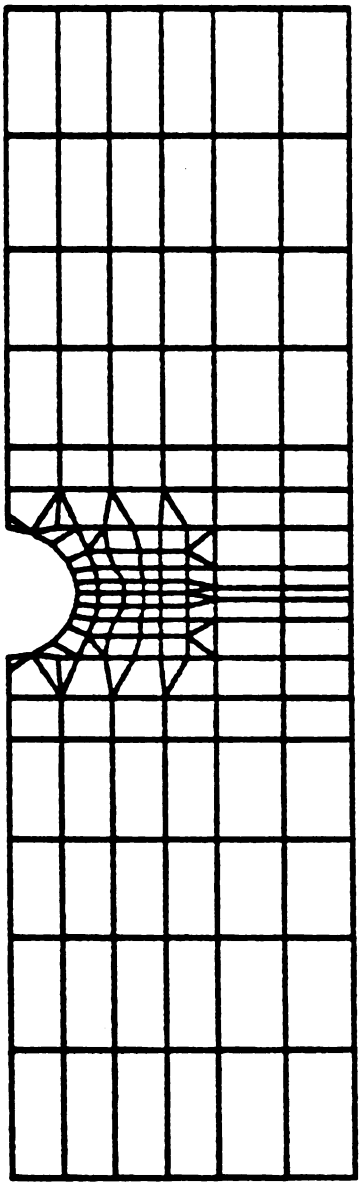


Figure V.10. FEM model for R/W = 0.2

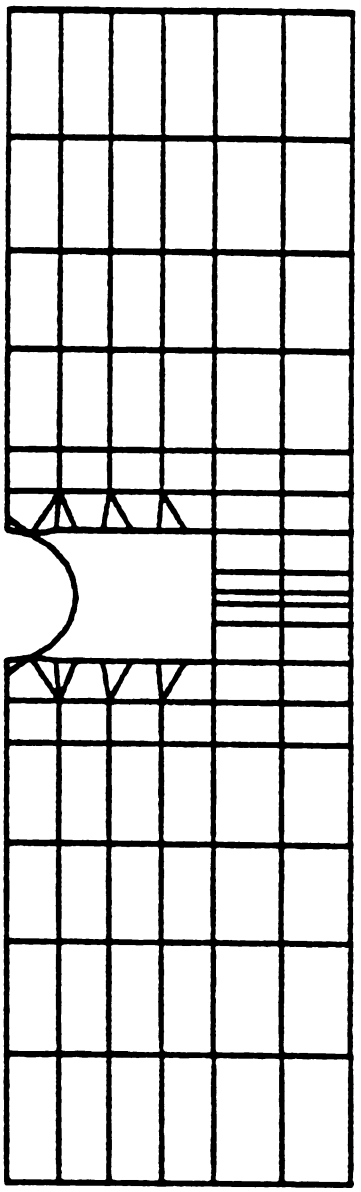


Figure V.11. FEMBOU model for R/W = 0.2

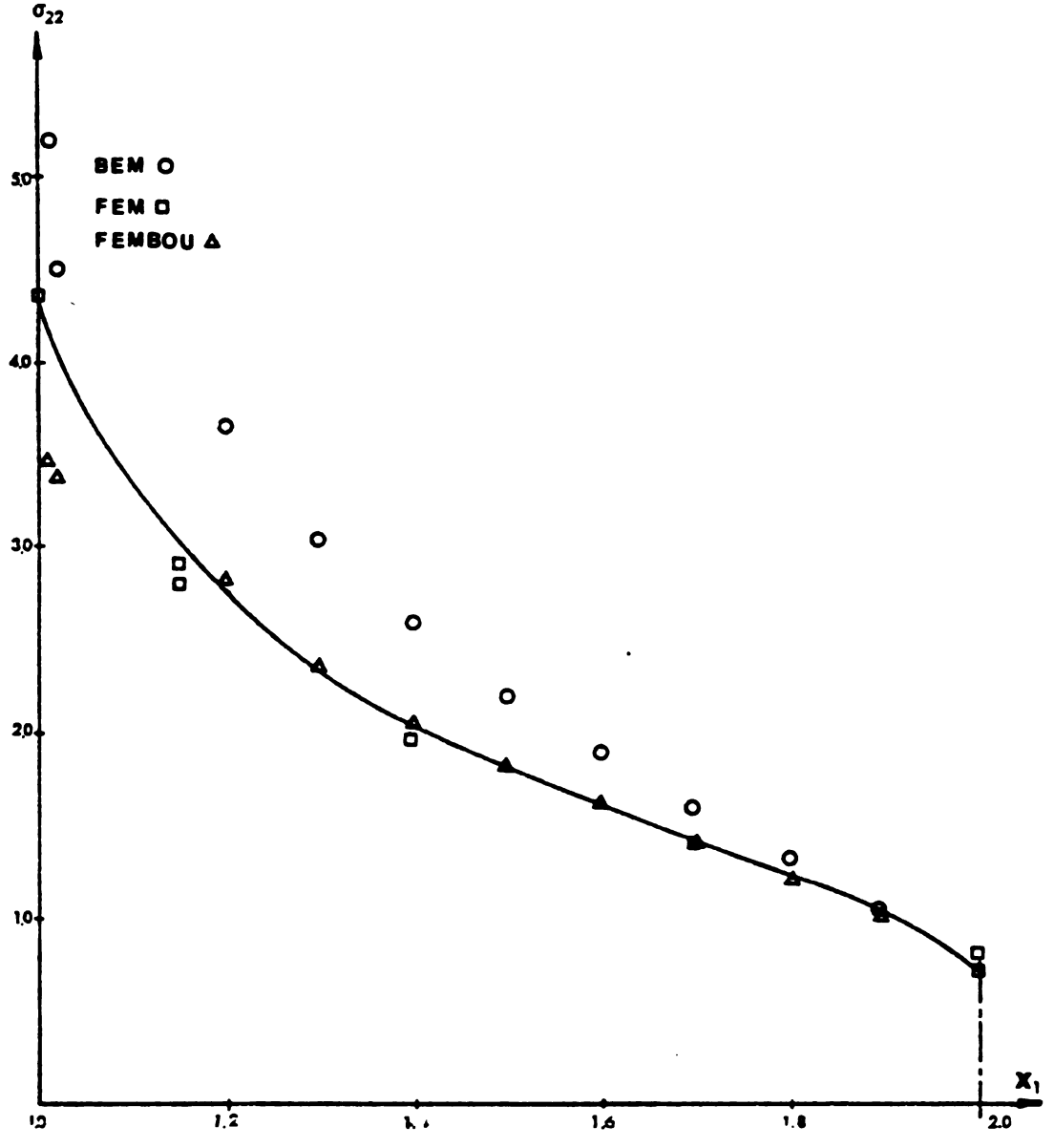


Figure V.12. Stresses σ_{22} on section a-a for example 2, R/W = 0.5

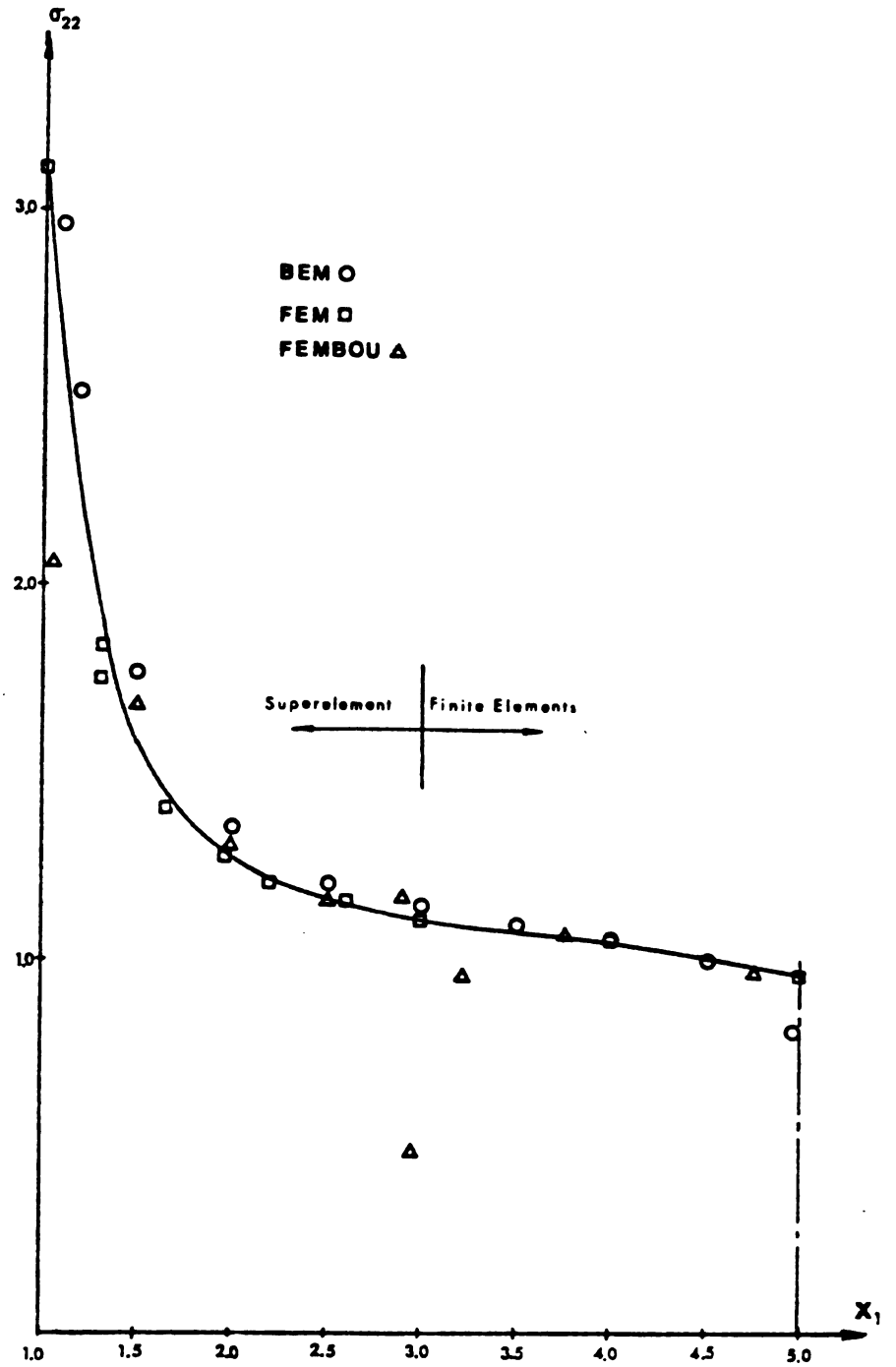


Figure V.13. Stresses σ_{22} on section a-a for example 2, R/W = 0.2

CPU time in seconds, on a PRIME 750 minicomputer is listed in Table V.2. The number of equations solved in each procedure and the number of components of the coefficient matrices are in Table V.3.

Table V.2 CPU time for problems of Figure V.5

Relation R/W	Program	CPU READ	CPU FORM	CPU SOLVE,STRESS	TOTAL CPU
0.5	BEM	1.03	8.17	17.13	26.33
(49 nodes	FEM	3.01	8.46	4.71	16.18
boundary)	FEMBOU	2.60	8.25	6.00	16.85
0.2	BEM	1.27	12.19	27.55*	41.01
(55 nodes	FEM	4.53	14.74	9.00	28.27
boundary)	FEMBOU	3.19	17.23	8.87	29.29

*27.55=18.52(solving)+0.62 (writting)+8.41 (stresses)

Table V.3. Number of equations and components of coefficient matrices for problems of Figure V.5.

Model	Program	No. of equations	No. of comp. in coefficient matrices
49 nodes	BEM	101	10201
	FEM	225	3127
	FEMBOU	197	3147
55 nodes	BEM	113	12769
	FEM	317	5808
	FEMBOU	241	5681

V.2 DISCUSSION OF RESULTS

Some conclusions can be drawn from the results of the two examples.

As expected, it is much easier to prepare and input the data for BEM than for FEM. This factor can be quantified by comparing the time that each program spends reading the data. This advantage of BEM will undoubtedly be even higher when solving three dimensional problems.

The time spent calculating the coefficient matrices [UC] and [UR] by BEM, is a little bit lower than the time spent creating the global stiffness matrices by FEM and FEMBOU. The comparison is made using 6 points of integration on each boundary element and 9 points of integration in each quadrilateral finite element. The difference is not substantial however. On the other hand, there appears to be a substantial difference in the time required for creation of the stiffness matrix of the superelement when the number of nodes goes beyond a certain value between 19 and 29. Recall that the computation of this stiffness matrix requires a matrix inversion process, the expense of which increases dramatically as the number of nodes increases.

For the same number of boundary nodes, fewer equations are solved in the BEM program, but since the BEM matrix of coefficients is neither symmetric nor banded, a great effort is necessary for solving the equations. Conversely, the high number of equations in the FEM method are solved quite efficiently, due to the symmetric and banded form of the global stiffness matrix. This difference can be appreciated by looking at Table V.3 and at Column 5 of Table V.2. On the other hand, the time for the combined program is almost the same as for FEM.

Due to the boundary discretization of the boundary integral equations, field calculations are not reliable near the boundary, particularly when non-zero tractions are applied on the boundary as is the case in Figure V.13 near the line joining the finite elements to the superelement. Good accuracy is expected at 1 1/2 mesh length from the border.

In the case of relatively thin elements, such as the plate for which R/w=0.5, the accuracy of the BEM program is not good compared to the accuracy of the other two programs working with analogous modells. On the other hand, the combined program produces the most accurate results at interior points far enough from the borders. This is an encouraging result in anticipation of future three dimensional programs.

In general, the FEM program proved to be the most accurate and most efficient for the problems solved. However a considerable amount of time was spent preparing the data for each problem. Very good results were obtained with the FEMBOU program at a speed similar to that for FEM and with less time spent creating the input data. Some attention must be paid to improving the boundary element stress calculations near or at the borders and to improving the effectiveness of some of the algorithms in the boundary element programs, i.e. solving equations in BEM and creating the stiffness matrix of the superelement in FEMBOU. Some suggestions are offered below.

V.3 RECOMMENDATIONS FOR FUTURE STUDY

The following suggestions or considerations are made regarding future work in this area.

It appears to be possible to calculate stresses near or on the free-traction portion of the boundary in the programs BEM nad FEMBOU, this calculation would require exact evaluation of the singularities of $[UC^{(u)}]$ of eq. (III.21), which involves a procedure similar to that for the calculation of the singularities of [UC] as explained in Chapter III. All the terms of $[UR^{(s)}]$ of eq. (III.21) can be computed by numerical integration. Anoter technique, such as extrapolation or a Finite-Boundary element combination, must be devised for the stress calculation on the loaded boundaries.

The efficiency of the FEMBOU program can be improved a great deal in the cases where the superelement is not completely imbedded within the finite elements. For this case, only the "stiffness" matrix, corresponding to the nodes in the common finite-boundary border must be calculated, which implies that eqs. (III.20) must be solved only for the unit displacement vectors involving the common boundary-finite nodes.

It would be wise to examine other methods for solving the system of equations in BEM. Should a good algorithm be found for this operation, the general efficiency of the program will be improved a great deal, [16].

Since the analytical calculation of singular entries is sometimes quite complicated, it might be advisable to use rigid body motion arguments to compute those entries, i.e. rigid body displacements in x_1 or x_2 directions must produce a zero stress field so that all the even

entries of the same row of [UC] must add to zero (the same applies for the odd entries). This argument, useful here for checking intermediate results, should be used to prepare the three dimensional program.

It is advisible to prepare a subroutine to generate and check graphically the subdivisions for a given Finite Element problem. This applies not only for the plane problem but particularly for the three dimensional case. Many articles about this topic can be found in the literature, [17,18,19]

Few changes are necessary to extend FEM to the three dimensional case. Another part must be added to the subroutine ELEMENT for the calculation of the stiffness matrix of the the three dimensional element. Small changes are required in some dimensions and formats of the program.

APPENDIX

COMPUTER PROGRAM LISTS

```
00001:C
00002:C*************************************************************************
00003:C
00004:       PROGRAM BEM
00005:C
00006:C     THIS PROGRAM EMPLOYS THE DIRECT BOUNDARY ELEMENT METHOD
00007:C     (USING STRAIGHT BOUNDARY ELEMENTS CHARACTERIZED BY LINEAR
00008:C     DISPLACEMENTS AND CONSTANT TRACTIONS) TO SOLVE PLANE STRESS
00009:C     PROBLEMS OF LINEAR ELASTICITY. UNIT THICKNESS IS ASSUMED.
00010:C
00011:C     WE DEFINE THE FOLLOWING INPUT PARAMETERS, VARIABLES
00012:C     AND ARRAYS:
00013:C
00014:C     TITLE   = THE TITLE OF THE PROBLEM. THIS MUST BE WRITTEN
00015:C               ON ONE LINE IN COLUMNS 1 THROUGH 80.
00016:C     N       = THE NUMBER OF NODES ON THE BOUNDARY. MUST BE
00017:C               ODD NUMBER. ENTER IN FREE FORMAT.
00018:C     E,PR    = MATERIAL PROPERTIES (E IS YOUNG'S MODULUS
00019:C               AND PR IS POISSON'S RATIO). ENTER IN FREE FORMAT.
00020:C
00021:C     X(I),Y(I)
00022:C             = COORDINATES OF NODES 1,N ENTERED COUNTER-CLOCKWISE
00023:C               IN FREE FORMAT.
00024:C
00025:C     J,K     = NODES, J, AND CORRESPONDING DIRECTIONS,K,
00026:C               AT WHICH DISPLACEMENTS ARE SPECIFIED(ZERO AND
00027:C               NON-ZERO). ENTER IN FREE FORMAT AND END BY INPUTTING
00028:C               0, 0.
00029:C
00030:C     J,K,F(2*J+K-2)
00031:C             = NODES,J, AND CORRESPONDING DIRECTIONS, K, AT WHICH
00032:C               NON-ZERO FORCES ARE SPECIFIED, AND SPECIFIED VALUES
00033:C               F. ENTER IN FREE FORMAT AND END BY INPUTTING 0, 0, 0.
00034:C
00035:C     LL      = NO. OF POINTS USED FOR NUMERICAL INTEGRATION.
00036:C               LL MAY BE 3,4,5, OR 6.
00037:C
00038:C     I,XF,YF
00039:C             = COORDINATES OF EACH INTERNAL POINT AT WHICH DIS-
00040:C               PLACEMENTS AND STRESSES ARE TO BE COMPUTED.
00041:C               ENTER IN FREE FORMAT, ONE POINT PER LINE. ALWAYS
00042:C               BEGIN LINE WITH I=1. ENTER I=INTEGER OTHER THAN ONE
00043:C               TO STOP.
00044:C
00045:C
00046:C*************************************************************************
00047:C
00048:       COMMON UR(203,200),NTBC(200,2)
00049:       COMMON R(6),W(6),W1(6),W2(6)
00050:       COMMON /CGAUSS/ UC(203,203),SAV(203)
00051:       DIMENSION X(100),Y(100),F(200)
00052:       DIMENSION TITLE(20)
00053:C
00054:       CALL CLOCK(START)
00055:       OPEN(UNIT=8,FILE='DATA')
00056:       OPEN(UNIT=10,FILE='RESULT')
00057:C
00058:C     INPUT DATA AND PRINT
00059:C
00060:       READ(8,1)TITLE
00061:1      FORMAT(20A4)
00062:C
00063:       READ(8,*)N
```

```
00064:       NN=N+N
00065:       NNP1=NN+1
00066:       NNP2=NN+2
00067:       NNP3=NN+3
00068:C
00069:       READ(8,*)E,PR
00070:C
00071:       READ(8,*)(X(I),Y(I),I=1,N)
00072:C
00073:       DO 2 I=1,NN
00074:       DO 2 K=1,2
00075:2      NTBC(I,K)=0
00076:C
00077:3      READ(8,*)J,K
00078:       IF(J.EQ.0) GO TO 4
00079:       NTBC(J,K)=1
00080:       GO TO 3
00081:4      CONTINUE
00082:C
00083:       DO 5 I=1,NN
00084:5      F(I)=0.0
00085:C
00086:6      READ(8,*)K,KK,RR
00087:       IF(K.EQ.0) GO TO 7
00088:       II=2*(K-1)+KK
00089:       F(II)=RR
00090:       GO TO 6
00091:7      CONTINUE
00092:C
00093:       READ(8,*)LL
00094:C
00095:       WRITE(10,8)TITLE
00096:8      FORMAT(/,20A4,/)
00097:C
00098:       WRITE(10,40)N
00099:40     FORMAT(/,5X,'NUMBER OF NODES = ',3X,I5)
00100:       WRITE(10,9)E,PR
00101:C
00102:9      FORMAT(/,5X,'YOUNGS MODULUS  = ',E12.6.//
00103:      +,5X,'POISSONS RATIO  = ',F10.4)
00104:C
00105:       WRITE(10,10)
00106:10     FORMAT(//,3X,'NODE NUMBER',6X,'NODAL COORDINATES AND BOUNDARY COND
00107:      +ITIONS')
00108:       DO 11 I=1,N
00109:11     WRITE(10,12)I,X(I),NTBC(I,1),Y(I),NTBC(I,2)
00110:12     FORMAT(/,3X,I5,5X,3(F15.5,I5))
00111:C
00112:C
00113:C  POINTS AND WEIGHTS FOR NUMERICAL INTEGRATION
00114:C
00115:       WRITE(10,13)LL
00116:13     FORMAT(//,'POINTS OF INTEGRATION',3X,I5)
00117:C
00118:       CALL CLOCK(FIN1)
00119:       IF(LL.EQ.3) THEN
00120:       R(1)=SQRT(0.6)
00121:       R(2)=0.0
00122:       R(3)=-R(1)
00123:       W(1)=5./9.
00124:       W(2)=8./9.
00125:       W(3)=5./9.
00126:C
00127:       ELSE IF(LL.EQ.4) THEN
00128:       R(1)=-0.861136311594O5
00129:       R(2)=-0.339981043584B5
```

```
00130:       R(3)=-R(2)
00131:       R(4)=-R(1)
00132:       W(1)=0.3478548451374S
00133:       W(2)=0.6521451548e254
00134:       W(3)=W(2)
00135:       W(4)=W(1)
00136:C
00137:       ELSE IF(LL.EQ.5) THEN
00138:       R(1)=-0.90617984593866
00139:       R(2)=-0.53846931010568
00140:       R(3)=0.0
00141:       R(4)=-R(2)
00142:       W(1)=0.23692688505618
00143:       W(2)=0.47862867049936
00144:       W(3)=0.56888888888888
00145:       W(4)=W(2)
00146:       W(5)=W(1)
00147:       R(5)=-R(1)
00148:C
00149:       ELSE IF(LL.EQ.6) THEN
00150:       R(1)=-0.93246951420315
00151:       R(2)=-0.66120938646626
00152:       R(3)=-0.2386191860831?
00153:       R(4)=-R(3)
00154:       R(5)=-R(2)
00155:       R(6)=-R(1)
00156:       W(1)=0.17132449237917
00157:       W(2)=0.3607615730481?
00158:       W(3)=0.46791393457269
00159:       W(4)=W(3)
00160:       W(5)=W(2)
00161:       W(6)=W(1)
00162:       END IF
00163:       DO 14 L=1,LL
00164:       W1(L)=1.-R(L)
00165:14     W2(L)=1.+R(L)
00166:C
00167:C  INITIALIZE MATRICES
00168:C
00169:       DO 15 I=1,NNP3
00170:       DO 15 J=1,NNP3
00171:       UC(J,I)=0.0
00172:       IF(I.GT.NN) GO TO 15
00173:       UR(J,I)=0.0
00174:15      CONTINUE
00175:C
00176:       DO 16 I=1,N
00177:       UC(2*I-1,NNP1)=-1.0
00178:       UC(2*I   ,NNP2)=-1.0
00179:       UR(NNP1,2*I-1)=1.0
00180:16     UR(NNP2,2*I   )=1.0
00181:C
00182:       DO 17 I=2,N
00183:       UC(2*I-1,NNP3)=Y(1)-Y(I)
00184:       UC(2*I   ,NNP3)=X(I)-X(1)
00185:       UR(NNP3,2*I-1)=Y(1)-Y(I)
00186:17     UR(NNP3,2*I   )=X(I)-X(1)
00187:C
00188:C     SOME CONSTANTS
00189:C
00190:       PR2=2.*PR
00191:       PR3=3.*PR
00192:       P7=-(3.-PR)/2.
00193:       PI=ACOS(-1.)
00194:C
00195:C  LOOP ON COLUMNS
```

```
00196:C
00197:       DO 19 J=1,N
00198:       JJ=2*J
00199:       IF(J.EQ.1) THEN
00200:         JM1=N
00201:         JJM2=NN
00202:       ELSE
00203:         JM1=J-1
00204:         JJM2=JJ-2
00205:       END IF
00206:       IF(J.EQ.N) THEN
00207:         JP1=1
00208:         JJP1=1
00209:       ELSE
00210:         JP1=J+1
00211:         JJP1=JJ+1
00212:       END IF
00213:C
00214:       UC(JJ-1,JJM2)=UC(JJ-1,JJM2)+2.-PR2
00215:       UC(JJ  ,JJM2-1)=UC(JJ  ,JJM2-1)-2.+PR2
00216:       UC(JJ-1,JJP1+1)=UC(JJ-1,JJP1+1)-2.+PR2
00217:       UC(JJ  ,JJP1)=UC(JJ  ,JJP1)+2.-PR2
00218:C
00219:       XM=X(J)-X(JM1)
00220:       YM=Y(J)-Y(JM1)
00221:       SF=SQRT(XM*XM+YM*YM)
00222:       CTF=YM/SF
00223:       STF=-XM/SF
00224:       XM=X(JP1)-X(J)
00225:       YM=Y(JP1)-Y(J)
00226:       SS=SQRT(XM*XM+YM*YM)
00227:       CTS=YM/SS
00228:       STS=-XM/SS
00229:       ARG=CTS*CTF+STS*STF
00230:       IF(ARG.GT.1.0) THEN
00231:       OM=PI*4.
00232:       ELSE
00233:        OM=(PI-ACOS(ARG))*4.
00234:       END IF
00235:C
00236:       AA=(2.+PR2)*(STS*CTS-STF*CTF)
00237:       BB=(2.+PR2)*(CTS*CTS-CTF*CTF)
00238:       CC=(2.-PR2)*LOG(SF/SS)
00239:C
00240:       UC(JJ-1,JJ-1)=OM+AA
00241:       UC(JJ,JJ-1)=-BB-CC
00242:       UC(JJ-1,JJ)=-BB+CC
00243:       UC(JJ,JJ)=OM-AA
00244:C
00245:       AA=1.-LOG(SF)
00246:       BB=STF*STF
00247:       CC=-CTF*STF
00248:       DD=CTF*CTF
00249:C
00250:       UR(JJ-1,JJ-1)=AA*(3.-PR)+BB*(1.+PR)
00251:       UR(JJ,JJ-1)=CC*(1.+PR)
00252:       UR(JJ-1,JJ)=UR(JJ,JJ-1)
00253:       UR(JJ,JJ)=AA*(3.-PR)+DD*(1.+PR)
00254:C
00255:       AA=1.-LOG(SS)
00256:       BB=STS*STS
00257:       CC=-CTS*STS
00258:       DD=CTS*CTS
00259:C
00260:       UR(JJ-1,JJP1)=AA*(3.-PR)+BB*(1.+PR)
00261:       UR(JJ,JJP1)=CC*(1.+PR)
```

```
00262:       UR(JJ-1,JJP1+1)=UR(JJ,JJP1)
00263:       UR(JJ,JJP1+1)=AA*(3.-PR)+DD*(1.+PR)
00264:C
00265:       XM=(X(J)+X(JM1))/2.
00266:       YM=(Y(J)+Y(JM1))/2.
00267:       B1=X(J)-XM
00268:       B2=Y(J)-YM
00269:       BB=B1*B1+B2*B2
00270:       P8=B2*(2.+PR2)
00271:       P9=B1*(2.+PR2)
00272:       P10=B2*(1.-PR)
00273:       P11=B1*(3.+PR)
00274:       P12=B1*(1.+PR3)
00275:       P13=B2*(3.+PR)
00276:       P14=B2*(1.+PR3)
00277:       P15=B1*(1.-PR)
00278:C
00279:C   LOOP ON ROWS
00280:C
00281:       DO 19 I=1,N
00282:       IF(I.EQ.J.OR.I.EQ.JM1) GO TO 19
00283:       II=2*I
00284:       A1=X(I)-XM
00285:       A2=Y(I)-YM
00286:       AA=A1*A1+A2*A2
00287:       AB=2.*(A1*B1+A2*B2)
00288:C
00289:C   LOOP ON POINTS OF INTEGRATION
00290:C
00291:       DO 18 L=1,LL
00292:       RR=AA-AB*R(L)+BB*R(L)*R(L)
00293:       A1B1=A1-B1*R(L)
00294:       UF3=A1B1/RR
00295:       UF6=A1B1*UF3
00296:       UF1=UF6*UF3
00297:       A1B1=A2-B2*R(L)
00298:       UF7=A1B1*UF3
00299:       UF4=A1B1/RR
00300:       UF8=A1B1*UF4
00301:       UF2=UF4*UF8
00302:       UF5=LOG(RR)
00303:C
00304:       EE=(P8*UF1+P9*UF2+P10*UF3-P11*UF4)*W(L)
00305:       UC(II-1,JJM2-1)=UC(II-1,JJM2-1)+W1(L)*EE
00306:       UC(II-1,JJ-1)=UC(II-1,JJ-1)+W2(L)*EE
00307:C
00308:       EE=(P9*UF1-P8*UF2-P12*UF3+P13*UF4)*W(L)
00309:       UC(II-1,JJM2)=UC(II-1,JJM2)+W1(L)*EE
00310:       UC(II-1,JJ)=UC(II-1,JJ)+W2(L)*EE
00311:C
00312:       EE=(P9*UF1-P8*UF2-P11*UF3+P14*UF4)*W(L)
00313:       UC(II,JJM2-1)=UC(II,JJM2-1)+W1(L)*EE
00314:       UC(II,JJ-1)=UC(II,JJ-1)+W2(L)*EE
00315:C
00316:       EE=(-P8*UF1-P9*UF2+P13*UF3-P15*UF4)*W(L)
00317:       UC(II,JJM2)=UC(II,JJM2)+EE*W1(L)
00318:       UC(II,JJ)=UC(II,JJ)+W2(L)*EE
00319:C
00320:       UR(II-1,JJ-1)=UR(II-1,JJ-1)+(P7*UF5+(1.+PR)*UF6)*W(L)/2.0
00321:       UR(II-1,JJ)=UR(II-1,JJ)+(1.+PR)*UF7*W(L)/2.0
00322:       UR(II,JJ-1)=UR(II-1,JJ)
00323:       UR(II,JJ)=UR(II,JJ)+(P7*UF5+(1.+PR)*UF8)*W(L)/2.0
00324:18      CONTINUE
00325:19      CONTINUE
00326:       CALL CLOCK(FIN2)
00327:C
```

```
00328:C     MODIFY UR
00329:C
00330:        DO 22 I=1,N
00331:        II=2*I
00332:        IIM1=II-1
00333:        A=1.0
00334:        DO 20 K=2,N
00335:        UR(IIM1,1)=UR(IIM1,1)+UR(IIM1,2*K-1)*A
00336:        UR(II  ,1)=UR(II  ,1)+UR(II  ,2*K-1)*A
00337:        UR(IIM1,2)=UR(IIM1,2)+UR(IIM1,2*K  )*A
00338:        UR(II  ,2)=UR(II  ,2)+UR(II  ,2*K  )*A
00339:20      A=-A
00340:        DO 21 K=2,N
00341:        UR(IIM1,2*K-1)=-UR(IIM1,2*K-3)+2.0*UR(IIM1,2*K-1)
00342:        UR(II  ,2*K-1)=-UR(II  ,2*K-3)+2.0*UR(II  ,2*K-1)
00343:        UR(IIM1,2*K  )=-UR(IIM1,2*K-2)+2.0*UR(IIM1,2*K  )
00344:21      UR(II  ,2*K  )=-UR(II  ,2*K-2)+2.0*UR(II  ,2*K  )
00345:22      CONTINUE
00346:C
00347:C     UNKNOWNS TO LEFT-HAND SIDE
00348:C
00349:        DO 24 I=1,N
00350:        DO 24 K=1,2
00351:        II=2*(I-1)+K
00352:        KK=NTBC(I,K)
00353:        IF(KK.EQ.0) GO TO 24
00354:        DO 23  L=1,NNP3
00355:        TEM=UR(L,II)
00356:        UR(L,II)=-UC(L,II)
00357:        UC(L,II)=-TEM
00358:23      CONTINUE
00359:24      CONTINUE
00360:C
00361:C    MULTIPLY KNOWNS BY MATRIX IN UR
00362:C
00363:        DO 25 I=1,NNP3
00364:        SAV(I)=0.0
00365:        DO 25 L=1,NN
00366:25      SAV(I)=SAV(I)+UR(I,L)*F(L)
00367:C
00368:C     SOLVE SYSTEM OF EQUATIONS
00369:C
00370:        CALL GAUSS(NNP3,IER)
00371:        CALL CLOCK(FIN3)
00372:C
00373:C     REARRANGE FORCES AND DISPLACEMENTS. OUTPUT RESULTS
00374:C
00375:        DO 26 I=1,N
00376:        DO 26 K=1,2
00377:        KK=NTBC(I,K)
00378:        II=2*(I-1)+K
00379:        IF(KK.EQ.1) THEN
00380:        TEM=F(II)
00381:        F(II)=SAV(II)
00382:        SAV(II)=TEM
00383:        ELSE
00384:         CONTINUE
00385:        END IF
00386:26      CONTINUE
00387:        WRITE(10,27)
00388:27      FORMAT(//,2X,'NODE NUMBER',28X,'DIRECTION  X',28X,'DIRECTION  Y',/
00389:     +,/,32X,'DISPLACEMENT',8X,'FORCE',15X,'DISPLACEMENT',8X,'FORCE')
00390:        G=E/((2.*(1.+PR))
00391:        DO 28 I=1,N
00392:        DIS1=SAV(2*I-1)/G
00393:        DIS2=SAV(2*I)/G
```

```
00394:28     WRITE(10,29)I,DIS1,F(2*I-1),DIS2,F(2*I)
00395:29     FORMAT(/,3X,I5,17X,2(5X,F12.6),6X,2(5X,F12.6))
00396:       CALL CLOCK(FIN4)
00397:C
00398:C     CALCULATIONS FOR FIELD POINTS.
00399:C
00400:       WRITE(10,39)
00401:39     FORMAT(/,5X,'RESULTS FOR FIELD POINTS',//,6X,'COORDINATES',
00402:      +15X,'DISPLACEMENTS',13X,'STRESSES',//)
00403:30     READ(8,*)IA,XF,YF
00404:       IF(IA.NE.1) GO TO 38
00405:       DO 31 I=1,NN
00406:       DO 31 K=1,5
00407:       UC(K,I)=0.0
00408:31     UR(K,I)=0.0
00409:       DO 32 J=1,N
00410:       JJ=2*J
00411:       IF(J.EQ.1) THEN
00412:         JJM2=NN
00413:         JJM3=NN-1
00414:         JM1=N
00415:       ELSE
00416:         JJM2=JJ-2
00417:         JJM3=JJ-3
00418:         JM1=J-1
00419:       END IF
00420:C
00421:       XM=(X(J)+X(JM1))/2.
00422:       YM=(Y(J)+Y(JM1))/2.
00423:       B1=X(J)-XM
00424:       B2=Y(J)-YM
00425:       BB=B1*B1+B2*B2
00426:       P8=B2*(2.+PR2)
00427:       P9=B1*(2.+PR2)
00428:       P10=B2*(1.-PR)
00429:       P11=B1*(3.+PR)
00430:       P12=B1*(1.+PR3)
00431:       P13=B2*(3.+PR)
00432:       P14=B2*(1.+PR3)
00433:       P15=B1*(1.-PR)
00434:C
00435:       A1=XF-XM
00436:       A2=YF-YM
00437:       AA=A1*A1+A2*A2
00438:       AB=2.*(A1*B1+A2*B2)
00439:C
00440:C    LOOP ON POINTS OF INTEGRATION
00441:C
00442:       DO 32 L=1,LL
00443:       RR=AA-AB*R(L)+BB*R(L)*R(L)
00444:       A1B1=A1-B1*R(L)
00445:       UF3 =A1B1/RR
00446:       UF6 =A1B1*UF3
00447:       UF1 =UF6*UF3
00448:       A1B1=A2-B2*R(L)
00449:       UF7 =A1B1*UF3
00450:       UF4 =A1B1/RR
00451:       UF8 =A1B1*UF4
00452:       UF2 =UF4*UF8
00453:       UF5 =LOG(RR)
00454:       UF9=UF3*UF3
00455:       UF10=UF4*UF4
00456:       UF11=1.0/RR
00457:       UF12=UF6*UF8*UF11
00458:       UF13=UF6*UF11
00459:       UF14=UF8*UF11
```

```
00460:C
00461:       EE=(P8*UF1+P9*UF2+P10*UF3-P11*UF4)*W(L)
00462:       UC(1,JJM2-1)=UC(1,JJM2-1)+W1(L)*EE
00463:       UC(1,JJ-1)=UC(1,JJ-1)+W2(L)*EE
00464:C
00465:       EE=(P9*UF1-P8*UF2-P12*UF3+P13*UF4)*W(L)
00466:       UC(1,JJM2)=UC(1,JJM2)+W1(L)*EE
00467:       UC(1,JJ)=UC(1,JJ)+W2(L)*EE
00468:C
00469:       EE=(P9*UF1-P8*UF2-P11*UF3+P14*UF4)*W(L)
00470:       UC(2,JJM2-1)=UC(2,JJM2-1)+W1(L)*EE
00471:       UC(2,JJ-1)=UC(2,JJ-1)+W2(L)*EE
00472:C
00473:       EE=(-P8*UF1-P9*UF2+P13*UF3-P15*UF4)*W(L)
00474:       UC(2,JJM2)=UC(2,JJM2)+EE*W1(L)
00475:       UC(2,JJ)=UC(2,JJ)+W2(L)*EE
00476:C
00477:       UR(1,JJ-1)=UR(1,JJ-1)+(P7*UF5+(1.+PR)*UF6)*W(L)/2.0
00478:       UR(1,JJ)=UR(1,JJ)+(1.+PR)*UF7*W(L)/2.0
00479:       UR(2,JJ-1)=UR(1,JJ)
00480:       UR(2,JJ)=UR(2,JJ)+(P7*UF5+(1.+PR)*UF8)*W(L)/2.0
00481:C
00482:       EE=(2.0+PR2)*UF1
00483:       UR(3,JJ-1)=UR(3,JJ-1)-(EE+(1.0-PR)*UF3)*W(L)
00484:       UR(4,JJ-1)=UR(4,JJ-1)+(EE-(1.0+3.0*PR)*UF3)*W(L)
00485:       UR(5,JJ  )=UR(5,JJ  )+(EE-(3.0+PR)*UF3)*W(L)
00486:C
00487:       EE=(2.0+PR2)*UF2
00488:       UR(3,JJ  )=UR(3,JJ  )+(EE-(1.0+3.0*PR)*UF4)*W(L)
00489:       UR(4,JJ  )=UR(4,JJ  )-(EE+(1.0-PR)*UF4)*W(L)
00490:       UR(5,JJ-1)=UR(5,JJ-1)+(EE-(3.0+PR)*UF4)*W(L)
00491:C
00492:C
00493:       EE=(2.*B2*(UF11+4.*UF13-8.*UF1*UF3)-4.*B1*UF7*(UF11-4.*UF9))
00494:      +*W(L)
00495:       UC(3,JJM3)=UC(3,JJM3)+EE*W1(L)
00496:       UC(3,JJ-1)=UC(3,JJ-1)+EE*W2(L)
00497:C
00498:       EE=(4.*B2*UF7*(UF11-4.0*UF9)-2.*B1*(UF11-8.0*UF12))*W(L)
00499:       UC(5,JJM3)=UC(5,JJM3)+EE*W1(L)
00500:       UC(5,JJ-1)=UC(5,JJ-1)+EE*W2(L)
00501:       UC(3,JJM2)=UC(3,JJM2)+EE*W1(L)
00502:       UC(3,JJ  )=UC(3,JJ  )+EE*W2(L)
00503:C
00504:       EE=((UF11-8.0*UF12)*2.*B2-UF7*(UF11-4.0*UF10)*4.*B1)*W(L)
00505:       UC(4,JJM3)=UC(4,JJM3)+EE*W1(L)
00506:       UC(4,JJ-1)=UC(4,JJ-1)+EE*W2(L)
00507:       UC(5,JJM2)=UC(5,JJM2)+EE*W1(L)
00508:       UC(5,JJ  )=UC(5,JJ  )+EE*W2(L)
00509:C
00510:C
00511:       EE=(4.*B2*UF7*(UF11-4.*UF10)-2.*B1*(UF11+4.*UF14-8.*UF2*UF4))
00512:      +*W(L)
00513:       UC(4,JJM2)=UC(4,JJM2)+EE*W1(L)
00514:       UC(4,JJ  )=UC(4,JJ  )+EE*W2(L)
00515:C
00516:32      CONTINUE
00517:C
00518:C     MODIFY UR AND OUTPUT FIELD RESULTS.
00519:C
00520:       A=1.0
00521:       DO 34 K=2,N
00522:       KK=2*K
00523:       KKM1=KK-1
00524:       DO 33 L=1,5
00525:       UR(L,1)=UR(L,1)+UR(L,KKM1)*A
```

```
00526:33      UR(L,2)=UR(L,2)+UR(L,M   )*A
00527:34      A=-A
00528:        DO 35 M=2,N
00529:        MM=2*M
00530:        MM1=MM-1
00531:        DO 35 L=1,5
00532:        UR(L,MM1)=-UR(L,M -3)+2.0*UR(L,M M1)
00533:35      UR(L,MM  )=-UR(L,MM-2)+2.0*UR(L,MM  )
00534:C
00535:        DIS1=0.0
00536:        DIS2=0.0
00537:        SIG1=0.0
00538:        SIG2=0.0
00539:        SIG3=0.0
00540:C
00541:        DO 36 I=1,NN
00542:        DIS1=DIS1-UC(1,I)*SAV(I)+UR(1,I)*F(I)
00543:        DIS2=DIS2-UC(2,I)*SAV(I)+UR(2,I)*F(I)
00544:        SIG1=SIG1-(1.0+PR)*UC(3,I)*SAV(I)+UR(3,I)*F(I)
00545:        SIG2=SIG2-(1.0+PR)*UC(4,I)*SAV(I)+UR(4,I)*F(I)
00546:36      SIG3=SIG3-(1.0+PR)*UC(5,I)*SAV(I)+UR(5,I)*F(I)
00547:        DIS1=DIS1/(8.0*PI)
00548:        DIS2=DIS2/(8.0*PI)
00549:        SIG1=SIG1/(8.0*PI)
00550:        SIG2=SIG2/(8.0*PI)
00551:        SIG3=SIG3/(8.0*PI)
00552:        WRITE(10,37)XF,YF,DIS1,DIS2,SIG1,SIG2,SIG3
00553:37      FORMAT(7(3X,F10.5),/)
00554:C
00555:        IF(IA.EQ.1) GO TO 30
00556:38      CALL CLOCK(FIN5)
00557:        WRITE(10,*)'CPU TIME AFTER READ AND PRINT          =',FIN1-START
00558:        WRITE(10,*)'CPU TIME AFTER CREATE UC AND UR        =',FIN2-START
00559:        WRITE(10,*)'CPU TIME AFTER SOLVING EQUATIONS       =',FIN3-START
00560:        WRITE(10,*)'CPU TIME AFTER WRITTING DISPLACEMENTS =',FIN4-START
00561:        WRITE(10,*)'TOTAL CPU TIME                         =',FIN5-START
00562:        CLOSE(UNIT=10)
00563:        CLOSE(UNIT=8)
00564:        CALL EXIT
00565:        END
00566:C
00567:C*******************************************************************************
00568:C
00569:C     SUBROUTINE GAUSS
00570:C
00571:C*******************************************************************************
00572:C
00573:        SUBROUTINE GAUSS(N,IER)
00574:C
00575:        COMMON /CGAUSS/ A(203,203),SAV(203)
00576:C
00577:        NM1=N-1
00578:        NP1=N+1
00579:C
00580:C     THE OUTER LOOP USES ELEMENTARY ROW OPERATIONS TO
00581:C     TRANSFORM A TO TRIANGULAR FORM
00582:C
00583:        DO 6 I=1,NM1
00584:C
00585:C     SEARCH FOR THE LARGEST ENTRY IN COLUMN I, ROWS I THROUGH N.
00586:C     IPIVOT IS THE ROW INDEX OF THE LARGEST ENTRY.
00587:C
00588:        PIVOT=0.0
00589:        DO 1 J=I,N
00590:        TEMP=ABS(A(J,I))
00591:        IF(PIVOT.GE.TEMP) GO TO 1
```

```
00592:       PIVOT=TEMP
00593:       IPIVOT=J
00594:1      CONTINUE
00595:       IF(PIVOT.EQ.0.0) GO TO 11
00596:       IF(IPIVOT.EQ.I) GO TO 3
00597:C
00598:C     INTERCHANGE ROW I AND ROW IPIVOT
00599:C
00600:       DO 2 K=I,N
00601:       TEMP=A(I,K)
00602:       A(I,K)=A(IPIVOT,K)
00603:       A(IPIVOT,K)=TEMP
00604:2      CONTINUE
00605:       TEMP=SAV(I)
00606:       SAV(I)=SAV(IPIVOT)
00607:       SAV(IPIVOT)=TEMP
00608:C
00609:C     ZEROS ENTRIES (I+1,I),(I+2,I),....,(N,I) IN MATRIX A
00610:C
00611:3      IP1=I+1
00612:       DO 5 K=IP1,N
00613:       Q=-A(K,I)/A(I,I)
00614:       A(K,I)=0.0
00615:       SAV(K)=Q*SAV(I)+SAV(K)
00616:       DO 4 J=IP1,N
00617:       A(K,J)=Q*A(I,J)+A(K,J)
00618:4      CONTINUE
00619:5      CONTINUE
00620:6      CONTINUE
00621:       IF(A(N,N).EQ.0.0) GO TO 11
00622:C
00623:C     BACKSOLVE TO GET SOLUTION TO AX=SAV
00624:C
00625:       SAV(N)=SAV(N)/A(N,N)
00626:       DO 8 K=1,NM1
00627:       Q=0.0
00628:       DO 7 J=1,K
00629:       Q=Q+A(N-K,NP1-J)*SAV(NP1-J)
00630:7      CONTINUE
00631:       SAV(N-K)=(SAV(N-K)-Q)/A(N-K,N-K)
00632:8      CONTINUE
00633:       IER=1
00634:       RETURN
00635:C
00636:C     ABNORMAL RETURN. THE MATRIX A MAY BE SINGULAR
00637:C
00638:11     IER=2
00639:       RETURN
00640:       END
BOTTOM
$
```

```
00001:C
00002:C*************************************************************************
00003:C
00004:      PROGRAM FEM
00005:C
00006:C     THIS PROGRAM EMPLOYS THE FINITE ELEMENT METHOD(USING
00007:C     THREE-NODE TRIANGULAR AND FOUR-NODE ISOPARAMETRIC
00008:C     QUADRILATERAL ELEMENTS) TO SOLVE PLANE STRESS PROBLEMS
00009:C     OF LINEAR ELASTICITY. UNIT THICNESS IS ASSUMED.
00010:C
00011:C     WE DEFINE THE FOLLOWING INPUT PARAMETERS,VARIABLES
00012:C     AND ARRAYS:
00013:C
00014:C     TITLE   = THE TITLE OF THE PROBLEM. THIS MUST BE WRITTEN
00015:C               ON ONE LINE IN COLUMNS 1 THROUGH 80.
00016:C     NMAT    = THE NUMBER OF MATERIALS OF WHICH THE BODY IS
00017:C               COMPOSED. ENTER IN FREE FORMAT.
00018:C     NNO     = THE TOTAL NUMBER OF NODES IN THE MODEL. ENTER IN
00019:C               FREE FORMAT.
00020:C     NDOF    = THE NUMBER OF DEGREES OF FREEDOM OF THE PROBLEM.
00021:C               THIS PARAMETER IS EQUAL TO 2 FOR PLANE PROBLEMS.ENTER IN
00022:C               FREE FORMAT.
00023:C     NEL     = THE NUMBER OF ELEMENTS USED. ENTER IN FREE FORMAT.
00024:C     NLO     = THE NUMBER OF LOAD CASES TO BE SOLVED FOR THE SAME
00025:C               GEOMETRY. ENTER IN FREE FORMAT.
00026:C
00027:C     E(I),PR(I)
00028:C             = MATERIAL PROPERTIES (E IS YOUNG'S MODULUS AND PR IS
00029:C               POISSON'S RATIO). ENTER FOR I=1,NMAT, ONE SET OF
00030:C               PROPERTIES PER LINE. ENTER E(I) IN COLUMNS 1 TROUGH 20
00031:C               IN E FORMAT AND PR(I) IN COLUMNS 21 TROUGH 40 IN F
00032:C               FORMAT.
00033:C
00034:C     MP(I),NEN(I),(NN(I,J),J=1,NEN(I))
00035:C             = MP(I) IS THE MATERIAL IDENTIFICATION OF THE ELEMENT
00036:C               (MP=1,2,...NMAT),NEN(I) IS THE NUMBER OF NODES OF THE
00037:C               ELEMENT, AND (NN(I,J),J=1,NEN(I)) ARE THE NUMBERS OF
00038:C               THE NODES WHICH FORM THE ELEMENT(LISTED COUNTER-
00039:C               CLOCKWISE).THESE PROPERTIES ARE READ IN FREE FORMAT.
00040:C
00041:C     X(I,1),NTBC(I,1),X(I,2),NTBC(I,2)
00042:C             = NODE PROPERTIES. X(I,1) IS THE COORDINATE X1 OF NODE I,
00043:C               NTBC(I,1) IS THE TYPE OF BOUNDARY CONDITION IN THE X1
00044:C               DIRECTION AT NODE I, X(I,2) IS THE COORDINATE X2 OF
00045:C               NODE I, NTBC(I,2) IS THE TYPE OF BOUNDARY CONDITION IN
00046:C               THE X2 DIRECTION AT NODE I. ENTER FOR I=1,NNO, ONE
00047:C               SET OF NODE PROPERTIES PER LINE. ENTER X(I,1) IN CO-
00048:C               LUMNS 1 THROUGH 15 IN F FORMAT, NTBC(I,1) IN COLUMNS
00049:C               16 THROUGH 20 IN I FORMAT, X(I,2) IN COLUMNS 21 THROUGH
00050:C               35 IN F FORMAT AND NTBC(I,2) IN COLUMNS 36 THROUGH 40
00051:C               IN I FORMAT.
00052:C
00053:C               THE BOUNDARY CONDITION CODE FOR NTBC(I,J) IS AS FOLLOWS
00054:C               NTBC(I,J) =-1 IF SPECIFIED ZERO DISPLACEMENT AT NODE I IN
00055:C               DIRECTION J.
00056:C               NTBC(I,J) = 0 IF SPECIFIED FORCE AT NODE I IN  DI-.
00057:C               RECTION J.
00058:C               NTBC(I,J) = 1 IF SPECIFIED NON-ZERO DISPLACEMENT AT
00059:C               NODE I IN DIRECTION K.
00060:C               THE DEFAULT VALUE FOR NTBC(I,J) IS ZERO.
00061:C
00062:C     LL      = NUMBER OF POINTS IN EACH DIRECTION USED FOR NUMERICAL
00063:C               INTEGRATION. LL MAY BE 2 OR 3.
00064:C
```

```
00065:C     ALSO, FOR EACH LOAD CASE,1 THROUGH NLO:
00066:C
00067:C     I,II,R = THE NODE NUMBER,DIRECTION AND VALUE OF EACH
00068:C              SPECIFIED NON-ZERO FORCE OR DISPLACEMENT. ENTER IN FREE
00069:C              FORMAT AND END BY INPUTTING 0,0,0.0.
00070:C
00071:C     R       = LOCAL COORDINATE FOR STRESS SOLUTION IN ELEMENTS TO
00072:C              BE DESIGNATED BELOW. R MUST LIE BETWEEN 0. AND 1. ENTER
00073:C              IN FREE FORMAT.
00074:C
00075:C              NOTE THAT R=0. IMPLIES THAT STRESSES ARE TO BE COMPUTED
00076:C              AT CENTER OF ELEMENTS ONLY WHEREAS R=1. IMPLIES THAT
00077:C              STRESSES WILL BE COMPUTED AT THE FOUR CORNERS.
00078:C
00079:C     NE1,NE2= INTERVALS OF ELEMENTS IN WHICH STRESSES ARE TO
00080:C              BE COMPUTED IN ASCENDING ORDER. STRESSES WILL BE COMPUTED
00081:C              IN ELEMENTS NE1 THROUGH NE2, NE1 THROUGH NE2,ETC. ENTER
00082:C              IN FREE FORMAT AND END BY INPUTTING 0,0.
00083:C
00084:C
00085:C*************************************************************************
00086:C
00087:       COMMON MEQ,MSD
00088:       COMMON / CPROFIL/ NMAT,NEL,NSD,NR,NNO,NDOF,NEQ,NEN(200)
00089:      +,NN(200,4),NTBC(200,2)
00090:       COMMON /DIAG/JDIAG(500),IDIAG(100)
00091:       COMMON / CRED/E(3),PR(3),G(3),MP(200),X(200,2)
00092:C
00093:       OPEN(UNIT=8,FILE='DATA1')
00094:       OPEN(UNIT=10,FILE='RESULT1')
00095:C
00096:       CALL CLOCK(START)
00097:       CALL RED1(NLO,LL)
00098:       CALL CLOCK(FIN1)
00099:C
00100:       CALL PROFIL1
00101:       CALL CLOCK(FIN2)
00102:C
00103:       CALL PFORM(LL)
00104:       CALL CLOCK(FIN3)
00105:C
00106:       CALL SOLVE(.TRUE.,.FALSE.,0)
00107:       CALL CLOCK(FIN4)
00108:C
00109:       DO 10 I=1,NLO
00110:       CALL MODIFY(I)
00111:       CALL SOLVE(.FALSE.,.TRUE.,I)
00112:       CALL STRESS
00113:10     CONTINUE
00114:       CALL CLOCK(FIN5)
00115:C
00116:       WRITE(10,*)'CPU TIME AFTER READ      = ',FIN1-START
00117:       WRITE(10,*)'CPU TIME AFTER PROFIL    = ',FIN2-START
00118:       WRITE(10,*)'CPU TIME AFTER PFORM     = ',FIN3-START
00119:       WRITE(10,*)'CPU TIME AFTER SOLVE1    = ',FIN4-START
00120:       WRITE(10,*)'TOTAL CPU TIME           = ',FIN5-START
00121:C
00122:       CLOSE(UNIT=10)
00123:       CLOSE(UNIT=8)
00124:C
00125:       CALL EXIT
00126:       END
00127:C
00128:C*************************************************************************
00129:C
00130:C     SUBROUTINE RED1
```

```
00131:C
00132:C*************************************************************************
00133:C
00134:      SUBROUTINE RED1(NLO,LL)
00135:C
00136:      COMMON /CPROFIL/ NMAT,NEL,NSD,NR,NNO,NDOF,NEQ,NEN(200)
00137:     +,NN(200,4),NTBC(200,2)
00138:      COMMON /DIAG/ JDIAG(500),IDIAG(100)
00139:      COMMON / CRED/ E(3),PR(3),G(3),MP(200),X(200,2)
00140:      DIMENSION TITLE(20)
00141:C
00142:      READ(8,1) TITLE
00143:1     FORMAT(20A4)
00144:C
00145:      READ(8,*) NMAT,NEL,NNO,NDOF,NLO
00146:C
00147:      READ(8,2)(E(I),PR(I),I=1,NMAT)
00148:2     FORMAT(E20.12,F20.12)
00149:C
00150:      DO 3 I=1,NEL
00151:      READ(8,*)MP(I),NEN(I),(NN(I,J),J=1,NEN(I))
00152:3     CONTINUE
00153:C
00154:      DO 4 I=1,NNO
00155:4     READ(8,5)(X(I,J),NTBC(I,J),J=1,NDOF)
00156:5     FORMAT(BZ,3(F15.5,I5))
00157:C
00158:      READ(8,*)LL
00159:C
00160:      WRITE(10,6) TITLE
00161:6     FORMAT(//,20A4,//)
00162:C
00163:      WRITE(10,7)NMAT,NEL,NNO,NDOF
00164:7     FORMAT(//,4X,'NO. OF MATERIALS',4X,'NO. OF ELEMENTS',
00165:     +4X,'NO. OF NODES',4X,'NO. OF DEGREES OF FREDOM',//,2(10X,I5)
00166:     +,2(13X,I5))
00167:C
00168:      WRITE(10,8)
00169:8     FORMAT(//,'MATERIAL NUMBER',4X,'YOUGS MODULUS',5X,
00170:     +'POISSON RATIO')
00171:C
00172:      WRITE(10,9)(I,E(I),PR(I),I=1,NMAT)
00173:9     FORMAT(//,I5,13X,E14.7,F20.6)
00174:C
00175:      WRITE(10,10)
00176:10    FORMAT(//,3X,'ELEMENT NUMBER',4X,'MATERIAL NUMBER',4X,
00177:     +'NUMBER OF NODES',4X,'NODE NUMBERS')
00178:C
00179:C
00180:      DO 13 I=1,NEL
00181:      WRITE(10,11)I,MP(I),NEN(I),(NN(I,J),J=1,NEN(I))
00182:11    FORMAT(/,4X,I5,2(15X,I5),7X,4(I5))
00183:      IF(NEN(I).LE.4) GO TO 13
00184:      WRITE(10,12)(NN(I,J),J=5,NEN(I))
00185:12    FORMAT(/,57X,4I5)
00186:13    CONTINUE
00187:C
00188:      WRITE(10,14)
00189:14    FORMAT(/,3X,'NODE NUMBER',6X,
00190:     +'NODAL COORDINATES AND BOUNDARY CONDITIONS')
00191:C
00192:      DO 15 I=1,NNO
00193:15    WRITE(10,16)I,(X(I,J),NTBC(I,J),J=1,NDOF)
00194:16    FORMAT(/,3X,I5,5X,3(F15.5,I5))
00195:C
00196:      WRITE(10,17)LL
```

```
00197:17    FORMAT(//,' NO. OF POINTS OF INTEGRATION IN EACH DIRECTION
00198:     +',I5,//)
00199:C
00200:      RETURN
00201:      END
00202:C
00203:C*******************************************************************************
00204:C
00205:C    SUBROUTINE PROFIL1
00206:C
00207:C*******************************************************************************
00208:C
00209:      SUBROUTINE PROFIL1
00210:C
00211:      COMMON /CPROFIL/ NMAT,NEL,NSD,NR,NNO,NDOF,NEQ,NEN(200)
00212:     +,NN(200,4),NTBC(200,2)
00213:      COMMON /DIAG/JDIAG(500),IDIAG(100)
00214:C
00215:C    COUNT EQUATIONS, INITIALIZE JDIAG, IDIAG
00216:C
00217:      NEQ=0
00218:      NSD=0
00219:      DO 1 N=1,NNO
00220:      DO 1 I=1,NDOF
00221:      J=NTBC(N,I)
00222:C
00223:      IF(J.LT.0) THEN
00224:      NTBC(N,I)=0
00225:C
00226:      ELSE IF(J.EQ.0) THEN
00227:      NEQ=NEQ+1
00228:      NTBC(N,I)=NEQ
00229:C
00230:      ELSE
00231:      NSD=NSD-1
00232:      NTBC(N,I)=NSD
00233:      ND=2*IABS(NSD)
00234:      IDIAG(ND-1)=20000
00235:      IDIAG(ND)=0
00236:      END IF
00237:1     CONTINUE
00238:C
00239:C    COMPUTE COLUMN LENGTHS OF STIFFNESS VECTOR
00240:C
00241:      DO 10 N=1,NEL
00242:      DO 9 I=1,NEN(N)
00243:      II=NN(N,I)
00244:      IF(II.EQ.0) GO TO 9
00245:      DO 8 K=1,NDOF
00246:C
00247:      KK=NTBC(II,K)
00248:      IF(KK)2,8,5
00249:C
00250:2     KK=IABS(KK)
00251:      DO 4 J=I,NEN(I)
00252:      JJ=NN(N,J)
00253:      IF(JJ.EQ.0) GO TO 4
00254:      DO 3 L=1,NDOF
00255:      LL=NTBC(JJ,L)
00256:      IF(LL.LE.0) GO TO 3
00257:      IDIAG(2*KK-1)=MINO(IDIAG(2*KK-1),LL)
00258:      IDIAG(2*KK)=MAXO(IDIAG(2*KK),LL)
00259:3     CONTINUE
00260:4     CONTINUE
00261:      GO TO 8
00262:C
```

```
00263:5      DO 7 J=1,NFN(N)
00264:       JJ=NN(N,J)
00265:       IF(JJ.EQ.0) GO TO 7
00266:C
00267:       DO 6 L=1,NDOF
00268:       LL=NTBC(JJ,L)
00269:C
00270:       IF(LL.LT.0) THEN
00271:       LL=IABS(LL)
00272:       IDIAG(2*LL-1)=MINO(IDIAG(2*LL-1),KK)
00273:       IDIAG(2*LL)=MAXO(IDIAG(2*LL),KK)
00274:C
00275:       ELSE IF(LL.GT.0) THEN
00276:       M=MAXO(KK,LL)
00277:       JDIAG(M)=MAXO(JDIAG(M),IABS(KK-LL))
00278:       ELSE
00279:       CONTINUE
00280:       END IF
00281:6      CONTINUE
00282:7      CONTINUE
00283:8      CONTINUE
00284:9      CONTINUE
00285:10     CONTINUE
00286:C
00287:C     COMPUTE TOTAL LENGTH OF STIFFNESS VECTOR
00288:C
00289:       NAD=1
00290:       JDIAG(1)=1
00291:       IF(NEQ.EQ.1) GO TO 12
00292:       DO 11 N=2,NEQ
00293:11     JDIAG(N)=JDIAG(N)+JDIAG(N-1)+1
00294:       NAD=JDIAG(NEQ)
00295:       WRITE(10,14)NEQ,NAD
00296:14     FORMAT(/,3X,'NO. OF EQUATIONS',I5,3X,'COMPONENTS OF STIFFNESS',
00297:      +I5)
00298:12     IF(NSD.EQ.0)RETURN
00299:C
00300:       NED=1
00301:       NSD=IABS(NSD)
00302:       IF(NSD.EQ.1) RETURN
00303:       DO 13 N=2,NSD
00304:13     IDIAG(2*N)=IDIAG(2*N-2)+1+IDIAG(2*N)-IDIAG(2*N-1)
00305:       NED=IDIAG(2*NSD)
00306:C
00307:       RETURN
00308:       END
00309:C
00310:C     FUNCTIONS MINO AND MAXO
00311:C
00312:       FUNCTION MINO(I,K)
00313:       IF(I.LE.K) THEN
00314:       MINO=I
00315:       ELSE
00316:       MINO=K
00317:       END IF
00318:       RETURN
00319:       END
00320:C
00321:       FUNCTION MAXO(I,K)
00322:       IF(I.GE.K) THEN
00323:       MAXO=I
00324:       ELSE
00325:       MAXO=K
00326:       END IF
00327:       RETURN
00328:       END
```

```
00329:C
00330:C*******************************************************************************
00331:C
00332:C     SUBROUTINE PFORM
00333:C
00334:C*******************************************************************************
00335:C
00336:       SUBROUTINE PFORM(LL)
00337:C
00338:       COMMON /CPROFIL/ NMAT,NEL,NSD,NR,NNO,NDOF,NEQ,NEN(200)
00339:      +,NN(200,4),NTBC(200,2)
00340:       COMMON /DIAG/ JDIAG(500),IDIAG(100)
00341:       COMMON /ABST/ A(7000),B(500),F(400)
00342:       COMMON /CRED/ E(3),PR(3),G(3),MP(200),X(200,2)
00343:       DIMENSION D(3),XE(4,2),NBC(4,2),S(8,8)
00344:C
00345:C     INITIALIZE A AND B
00346:C
00347:       DO 2 I=1,JDIAG(NEQ)
00348:2      A(I)=0.0
00349:       IF(NSD.EQ.0) GO TO 100
00350:       DO 3 I=1,IDIAG(2*NSD)
00351:3      B(I)=0.0
00352:C
00353:C    CALCULATE MATERIAL PROPERTIES
00354:C
00355:100    DO 4 N=1,NMAT
00356:       W1=PR(N)
00357:       E(N)=E(N)/(1.0-PR(N)**2)
00358:       PR(N)=E(N)*W1
00359:4      G(N)=E(N)*(1.0-W1)/2.0
00360:C
00361:C     LOOP ON THE ELEMENTS
00362:C
00363:       DO 10 N=1,NEL
00364:C
00365:C     LOCALIZE PROPERTIES IN ARRAYS XE(I,L),NBC(I,L),D(L)
00366:C
00367:       MM=MP(N)
00368:       D(1)=E(MM)
00369:       D(2)=PR(MM)
00370:       D(3)=G(MM)
00371:C
00372:       DO 8 I=1,NEN(N)
00373:       II=NN(N,I)
00374:       IF(II.NE.0) GO TO 6
00375:       DO 5 L=1,NDOF
00376:       XE(I,L)=0.0
00377:5      NBC(I,L)=0
00378:       GO TO 8
00379:6      DO 7 L=1,NDOF
00380:       XE(I,L)=X(II,L)
00381:       NBC(I,L)=NTBC(II,L)
00382:7      CONTINUE
00383:8      CONTINUE
00384:C
00385:C     INITIALIZE LOCAL STIFFNESS MATRIX
00386:C
00387:       DO 9  I=1,8
00388:       DO 9  J=1,8
00389:9      S(J,I)=0.0
00390:C
00391:C     CALCULATE STIFFNESS OF ELEMENT
00392:C
00393:       CALL ELEMENT(D,XE,S,NEN(N),NDOF,LL)
00394:C
```

```
00395:C     ADD THE VALUES OF LOCAL STIFFNESS INTO GLOBAL
00396:C
00397:       CALL ADDSTIF(S,NBC,NEN(N),NDOF)
00398:10     CONTINUE
00399:C
00400:C     REARRANGE THE ARRAY NTBC FOR REACTIONS
00401:C
00402:       NR=0
00403:       DO 11 I=1,NNO
00404:       DO 11 K=1,NDOF
00405:       KK=NTBC(I,K)
00406:       IF(KK.NE.0) GO TO 11
00407:       NR=NR+1
00408:       NTBC(I,K)=-NSD-NR
00409:11     CONTINUE
00410:C
00411:       RETURN
00412:       END
00413:C
00414:C*******************************************************************************
00415:C
00416:C     SUBROUTINE ELEMENT
00417:C
00418:C*******************************************************************************
00419:C
00420:       SUBROUTINE ELEMENT(D,XE,S,NIN,NDOF,LL)
00421:C       .
00422:       DIMENSION SG(9),WG(9),TG(9),XE(4,2),SHP(4,3)
00423:      +,S(8,8),D(3)
00424:C
00425:C     CALCULATE POINTS AND WEIGHTS FOR NUMERICAL INTEGRATION
00426:C
00427:       CALL PGAUSS(LL,SG,TG,WG)
00428:       LINT=LL*LL
00429:       DO 2 L=1,LINT
00430:C
00431:C     CALCULATE SHAPE FUNCTIONS AND THEIR DERIVATIVES AT
00432:C     THE POINTS OF INTEGRATION
00433:C
00434:       CALL SHAPE(SG(L),TG(L),XE,SHP,XSJ,NDOF,NIN)
00435:C
00436:C    CORRECT ELEMENT OF AREA
00437:C
00438:       XSJ=XSJ*WG(L)
00439:C
00440:C     CREATE PRODUCTS OF DERIVATIVES
00441:C
00442:       J1=1
00443:       DO 2 J=1,NIN
00444:       W11=SHP(J,1)*XSJ
00445:       W12=SHP(J,2)*XSJ
00446:       K1=J1
00447:C
00448:       DO 1 K=J,NIN
00449:       S(J1,K1)=S(J1,K1)+W11*SHP(K,1)
00450:       S(J1,K1+1)=S(J1,K1+1)+W11*SHP(K,2)
00451:       S(J1+1,K1)=S(J1+1,K1)+W12*SHP(K,1)
00452:       S(J1+1,K1+1)=S(J1+1,K1+1)+W12*SHP(K,2)
00453:1      K1=K1+NDOF
00454:2      J1=J1+NDOF
00455:       NSL=NIN*NDOF
00456:C
00457:C    REARRANGE PRODUCTS INTO STIFFNESS MATRIX
00458:C
00459:       DO 4 J=1,NSL,NDOF
00460:       DO 4 I=J,NSL,NDOF
```

```
00461:       W11=S(J,K)
00462:       W12=S(J,K+1)
00463:       W21=S(J+1,K)
00464:       W22=S(J+1,K+1)
00465:C
00466:       S(J,K)=D(1)*W11+D(3)*W22
00467:       S(J,K+1)=D(2)*W12+D(3)*W21
00468:       S(J+1,K)=D(2)*W21+D(3)*W12
00469:       S(J+1,K+1)=D(1)*W22+D(3)*W11
00470:C
00471:       S(K,J)=S(J,K)
00472:       S(K,J+1)=S(J+1,K)
00473:       S(K+1,J)=S(J,K+1)
00474:4      S(K+1,J+1)=S(J+1,K+1)
00475:C
00476:       RETURN
00477:       END
00478:C
00479:C*******************************************************************************
00480:C
00481:C     SUBROUTINE SHAPE
00482:C
00483:C*******************************************************************************
00484:C
00485:       SUBROUTINE SHAPE(SS,TT,XE,SHP,XSJ,NDOF,NIN)
00486:C
00487:       DIMENSION SHP(4,3),XE(4,2),S(4),T(4),XS(2,2)
00488:      +,SX(2,2)
00489:C
00490:       DATA S/-0.5,0.5,0.5,-0.5/,T/-0.5,-0.5,0.5,0.5/
00491:C
00492:C    FORM 4 NODE QUADRILATERAL SHAPE FUNCTIONS
00493:C
00494:       DO 1 I=1,4
00495:       SHP(I,3)=(0.5+S(I)*SS)*(0.5+T(I)*TT)
00496:       SHP(I,1)=S(I)*(0.5+T(I)*TT)
00497:1      SHP(I,2)=T(I)*(0.5+S(I)*SS)
00498:       IF(NIN.GE.4) GO TO 3
00499:C
00500:C     FORM TRIANGLE SHAPE FUNCTIONS BY ADDING THIRD AND FOURTH TOGETHER
00501:C
00502:       DO 2 I=1,3
00503:2      SHP(3,I)=SHP(3,I)+SHP(4,I)
00504:C
00505:C     CONSTRUCT JACOBIAN AND ITS INVERSE
00506:C
00507:3      DO 4 I=1,NDOF
00508:       DO 4 J=1,2
00509:       XS(I,J)=0.0
00510:C
00511:       DO 4  K=1,NIN
00512:4      XS(I,J)=XS(I,J)+XE(K,I)*SHP(K,J)
00513:C
00514:       XSJ=XS(1,1)*XS(2,2)-XS(1,2)*XS(2,1)
00515:C
00516:       SX(1,1)=XS(2,2)/XSJ
00517:       SX(2,2)=XS(1,1)/XSJ
00518:       SX(1,2)=-XS(1,2)/XSJ
00519:       SX(2,1)=-XS(2,1)/XSJ
00520:C
00521:C     FORM GLOBAL DERIVATIVES
00522:C
00523:       DO 5 I=1,NIN
00524:       TP=SHP(I,1)*SX(1,1)+SHP(I,2)*SX(2,1)
00525:       SHP(I,2)=SHP(I,1)*SX(1,2)+SHP(I,2)*SX(2,2)
00526:5      SHP(I,1)=TP
```

```
00527:C
00528:       RETURN
00529:       END
00530:C
00531:C*******************************************************************************
00532:C
00533:C     SUBROUTINE GAUSS
00534:C
00535:C*******************************************************************************
00536:C
00537:       SUBROUTINE PGAUSS(L,R,Z,W)
00538:C
00539:       DIMENSION LR(9),LZ(9),LW(9)
00540:      +,R(9),Z(9),W(9)
00541:       DATA LR/-1,1,1,-1,0,1,0,-1,0/,LZ/-1,-1,1,1,-1,0,1,0,0/
00542:       DATA LW/4*25,4*40,64/
00543:C
00544:C     GAUSS POINTS AND WEIGHTS FOR TWO DIMENSIONS
00545:C
00546:       GO TO (2,4),L
00547:C
00548:C      2*2 INTEGRATION
00549:C
00550:2      G=1./SQRT(3.)
00551:       DO 3 LI=1,4
00552:       R(LI)=G*LR(LI)
00553:       Z(LI)=G*LZ(LI)
00554:3      W(LI)=1.
00555:       RETURN
00556:C
00557:C      3*3  INTEGRATION
00558:C
00559:4      G=SQRT(0.6)
00560:       H=1./81.
00561:       DO 5 LI=1,9
00562:       R(LI)=G*LR(LI)
00563:       Z(LI)=G*LZ(LI)
00564:5      W(LI)=H*LW(LI)
00565:C
00566:       RETURN
00567:       END
00568:C
00569:C*******************************************************************************
00570:C
00571:C     SUBROUTINE ADDSTIF
00572:C
00573:C*******************************************************************************
00574:C
00575:       SUBROUTINE ADDSTIF(S,NBC,NIN,NDOF)
00576:C
00577:       COMMON /ABST/ A(7000),B(500),F(400)
00578:       COMMON /DIAG/ JDIAG(500),IDIAG(100)
00579:       DIMENSION S(8,8),NBC(4,2)
00580:C
00581:       JC=0
00582:       DO 3 J=1,NIN
00583:       DO 3 JJ=1,NDOF
00584:       JC=JC+1
00585:       K=NBC(J,JJ)
00586:       IF(K.LT.0) THEN
00587:C
00588:C     LOCAL STIFFNESS IN B FOR SPECIFIED NON-ZERO DISPLACEMENTS
00589:C
00590:       K=IABS(K)
00591:       IF(K.EQ.1) THEN
00592:        L=1
```

```
00593:        ELSE
00594:        L=IDIAG(2*(I-1))+1
00595:       END IF
00596:       LL=IDIAG(2*I-1)
00597:       L=L-LL
00598:       IC=0
00599:       DO 1 I=1,NIN
00600:       DO 1 II=1,NDOF
00601:       IC=IC+1
00602:       M=NBC(I,II)
00603:       IF(M.LE.0) GO TO 1
00604:       M=L+M
00605:       B(M)=B(M)+S(IC,JC)
00606:1      CONTINUE
00607:C
00608:C     LOCAL STIFFNESS IN A FOR SPECIFIED FORCE
00609:C
00610:       ELSE IF(K.GT.0) THEN
00611:       L=JDIAG(K)-K
00612:       IC=0
00613:       DO 2 I=1,NIN
00614:       DO 2 II=1,NDOF
00615:       IC=IC+1
00616:       M=NBC(I,II)
00617:       IF(M.GT.K.OR.M.LE.0) GO TO 2
00618:       M=M+L
00619:       A(M)=A(M)+S(IC,JC)
00620:2      CONTINUE
00621:       END IF
00622:3      CONTINUE
00623:C
00624:       RETURN
00625:       END
00626:C
00627:C*******************************************************************************
00628:C
00629:C     SUBROUTINE SOLVE
00630:C
00631:C*******************************************************************************
00632:C
00633:       SUBROUTINE SOLVE(AFAC,BACK,LL)
00634:C
00635:       LOGICAL AFAC,BACK
00636:       COMMON /ABST/ A(7000),C(500),B(400)
00637:       COMMON /DIAG/ JDIAG(500),IDIAG(100)
00638:       COMMON /CPROFIL/ NMAT,NEL,NSD,NR,NNO,NDOF,NEQ,NEN(200)
00639:      +,NN(200,4),NTBC(200,2)
00640:       DIMENSION R(2)
00641:C
00642:C     PUT STIFFNESS A(I) INTO TRIANGULAR FORM
00643:C
00644:       AENGY=0.0
00645:       JR=0
00646:C
00647:C     LOOP ON COLUMNS
00648:C
00649:       DO 6 J=1,NEQ
00650:       JD=JDIAG(J)
00651:       JH=JD-JR
00652:       IS=J-JH+2
00653:       IF(JH-2)6,3,1
00654:1      IF(.NOT.AFAC) GO TO 5
00655:       IE=J-1
00656:       I=JR+2
00657:C
00658:C     LOOP ON ROWS
```

```
00659:C
00660:       DO 2 I=IS,IE
00661:       IR=JDIAG(I-1)
00662:       ID=JDIAG(I)
00663:       IH=MIN0(ID-IR-1,I-IS+1)
00664:       IF(IH.GT.0)A(I)=A(I)-DOT(A(K-IH),A(ID-IH),IH)
00665:2      K=K+1
00666:3      IF(.NOT.AFAC) GO TO 5
00667:       IR=JR+1
00668:       IE=JD-1
00669:       K=J-JD
00670:       DO 4 I=IR,IE
00671:       ID=JDIAG(K+I)
00672:       IF(A(ID).EQ.0.0)GO TO 4
00673:       D=A(I)
00674:       A(I)=A(I)/A(ID)
00675:       A(JD)=A(JD)-D*A(I)
00676:4      CONTINUE
00677:5      IF(BACK)B(J)=B(J)-DOT(A(JR+1),B(IS-1),JH-1)
00678:6      JR=JD
00679:       IF(.NOT.BACK) RETURN
00680:C
00681:C     SOLVE FOR LOAD VECTOR IN B
00682:C
00683:       DO 7 I=1,NEQ
00684:       ID=JDIAG(I)
00685:       IF(A(ID).NE.0.0) B(I)=B(I)/A(ID)
00686:7      AENGY=AENGY+B(I)*B(I)*A(ID)
00687:       J=NEQ
00688:       JD=JDIAG(J)
00689:8      D=B(J)
00690:       J=J-1
00691:       IF(J.LE.0) GO TO 11
00692:       JR=JDIAG(J)
00693:       IF(JD-JR.LE.1) GO TO 10
00694:       IS=J-JD+JR+2
00695:       K=JR-IS+1
00696:       DO 9 I=IS,J
00697:9      B(I)=B(I)-A(I+K)*D
00698:10     JD=JR
00699:       GO TO 8
00700:11     CONTINUE
00701:       WRITE(10,12)LL
00702:12     FORMAT(//,3X,'FINAL DISPLACEMENT FOR LOAD NO.',
00703:      +2X,I5,//,3X,'NODE NUMBER',10X,'DISPLACEMENT X',8X,
00704:      +'DISPLACEMENT Y')
00705:       CALL CLOCK(FINISH)
00706:C
00707:C     PRINT DISPLACEMENTS
00708:C
00709:       DO 14 J=1,NNO
00710:       DO 13 JJ=1,NDOF
00711:       JR=NTBC(J,JJ)
00712:        IF(JR.LT.-NSD) THEN
00713:       R(JJ)=0.0
00714:       ELSE IF(JR.GT.0) THEN
00715:       R(JJ)=B(JR)
00716:        ELSE
00717:       R(JJ)=B(NEQ+IABS(JR))
00718:        END IF
00719:13     CONTINUE
00720:14     WRITE(10,15)J,R(1),R(2)
00721:15     FORMAT(/,5X,I5,12X,F13.7,9X,F13.7)
00722:C
00723:       RETURN
00724:       END
```

```
00725:C
00726:C     FUNCTION DOT
00727:C
00728:       FUNCTION DOT(A,B,N)
00729:       DIMENSION A(N),B(N)
00730:       DOT=0.0
00731:       DO 1 I=1,N
00732:1      DOT=DOT+A(I)*B(I)
00733:       RETURN
00734:       END
00735:C
00736:C*******************************************************************************
00737:C
00738:C     SUBROUTINE MODIFY
00739:C
00740:C*******************************************************************************
00741:C
00742:       SUBROUTINE MODIFY(LL)
00743:C
00744:       COMMON /ABST/ A(7000),B(500),F(400)
00745:       COMMON /DIAG/ JDIAG(500),IDIAG(100)
00746:       COMMON /CPROFIL/ NMAT,NEL,NSD,NR,NNO,NDOF,NEQ,NEN(200)
00747:      +,NN(200,4),NTBC(200,2)
00748:C
00749:C     READ AND PRINT NON-ZERO SPECIFIED FORCES AND DISPLACEMENTS
00750:C
00751:       DO 1 K=1,(NEQ+2*NSD+NR)
00752:1      F(K)=0.0
00753:       WRITE(10,2)LL
00754:2      FORMAT(1H,//,5X,'LOAD CASE NO.',2X,I5,//,
00755:      +5X,'NON-ZERO SPECIFIED DISPLACEMENTS OR FORCES',
00756:      +//,5X,'NODE NUMBER',2X,'COORDINATE NO.',5X,'CONDITION AND VALUE')
00757:3      READ(8,*) I,II,R
00758:       IF(I.EQ.0) GO TO 6
00759:       IK=NTBC(I,II)
00760:C
00761:       IF(IK.GT.0) THEN
00762:       F(IK)=R
00763:       WRITE(10,4)I,II,R
00764:4      FORMAT(//,8X,I5,8X,I5,10X,'FORCE',12X,F10.5)
00765:C
00766:       ELSE IF(IK.LT.-NSD) THEN
00767:       WRITE(10,*)'ERROR  NODE     ',I,'    ',II,'  IS ZERO DISPLACEMENT'
00768:C
00769:       ELSE
00770:       IK=IABS(IK)
00771:       F(IK+NEQ)=R
00772:       WRITE(10,5)I,II,R
00773:5      FORMAT(//,8X,I5,8X,I5,10X,'DISPLACEMENT',5X,F10.5)
00774:       END IF
00775:       GO TO 3
00776:C
00777:C     MODIFY RIGHT HAND SIDE OF EQUATION FOR SPECIFIED NON-ZERO
00778:C     DISPLACEMENTS
00779:C
00780:6      IF(NSD.EQ.0)RETURN
00781:       DO 8 I=1,NEQ
00782:       JR=0
00783:       DO 7 J=1,NSD
00784:       JJ=IDIAG(2*J-1)
00785:       II=IDIAG(2*J)
00786:       JH=II-JR
00787:       JK=JJ-I
00788:       IF(JK.GT.0) GO TO 7
00789:       IF(JK.LT.0.AND.I.GT.(JJ+JH-1)) GO TO 7
00790:       IF(JK.EQ.0) THEN
```

```
00791:       F(I)=F(I)-F(NEQ+J)*B(JR+1)
00792:       ELSE
00793:       F(I)=F(I)-F(NEQ+J)*B(JR+1+I-JJ)
00794:       END IF
00795:7      JR=JR
00796:8      CONTINUE
00797:C
00798:       RETURN
00799:       END
00800:C
00801:C*******************************************************************************
00802:C
00803:C     SUBROUTINE STRESS
00804:C
00805:C*******************************************************************************
00806:C
00807:       SUBROUTINE STRESS
00808:C
00809:       COMMON / CPROFIL/ NMAT,NEL,NSD,NR,NNO,NDOF,NEQ,NEN(200)
00810:      +,NN(200,4),NTBC(200,2)
00811:       COMMON /CRED/ E(3),PR(3),G(3),MP(200),X(200,2)
00812:       COMMON /ABST/ A(7000),B(500),F(400)
00813:       DIMENSION D(3),XE(4,2),UE(8),STR(3),SIG(3)
00814:      +,SG(4),TG(4),SHP(4,3),S(8,8)
00815:C
00816:C
00817:C     LOCALIZE MATERIAL PROPERTIES,NODAL COORDINATES
00818:C     AND NODAL DISPLACEMENTS FOR EACH ELEMENT
00819:C
00820:       IK=0
00821:       READ(8,*)R
00822:       READ(8,*)NE1,NE2
00823:       SG(1)=-R
00824:       SG(2)=R
00825:       SG(3)=R
00826:       SG(4)=-R
00827:       TG(1)=-R
00828:       TG(2)=-R
00829:       TG(3)=R
00830:       TG(4)=R
00831:       WRITE(10,1)
00832:1      FORMAT(//,'ELEMENT NO.',3X,'COORDINATES',20X,'STRESSES')
00833:C
00834:       DO 10 I=1,NEL
00835:       IF(IK.EQ.1) THEN
00836:          READ(8,*)NE1,NE2
00837:          IK=0
00838:          ELSE
00839:          CONTINUE
00840:       END IF
00841:       IF(NE2.EQ.I)IK=1
00842:       M=MP(I)
00843:       D(1)=E(M)
00844:       D(2)=PR(M)
00845:       D(3)=G(M)
00846:       NIN=NEN(I)
00847:C
00848:       DO 2 J=1,NIN
00849:       JJ=NN(I,J)
00850:       J2=2*(J-1)
00851:       DO 2 K=1,NDOF
00852:       J2=J2+1
00853:       IF(JJ.EQ.0) THEN
00854:       XE(J,K)=0.0
00855:       UE(J2)=0.0
00856:       ELSE
```

```
00857:        XE(J,K)=X(JJ,K)
00858:        KK=NTBC(JJ,K)
00859:        IF(KK.LT.-NSD) THEN
00860:          UE(J2)=0.0
00861:          ELSE IF(KK.GT.0) THEN
00862:          UE(J2)=F(KK)
00863:          ELSE
00864:        UE(J2)=F(NEQ+IABS(KK))
00865:        END IF
00866:        END IF
00867:2       CONTINUE
00868:C
00869:C     CALCULATE REACTIONS
00870:C
00871:        M=0
00872:        DO 3 LL=1,8
00873:        DO 3 LLL=1,8
00874:3       S(LL,LLL)=0.0
00875:        DO 5 J=1,NEN(I)
00876:        JJ=NN(I,J)
00877:        J2=2*(J-1)
00878:        DO 5 K=1,NDOF
00879:        J2=J2+1
00880:        KK=NTBC(JJ,K)
00881:        IF(KK.GT.0) GO TO 5
00882:        IF(M.EQ.0)CALL ELEMENT(D,XE,S,NEN(I),NDOF,3)
00883:        M=1
00884:        MC=NEQ+NSD+IABS(KK)
00885:        DO 4 L=1,8
00886:4       F(MC)=F(MC)+S(J2,L)*UE(L)
00887:5       CONTINUE
00888:C
00889:C     CALCULATE STRAINS AND STRESSES
00890:C
00891:        IF(NE1.EQ.0) GO TO 10
00892:        IF(I.LT.NE1.OR.I.GT.NE2) GO TO  10
00893:        LL=4
00894:        IF(NIN.EQ.3.OR.R.EQ.0.0) LL=1
00895:        DO 8 L=1,LL
00896:C
00897:C     CALCULATE SHAPE FUNCTIONS AND THEIR DERIVATIVES
00898:C
00899:        CALL SHAPE(SG(L),TG(L),XE,SHP,XSJ,NDOF,NIN)
00900:        DO 6 K=1,3
00901:6       STR(K)=0.0
00902:        XC=0.0
00903:        YC=0.0
00904:        DO 7 J=1,NEN(I)
00905:        J2=2*J
00906:        J1=J2-1
00907:        XC=XC+SHP(J,3)*XE(J,1)
00908:        YC=YC+SHP(J,3)*XE(J,2)
00909:        STR(1)=STR(1)+SHP(J,1)*UE(J1)
00910:        STR(2)=STR(2)+SHP(J,2)*UE(J2)
00911:7       STR(3)=STR(3)+SHP(J,1)*UE(J2)+SHP(J,2)*UE(J1)
00912:C
00913:        SIG(1)=D(1)*STR(1)+D(2)*STR(2)
00914:        SIG(2)=D(2)*STR(1)+D(1)*STR(2)
00915:        SIG(3)=D(3)*STR(3)
00916:C
00917:8       WRITE(10,9)I,XC,YC,SIG(1),SIG(2),SIG(3)
00918:9       FORMAT(/,I5,5(4X,F12.6))
00919:10       CONTINUE
00920:C
00921:C     OUTPUT OF REACTIONS
00922:C
```

```
00923:       WRITE(10,11)
00924:11     FORMAT(//,3X,'NODE NO.',3X,'DIRECTION',3X,'REACTION',/)
00925:       DO 13 I=1,NND
00926:       DO 13 K=1,NDOF
00927:       KK=NTBC(I,K)
00928:       IF(KK.GT.0) GO TO 13
00929:       MC=NEQ+NSD+IABS(KK)
00930:       R=F(MC)
00931:       WRITE(10,12)I,K,R
00932:12     FORMAT(3X,I5,6X,I5,3X,F12.6,/)
00933:13     CONTINUE
00934:C
00935:       RETURN
00936:       END
BOTTOM
$
```

```
00001:C
00002:C
00003:C*************************************************************************
00004:C
00005:       PROGRAM FEMBOU
00006:C
00007:C     THIS PROGRAM EMPLOYS A COMBINATION OF THE FINITE ELEMENT
00008:C     AND DIRECT BOUNDARY ELEMENT METHODS FOR THE SOLUTION OF
00009:C     PLANE STRESS PROBLEMS OF LINEAR ELASTICITY. UNIT THICKNESS
00010:C     IS ASSUMED. FINITE ELEMENTS MAY BE DEFINED BY THREE OR FOUR
00011:C     NODES AND BEM-GENERATED ELEMENTS MAY BE DEFINED BY ANY ODD
00012:C     NUMBER OF NODES GREATER THAN FOUR.
00013:C
00014:C     WE DEFINE THE FOLLOWING INPUT PARAMETERS,VARIABLES
00015:C     AND ARRAYS:
00016:C
00017:C     TITLE  = THE TITLE OF THE PROBLEM. THIS MUST BE WRITTEN
00018:C              ON ONE LINE IN COLUMNS 1 THROUGH 80.
00019:C     NMAT   = THE NUMBER OF MATERIALS OF WHICH THE BODY IS
00020:C              COMPOSED. ENTER IN FREE FORMAT.
00021:C     NNO    = THE TOTAL NUMBER OF NODES IN THE MODEL. ENTER IN
00022:C              FREE FORMAT.
00023:C     NDOF   = THE NUMBER OF DEGREES OF FREEDOM OF THE PROBLEM.
00024:C              THIS PARAMETER IS EQUAL TO 2 FOR PLANE PROBLEMS.ENTER IN
00025:C              FREE FORMAT.
00026:C     NEL    = THE NUMBER OF ELEMENTS USED. ENTER IN FREE FORMAT.
00027:C     NLO    = THE NUMBER OF LOAD CASES TO BE SOLVED FOR THE SAME
00028:C              GEOMETRY. ENTER IN FREE FORMAT.
00029:C
00030:C     E(I),PR(I)
00031:C            = MATERIAL PROPERTIES (E IS YOUNG'S MODULUS AND PR IS
00032:C              POISSON'S RATIO). ENTER FOR I=1,NMAT, ONE SET OF
00033:C              PROPERTIES PER LINE. ENTER E(I) IN COLUMNS 1 TROUGH 20
00034:C              IN E FORMAT AND PR(I) IN COLUMNS 21 TROUGH 40 IN F
00035:C              FORMAT.
00036:C
00037:C     MP(I),NEN(I),(NN(I,J),J=1,NEN(I))
00038:C            = MP(I) IS THE MATERIAL IDENTIFICATION OF THE ELEMENT
00039:C              (MP=1,2,...NMAT),NEN(I) IS THE NUMBER OF NODES OF THE
00040:C              ELEMENT, AND (NN(I,J),J=1,NEN(I)) ARE THE NUMBERS OF
00041:C              THE NODES WHICH FORM THE ELEMENT(LISTED COUNTER-
00042:C              CLOCKWISE).THESE PROPERTIES ARE READ IN FREE FORMAT.
00043:C
00044:C     X(I,1),NTBC(I,1),X(I,2),NTBC(I,2)
00045:C            = NODE PROPERTIES. X(I,1) IS THE COORDINATE X1 OF NODE I,
00046:C              NTBC(I,1) IS THE TYPE OF BOUNDARY CONDITION IN THE X1
00047:C              DIRECTION AT NODE I, X(I,2) IS THE COORDINATE X2 OF
00048:C              NODE I, NTBC(I,2) IS THE TYPE OF BOUNDARY CONDITION IN
00049:C              THE X2 DIRECTION AT NODE I. ENTER FOR I=1,NNO, ONE
00050:C              SET OF NODE PROPERTIES PER LINE. ENTER X(I,1) IN CO-
00051:C              LUMNS 1 THROUGH 15 IN F FORMAT, NTBC(I,1) IN COLUMNS
00052:C              16 THROUGH 20 IN I FORMAT, X(I,2) IN COLUMNS 21 THROUGH
00053:C              35 IN F FORMAT AND NTBC(I,2) IN COLUMNS 36 THROUGH 40
00054:C              IN I FORMAT.
00055:C
00056:C              THE BOUNDARY CONDITION CODE FOR NTBC(I,J) IS AS FOLLOWS
00057:C              NTBC(I,J) =-1 IF SPECIFIED ZERO DISPLACEMENT AT NODE I IN
00058:C              DIRECTION J.
00059:C              NTBC(I,J) = 0 IF SPECIFIED FORCE AT NODE I IN  DI-.
00060:C              RECTION J.
00061:C              NTBC(I,J) = 1 IF SPECIFIED NON-ZERO DISPLACEMENT AT
00062:C              NODE I IN DIRECTION J.
00063:C              THE DEFAULT VALUE FOR NTBC(I,J) IS ZERO.
```

```
00064:C
00065:C
00066:C     ALSO, FOR EACH LOAD CASE,1 THROUGH NLO:
00067:C
00068:C     I,II,R = THE NODE NUMBER,DIRECTION AND VALUE OF EACH
00069:C              SPECIFIED NON-ZERO FORCE OR DISPLACEMENT. ENTER IN FREE
00070:C              FORMAT AND END BY INPUTTING 0,0,0.0.
00071:C
00072:C     R        = LOCAL COORDINATE FOR STRESS SOLUTION IN FINITE ELEMENTS TO
00073:C              BE DESIGNATED BELOW. R MUST LIE BETWEEN 0. AND 1. ENTER
00074:C              IN FREE FORMAT.
00075:C
00076:C              NOTE THAT R=0. IMPLIES THAT STRESSES ARE TO BE COMPUTED
00077:C              AT CENTER OF ELEMENTS ONLY WHEREAS R=1. IMPLIES THAT
00078:C              STRESSES WILL BE COMPUTED AT THE FOUR CORNERS.
00079:C
00080:C
00081:C     NE1,NE2= INTERVALS OF FINITE ELEMENTS IN WHICH STRESSES ARE TO
00082:C              BE COMPUTED IN ASCENDING ORDER. STRESSES WILL BE COMPUTED
00083:C              IN ELEMENTS NE1 THROUGH NE2, NE1 THROUGH NE2,ETC. ENTER
00084:C              IN FREE FORMAT AND END BY INPUTTING 0,0.
00085:C
00086:C     I,XF,YF
00087:C
00088:C            = COORDINATES OF EACH INTERNAL POINT IN BEM-GENERATED ELEMENTS
00089:C              AT WHICH DISPLACEMENTS AND STRESSES ARE TO BE COMPUTED.
00090:C              ENTER IN FREE FORMAT, ONE POINT PER LINE: ALWAYS BEGIN LINE
00091:C              WITH I=1. ENTER I=INTEGER OTHER THAN ONE TO STOP.
00092:C
00093:C
00094:C
00095:C*************************************************************************
00096:C
00097:       COMMON / CPROFIL/ NMAT,NEL,NSD,NR,NNO,NDOF,NEQ,NEN(100)
00098:      +,NN(100,30),NTBC(200,2)
00099:       COMMON /DIAG/JDIAG(300),IDIAG(100)
00100:       COMMON / CRED/E(3),PR(3),G(3),MP(100),X(200,2)
00101:C
00102:       OPEN(UNIT=8,FILE='DATA1')
00103:       OPEN(UNIT=10,FILE='RESULT1')
00104:C
00105:       CALL CLOCK(START)
00106:       CALL RED1(NLO)
00107:       CALL CLOCK(FIN1)
00108:C
00109:       CALL PROFIL1
00110:       CALL CLOCK(FIN2)
00111:C
00112:       CALL PFORM
00113:       CALL CLOCK(FIN3)
00114:C
00115:       CALL SOLVE(.TRUE.,.FALSE.,0)
00116:       CALL CLOCK(FIN4)
00117:       DO 10 I=1,NLO
00118:C
00119:       CALL MODIFY(I)
00120:C
00121:       CALL SOLVE(.FALSE.,.TRUE.,I)
00122:       CALL STRESS
00123:10     CONTINUE
00124:C
00125:       CALL CLOCK(FIN5)
00126:       WRITE(10,*)'CPU TIME AFTER READ      =',FIN1-START
00127:       WRITE(10,*)'CPU TIME AFTER PROFIL    =',FIN2-START
00128:       WRITE(10,*)'CPU TIME AFTER PFORM     =',FIN3-START
00129:       WRITE(10,*)'CPU TIME AFTER SOLVE(1) =',FIN4-START
```

```
00130:       WRITE(10,*)'TOTAL CPU TIME        =',FIND-START
00131:C
00132:       CLOSE(UNIT=10)
00133:       CLOSE(UNIT=8)
00134:       CALL EXIT
00135:       END
00136:C
00137:C*************************************************************************
00138:C
00139:C     SUBROUTINE RED1
00140:C
00141:C*************************************************************************
00142:C
00143:       SUBROUTINE RED1(NLO)
00144:C
00145:       COMMON /CPROFIL/ NMAT,NEL,NSD,NR,NNO,NDOF,NED,NEN(100)
00146:      +,NN(100,30),NTBC(200,2)
00147:       COMMON /DIAG/ JDIAG(300),IDIAG(100)
00148:       COMMON / CRED/ E(3),PR(3),G(3),MP(100),X(200,2)
00149:       DIMENSION TITLE(20)
00150:C
00151:       READ(8,1) TITLE
00152:1      FORMAT(20A4)
00153:C
00154:       READ(8,*) NMAT,NEL,NNO,NDOF,NLO
00155:C
00156:       READ(8,2)(E(I),PR(I),I=1,NMAT)
00157:2      FORMAT(E20.12,F20.12)
00158:C
00159:       DO 3 I=1,NEL
00160:       READ(8,*)MP(I),NEN(I),(NN(I,J),J=1,NEN(I))
00161:3      CONTINUE
00162:C
00163:       DO 4 I=1,NNO
00164:4      READ(8,5)(X(I,J),NTBC(I,J),J=1,NDOF)
00165:5      FORMAT(BZ,3(F15.5,I5))
00166:C
00167:C     PRINT DATA
00168:C
00169:       WRITE(10,6) TITLE
00170:6      FORMAT(//,20A4,//)
00171:C
00172:       WRITE(10,8)
00173:8      FORMAT(//,'MATERIAL NUMBER',4X,'YOUGS MODULUS',5X,
00174:      +'POISSON RATIO')
00175:       WRITE(10,9)(I,E(I),PR(I),I=1,NMAT)
00176:9      FORMAT(//,I5,13X,E14.7,F20.6)
00177:C
00178:       WRITE(10,10)
00179:10     FORMAT(//,3X,'ELEMENT NUMBER',4X,'MATERIAL NUMBER',4X,
00180:      +'NUMBER OF NODES',4X,'NODE NUMBERS')
00181:C
00182:       DO 13 I=1,NEL
00183:       IF(NEN(I).EQ.3) THEN
00184:       WRITE(10,11)I,MP(I),NEN(I),(NN(I,J),J=1,3)
00185:       ELSE
00186:       WRITE(10,11)I,MP(I),NEN(I),(NN(I,J),J=1,4)
00187:11     FORMAT(/,4X,I5,2(15X,I5),7X,4(I5))
00188:       END IF
00189:       IF(NEN(I).LE.4) GO TO 13
00190:       WRITE(10,12)(NN(I,J),J=5,NEN(I))
00191:12     FORMAT(/,56X,4I5)
00192:13     CONTINUE
00193:C
00194:       WRITE(10,14)
00195:14     FORMAT(/,3X,'NODE NUMBER',6X,
```

```
00194:      +'NODAL COORDINATES AND BOUNDARY CONDITIONS')
00197:       DO 15 I=1,NNO
00198:15     WRITE(10,16)I,(X(I,J),NTBC(I,J),J=1,NDOF)
00199:16     FORMAT(/,3X,I5,5X,3(F15.5,I5))
00200:C
00201:       RETURN
00202:       END
00203:C
00204:C*******************************************************************************
00205:C
00206:C     SUBROUTINE PROFIL1
00207:C
00208:C*******************************************************************************
00209:C
00210:       SUBROUTINE PROFIL1
00211:C
00212:       COMMON /CPROFIL/ NMAT,NEL,NSD,NR,NNO,NDOF,NEQ,NEN(100)
00213:      +,NN(100,30),NTBC(200,2)
00214:       COMMON /DIAG/JDIAG(300),IDIAG(100)
00215:C
00216:C     COUNT EQUATIONS, INITIALIZE JDIAG, IDIAG
00217:C
00218:       NEQ=0
00219:       NSD=0
00220:       DO 1 N=1,NNO
00221:       DO 1 I=1,NDOF
00222:       J=NTBC(N,I)
00223:C
00224:       IF(J.LT.0) THEN
00225:       NTBC(N,I)=0
00226:C
00227:       ELSE IF(J.EQ.0) THEN
00228:       NEQ=NEQ+1
00229:       NTBC(N,I)=NEQ
00230:C
00231:       ELSE
00232:       NSD=NSD-1
00233:       NTBC(N,I)=NSD
00234:       ND=2*IABS(NSD)
00235:       IDIAG(ND-1)=20000
00236:       IDIAG(ND)=0
00237:       END IF
00238:1      CONTINUE
00239:C
00240:C     COMPUTE COLUMN LENGTHS OF STIFFNESS VECTOR
00241:C
00242:       DO 10 N=1,NEL
00243:       DO 9 I=1,NEN(N)
00244:       II=NN(N,I)
00245:       IF(II.EQ.0) GO TO 9
00246:       DO 8 K=1,NDOF
00247:C
00248:       KK=NTBC(II,K)
00249:       IF(KK)2,8,5
00250:C
00251:2      KK=IABS(KK)
00252:       DO 4 J=I,NEN(I)
00253:       JJ=NN(N,J)
00254:       IF(JJ.EQ.0) GO TO 4
00255:       DO 3 L=1,NDOF
00256:       LL=NTBC(JJ,L)
00257:       IF(LL.LE.0) GO TO 3
00258:       IDIAG(2*KK-1)=MIN0(IDIAG(2*KK-1),LL)
00259:       IDIAG(2*KK)=MAX0(IDIAG(2*KK),LL)
00260:3      CONTINUE
00261:4      CONTINUE
```

```
00262:        GO TO 8
00263:C
00264:5       DO 7 J=1,NEN(N)
00265:        JJ=NN(N,J)
00266:        IF(JJ.EQ.0) GO TO 7
00267:C
00268:        DO 6 L=1,NDOF
00269:        LL=NTBC(JJ,L)
00270:C
00271:        IF(LL.LT.0) THEN
00272:        LL=IABS(LL)
00273:        IDIAG(2*LL-1)=MINO(IDIAG(2*LL-1),KK)
00274:        IDIAG(2*LL)=MAXO(IDIAG(2*LL),KK)
00275:C
00276:        ELSE IF(LL.GT.0) THEN
00277:        M=MAXO(KK,LL)
00278:        JDIAG(M)=MAXO(JDIAG(M),IABS(KK-LL))
00279:        ELSE
00280:        CONTINUE
00281:        END IF
00282:6       CONTINUE
00283:7       CONTINUE
00284:8       CONTINUE
00285:9       CONTINUE
00286:10      CONTINUE
00287:C
00288:C     COMPUTE TOTAL LENGTH OF STIFFNESS VECTOR
00289:C
00290:        NAD=1
00291:        JDIAG(1)=1
00292:        IF(NEQ.EQ.1) GO TO 12
00293:        DO 11 N=2,NEQ
00294:11      JDIAG(N)=JDIAG(N)+JDIAG(N-1)+1
00295:        NAD=JDIAG(NEQ)
00296:        WRITE(10,14) NEQ,JDIAG(NEQ)
00297:14      FORMAT(/,'NO. OF EQUATIONS',I5,3X,'COMPONENTS OF STIFFNESS',
00298:       +I5)
00299:12      IF(NSD.EQ.0)RETURN
00300:C
00301:        NED=1
00302:        NSD=IABS(NSD)
00303:        IF(NSD.EQ.1) RETURN
00304:        DO 13 N=2,NSD
00305:13      IDIAG(2*N)=IDIAG(2*N-2)+1+IDIAG(2*N)-IDIAG(2*N-1)
00306:        NED=IDIAG(2*NSD)
00307:C
00308:        RETURN
00309:        END
00310:C
00311:C     FUNCTIONS MINO AND MAXO
00312:C
00313:        FUNCTION MINO(I,K)
00314:        IF(I.LE.K) THEN
00315:        MINO=I
00316:        ELSE
00317:        MINO=K
00318:        END IF
00319:        RETURN
00320:        END
00321:C
00322:        FUNCTION MAXO(I,K)
00323:        IF(I.GE.K) THEN
00324:        MAXO=I
00325:        ELSE
00326:        MAXO=K
00327:        END IF
```

```
00328:       RETURN
00329:       END
00330:C
00331:C*******************************************************************************
00332:C
00333:C    SUBROUTINE PFORM
00334:C
00335:C*******************************************************************************
00336:C
00337:       SUBROUTINE PFORM
00338:C
00339:       COMMON /CPROFIL/ NMAT,NEL,NSD,NR,NNO,NDOF,NEQ,NEN(100)
00340:      +,NN(100,30),NTBC(200,2)
00341:       COMMON /DIAG/ JDIAG(300),IDIAG(100)
00342:       COMMON /ABST/ A(7000),B(500),F(500)
00343:       COMMON /CRED/ E(3),PR(3),G(3),MP(100),X(200,2)
00344:       COMMON /CBOUN/ SB(3364)
00345:       DIMENSION D(3),XE(58),NBC(58),S(8,8)
00346:C
00347:       LL=3
00348:       WRITE(10,1)LL
00349:1      FORMAT(//,'NO. OF POINTS OF INTEGRATION ON EACH DIRECTION',
00350:      +I5,//)
00351:C
00352:C    INITIALIZE A AND B
00353:C
00354:       DO 2 I=1,JDIAG(NEQ)
00355:2      A(I)=0.0
00356:       IF(NSD.EQ.0) GO TO 4
00357:       DO 3 I=1,IDIAG(2*NSD)
00358:3      B(I)=0.0
00359:C
00360:C   CALCULATE MATERIAL PROPERTIES
00361:C
00362:4      DO 5 N=1,NMAT
00363:       W1=PR(N)
00364:       E(N)=E(N)/(1.0-PR(N)**2)
00365:       PR(N)=E(N)*W1
00366:5      G(N)=E(N)*(1.0-W1)/2.0
00367:C
00368:C    LOOP ON THE ELEMENTS
00369:C
00370:       OPEN(UNIT=9,FILE='LOCAL')
00371:C
00372:       DO 10 N=1,NEL
00373:C
00374:C    LOCALIZE PROPERTIES IN LOCAL ARRAYS XE(I),NBC(I),D(L)
00375:C
00376:       MM=MP(N)
00377:       D(1)=E(MM)
00378:       D(2)=PR(MM)
00379:       D(3)=G(MM)
00380:C
00381:       DO 8 I=1,NEN(N)
00382:       II=NN(N,I)
00383:       IF(II.NE.0) GO TO 6
00384:       NBC(I)=0
00385:       NBC(I+NEN(N))=0
00386:       XE(I)=0.0
00387:       XE(I+NEN(N))=0.0
00388:       GO TO 8
00389:6      XE(I)=X(II,1)
00390:       XE(I+NEN(N))=X(II,2)
00391:       NBC(I)=NTBC(II,1)
00392:       NBC(I+NEN(N))=NTBC(II,2)
00393:8      CONTINUE
```

```
00394:C
00395:C     INITIALIZE LOCAL STIFFNESS MATRIX
00396:C
00397:       IF(NEN(N).LE.4) THEN
00398:       DO 9  I=1,8
00399:       DO 9 K=1,8
00400:9      S(K,I)=0.0
00401:C
00402:C     CALCULATE STIFFNESS OF FINITE ELEMENTS
00403:C
00404:       CALL ELEMENT(D,XE,S,NEN(N),NDOF,LL)
00405:       CALL ADDSTIF(S,NBC,NEN(N),NDOF,8)
00406:C
00407:       ELSE
00408:       NN1=2*NEN(N)
00409:       NN12=NN1*NN1
00410:C
00411:C      CALCULATE STIFFNESS OF SUPER-ELEMENTS
00412:C
00413:       CALL BOUN(NEN(N),NN1,D,XE,SB,B,F,1)
00414:       CALL ADDSTIF(SB,NBC,NEN(N),NDOF,NN1)
00415:       WRITE(9,*)(SB(I),I=1,NN12)
00416:       END IF
00417:10     CONTINUE
00418:C
00419:C
00420:C     REARRANGE THE ARRAY NTBC FOR REACTIONS
00421:C
00422:       NR=0
00423:       DO 11 I=1,NNO
00424:       DO 11 K=1,NDOF
00425:       KK=NTBC(I,K)
00426:       IF(KK.NE.0) GO TO 11
00427:       NR=NR+1
00428:       NTBC(I,K)=-NSD-NR
00429:11     CONTINUE
00430:C
00431:       RETURN
00432:       END
00433:C
00434:C*******************************************************************************
00435:C
00436:C     SUBROUTINE ELEMENT
00437:C
00438:C*******************************************************************************
00439:C
00440:       SUBROUTINE ELEMENT(D,XE,S,NIN,NDOF,LL)
00441:C
00442:       DIMENSION SG(9),WG(9),TG(9),XE(NIN,2),SHF(4,3)
00443:      +,S(8,8),D(3)
00444:C
00445:C      CALCULATE POINTS AND WEIGTHS FOR NUMERICAL INTEGRATION
00446:C
00447:       CALL PGAUSS(LL,SG,TG,WG)
00448:       LINT=LL*LL
00449:       DO 2 L=1,LINT
00450:C
00451:C      CALCULATE SHAPE FUNCTIONS AND THEIR DERIVATIVES AT
00452:C     THE POINTS OF INTEGRATION
00453:C
00454:       CALL SHAPE(SG(L),TG(L),XE,SHF,XSJ,NDOF,NIN)
00455:C
00456:C    CORRECT ELEMENT OF AREA
00457:C
00458:       XSJ=XSJ*WG(L)
00459:C
```

```
00460:C   CREATE PRODUCT OF DERIVATIVES AND STIFFNESS
00461:C
00462:       J1=1
00463:       DO 2 J=1,NIN
00464:       W11=SHP(J,1)*XSJ
00465:       W12=SHP(J,2)*XSJ
00466:       K1=J1
00467:C
00468:       DO 1 K=J,NIN
00469:       S(J1,K1)=S(J1,K1)+W11*SHP(K,1)
00470:       S(J1,K1+1)=S(J1,K1+1)+W11*SHP(K,2)
00471:       S(J1+1,K1)=S(J1+1,K1)+W12*SHP(K,1)
00472:       S(J1+1,K1+1)=S(J1+1,K1+1)+W12*SHP(K,2)
00473:1      K1=K1+NDOF
00474:2      J1=J1+NDOF
00475:       NSL=NIN*NDOF
00476:C
00477:C   REARRANGE PRODUCTS INTO STIFFNESS MATRIX
00478:C
00479:       DO 4 J=1,NSL,NDOF
00480:       DO 4 K=J,NSL,NDOF
00481:       W11=S(J,K)
00482:       W12=S(J,K+1)
00483:       W21=S(J+1,K)
00484:       W22=S(J+1,K+1)
00485:C
00486:       S(J,K)=D(1)*W11+D(3)*W22
00487:       S(J,K+1)=D(2)*W12+D(3)*W21
00488:       S(J+1,K)=D(2)*W21+D(3)*W12
00489:       S(J+1,K+1)=D(1)*W22+D(3)*W11
00490:C
00491:       S(K,J)=S(J,K)
00492:       S(K,J+1)=S(J+1,K)
00493:       S(K+1,J)=S(J,K+1)
00494:4      S(K+1,J+1)=S(J+1,K+1)
00495:C
00496:       RETURN
00497:       END
00498:C
00499:C*******************************************************************************
00500:C
00501:C    SUBROUTINE SHAPE
00502:C
00503:C*******************************************************************************
00504:C
00505:       SUBROUTINE SHAPE(SS,TT,XE,SHP,XSJ,NDOF,NIN)
00506:C
00507:       DIMENSION SHP(4,3),XE(NIN,2),S(4),T(4),XS(2,2)
00508:      +,SX(2,2)
00509:       DATA S/-0.5,0.5,0.5,-0.5/,T/-0.5,-0.5,0.5,0.5/
00510:C
00511:C   FORM 4 NODE QUADRILATERAL SHAPE FUNCTIONS
00512:C
00513:       DO 1 I=1,4
00514:       SHP(I,3)=(0.5+S(I)*SS)*(0.5+T(I)*TT)
00515:       SHP(I,1)=S(I)*(0.5+T(I)*TT)
00516:1      SHP(I,2)=T(I)*(0.5+S(I)*SS)
00517:       IF(NIN.GE.4) GO TO 3
00518:C
00519:C   FORM TRIANGLE BY ADDING THIRD AND FOURTH TOGETHER
00520:C
00521:       DO 2 I=1,3
00522:2      SHP(3,I)=SHP(3,I)+SHP(4,I)
00523:C
00524:C    CONSTRUCT JACOBIAN AND ITS INVERSE
00525:C
```

```
00526:3       DO 4 I=1,NDOF
00527:        DO 4 J=1,2
00528:        XS(I,J)=0.0
00529:C
00530:        DO 4  K=1,NIN
00531:4       XS(I,J)=XS(I,J)+XE(K,I)*SHP(K,J)
00532:C
00533:        XSJ=XS(1,1)*XS(2,2)-XS(1,2)*XS(2,1)
00534:C
00535:        SX(1,1)=XS(2,2)/XSJ
00536:        SX(2,2)=XS(1,1)/XSJ
00537:        SX(1,2)=-XS(1,2)/XSJ
00538:        SX(2,1)=-XS(2,1)/XSJ
00539:C
00540:C     FORM GLOBAL DERIVATIVES
00541:C
00542:        DO 5 I=1,NIN
00543:        TP=SHP(I,1)*SX(1,1)+SHP(I,2)*SX(2,1)
00544:        SHP(I,2)=SHP(I,1)*SX(1,2)+SHP(I,2)*SX(2,2)
00545:5       SHP(I,1)=TP
00546:C
00547:        RETURN
00548:        END
00549:C
00550:C*******************************************************************************
00551:C
00552:C     SUBROUTINE PGAUSS
00553:C
00554:C*******************************************************************************
00555:C
00556:        SUBROUTINE PGAUSS(L,R,Z,W)
00557:C
00558:        DIMENSION LR(9),LZ(9),LW(9)
00559:       +,R(9),Z(9),W(9)
00560:        DATA LR/-1,1,1,-1,0,1,0,-1,0/,LZ/-1,-1,1,1,-1,0,1,0,0/
00561:        DATA LW/4*25,4*40,64/
00562:C
00563:C       3*3  INTEGRATION
00564:C
00565:        G=SQRT(0.6)
00566:        H=1./81.
00567:        DO 5 LI=1,9
00568:        R(LI)=G*LR(LI)
00569:        Z(LI)=G*LZ(LI)
00570:5       W(LI)=H*LW(LI)
00571:C
00572:        RETURN
00573:        END
00574:C
00575:C*******************************************************************************
00576:C
00577:C     SUBROUTINE ADDSTIF
00578:C
00579:C*******************************************************************************
00580:C
00581:        SUBROUTINE ADDSTIF(S,NBC,NIN,NDOF,NN1)
00582:C
00583:        COMMON /ABST/ A(7000),B(500),F(500)
00584:        COMMON /DIAG/ JDIAG(300),IDIAG(100)
00585:        DIMENSION S(NN1,NN1),NBC(NIN,2)
00586:C
00587:        JC=0
00588:        DO 3 J=1,NIN
00589:        DO 3 JJ=1,NDOF
00590:        JC=JC+1
00591:        K=NBC(J,JJ)
```

```
00592:       IF(K.LT.0) THEN
00593:C
00594:C     LOCAL STIFFNESS IN B FOR SPECIFIED NON-ZERO DISPLACEMENTS
00595:C
00596:       K=IABS(K)
00597:       IF(K.EQ.1) THEN
00598:        L=1
00599:        ELSE
00600:        L=IDIAG(2*(K-1))+1
00601:       END IF
00602:       LL=IDIAG(2*K-1)
00603:       L=L-LL
00604:       IC=0
00605:       DO 1 I=1,NIN
00606:       DO 1 II=1,NDOF
00607:       IC=IC+1
00608:       M=NBC(I,II)
00609:       IF(M.LE.0) GO TO 1
00610:       M=L+M
00611:       B(M)=B(M)+S(IC,JC)
00612:1      CONTINUE
00613:C
00614:C     LOCAL STIFFNESS IN A FOR SPECIFIED FORCE
00615:C
00616:       ELSE IF(K.GT.0) THEN
00617:       L=JDIAG(K)-K
00618:       IC=0
00619:       DO 2 I=1,NIN
00620:       DO 2 II=1,NDOF
00621:       IC=IC+1
00622:       M=NBC(I,II)
00623:       IF(M.GT.K.OR.M.LE.0) GO TO 2
00624:       M=M+L
00625:       A(M)=A(M)+S(IC,JC)
00626:2      CONTINUE
00627:       END IF
00628:3      CONTINUE
00629:C
00630:       RETURN
00631:       END
00632:C
00633:C*******************************************************************************
00634:C
00635:C     SUBROUTINE SOLVE
00636:C
00637:C*******************************************************************************
00638:C
00639:       SUBROUTINE SOLVE(AFAC,BACK,LL)
00640:C
00641:       LOGICAL AFAC,BACK
00642:       COMMON /ABST/ A(7000),C(500),B(500)
00643:       COMMON /DIAG/ JDIAG(300),IDIAG(100)
00644:       COMMON /CPROFIL/ NMAT,NEL,NSD,NR,NNO,NDOF,NEQ,NEN(100)
00645:      +,NN(100,30),NTBC(200,2)
00646:       DIMENSION R(2)
00647:C
00648:C     PUT STIFFNESS A IN TRIANGULAR FORM
00649:C
00650:       AENGY=0.0
00651:       JR=0
00652:C
00653:C     LOOP ON COLUMNS
00654:C
00655:       DO 6 J=1,NEQ
00656:       JD=JDIAG(J)
00657:       JH=JD-JR
```

```
00658:       IS=J-JH+2
00659:       IF(JH-2)6,3,1
00660:1      IF(.NOT.AFAC) GO TO 5
00661:       IE=J-1
00662:       K=JR+2
00663:C
00664:C     LOOP ON ROWS
00665:C
00666:       DO 2 I=IS,IE
00667:       IR=JDIAG(I-1)
00668:       ID=JDIAG(I)
00669:       IH=MINO(ID-IR-1,I-IS+1)
00670:       IF(IH.GT.0)A(K)=A(K)-DOT(A(K-IH),A(ID-IH),IH)
00671:2      K=K+1
00672:3      IF(.NOT.AFAC) GO TO 5
00673:       IR=JR+1
00674:       IE=JD-1
00675:       K=J-JD
00676:       DO 4 I=IR,IE
00677:       ID=JDIAG(K+I)
00678:       IF(A(ID).EQ.0.0)GO TO 4
00679:       D=A(I)
00680:       A(I)=A(I)/A(ID)
00681:       A(JD)=A(JD)-D*A(I)
00682:4      CONTINUE
00683:5      IF(BACK)B(J)=B(J)-DOT(A(JR+1),B(IS-1),JH-1)
00684:6      JR=JD
00685:       IF(.NOT.BACK) RETURN
00686:C
00687:C     SOLVE FOR LOAD VECTOR IN B
00688:C
00689:       DO 7 I=1,NEQ
00690:       ID=JDIAG(I)
00691:       IF(A(ID).NE.0.0) B(I)=B(I)/A(ID)
00692:7      AENGY=AENGY+B(I)*B(I)*A(ID)
00693:       J=NEQ
00694:       JD=JDIAG(J)
00695:8      D=B(J)
00696:       J=J-1
00697:       IF(J.LE.0) GO TO 11
00698:       JR=JDIAG(J)
00699:       IF(JD-JR.LE.1) GO TO 10
00700:       IS=J-JD+JR+2
00701:       K=JR-IS+1
00702:       DO 9 I=IS,J
00703:9      B(I)=B(I)-A(I+K)*D
00704:10     JD=JR
00705:       GO TO 8
00706:11     CONTINUE
00707:       WRITE(10,12)LL
00708:12     FORMAT(//,3X,'FINAL DISPLACEMENT FOR LOAD NO.',
00709:      +2X,I5,//,3X,'NODE NUMBER',10X,'DISPLACEMENT X',8X,
00710:      +'DISPLACEMENT Y')
00711:       CALL CLOCK(FINISH)
00712:C
00713:C     PRINT DISPLACEMENTS
00714:C
00715:       DO 15 J=1,NNO
00716:       DO 13 JJ=1,NDOF
00717:       JR=NTBC(J,JJ)
00718:        IF(JR.LT.-NSD) THEN
00719:       R(JJ)=0.0
00720:       ELSE IF(JR.GT.0) THEN
00721:       R(JJ)=B(JR)
00722:        ELSE
00723:       R(JJ)=B(NEQ+IABS(JR))
```

```
00724:        END IF
00725:13      CONTINUE
00726:15      WRITE(10,16)J,R(1),R(2)
00727:16      FORMAT(/,5X,I5,12X,F13.7,9X,F13.7)
00728:C
00729:        RETURN
00730:        END
00731:C
00732:C     FUNCTION DOT
00733:C
00734:        FUNCTION DOT(A,B,N)
00735:        DIMENSION A(N),B(N)
00736:        DOT=0.0
00737:        DO 1 I=1,N
00738:1       DOT=DOT+A(I)*B(I)
00739:        RETURN
00740:        END
00741:C
00742:C*******************************************************************************
00743:C
00744:C     SUBROUTINE MODIFY
00745:C
00746:C*******************************************************************************
00747:C
00748:        SUBROUTINE MODIFY(LL)
00749:C
00750:        COMMON /ABST/ A(7000),B(500),F(500)
00751:        COMMON /DIAG/ JDIAG(300),IDIAG(100)
00752:        COMMON /CPROFIL/ NMAT,NEL,NSD,NR,NNO,NDOF,NEQ,NEN(100)
00753:       +,NN(100,30),NTBC(200,2)
00754:C
00755:C     READ AND PRINT NON-ZERO SPECIFIED FORCES AND DISPLACEMENTS
00756:C
00757:        DO 1 k=1,(NEQ+2*NSD+NR)
00758:1       F(K)=0.0
00759:        WRITE(10,2)LL
00760:2       FORMAT(1H,//,5X,'LOAD CASE NO.',2X,I5,//,
00761:       +5X,'NON-ZERO SPECIFIED DISPLACEMENTS OR FORCES',
00762:       +//,5X,'NODE NUMBER',2X,'COORDINATE NO.',5X,'CONDITION AND VALUE')
00763:3       READ(8,*) I,II,R
00764:        IF(I.EQ.0) GO TO 6
00765:        IK=NTBC(I,II)
00766:C
00767:        IF(IK.GT.0) THEN
00768:        F(IK)=R
00769:        WRITE(10,4)I,II,R
00770:4       FORMAT(//,8X,I5,8X,I5,10X,'FORCE',12X,F10.5)
00771:C
00772:        ELSE IF(IK.LT.-NSD) THEN
00773:        WRITE(10,*)'ERROR  NODE     ',I,'    ',II,'  IS ZERO DISPLACEMENT'
00774:C
00775:        ELSE
00776:        IK=IABS(IK)
00777:        F(IK+NEQ)=R
00778:        WRITE(10,5)I,II,R
00779:5       FORMAT(//,8X,I5,8X,I5,10X,'DISPLACEMENT',5X,F10.5)
00780:        END IF
00781:        GO TO 3
00782:C
00783:C     MODIFY RIGHT HAND SIDE OF EQUATION IF SPECIFIED NON-ZERO
00784:C     DISPLACEMENTS
00785:C
00786:6       IF(NSD.EQ.0)RETURN
00787:        DO 8 I=1,NEQ
00788:        JR=0
00789:        DO 7 J=1,NSD
```

```
00790:       JJ=IDIAG(2*J-1)
00791:       II=IDIAG(2*J)
00792:       JH=II-JR
00793:       JR=JJ-I
00794:       IF(JR.GT.0) GO TO 7
00795:       IF(JR.LT.0.AND.I.GT.(JJ+JH-1)) GO TO 7
00796:       IF(JR.EQ.0) THEN
00797:       F(I)=F(I)-F(NEQ+J)*B(JR+1)
00798:       ELSE
00799:       F(I)=F(I)-F(NEQ+J)*B(JR+1+I-JJ)
00800:       END IF
00801:7      JR=JR
00802:8      CONTINUE
00803:C
00804:       RETURN
00805:       END
00806:C
00807:C*******************************************************************************
00808:C
00809:C     SUBROUTINE STRESS
00810:C
00811:C*******************************************************************************
00812:C
00813:       SUBROUTINE STRESS
00814:C
00815:       COMMON / CPROFIL/ NMAT,NEL,NSD,NR,NNO,NDOF,NEQ,NEN(100)
00816:      +,NN(100,30),NTBC(200,2)
00817:       COMMON /CRED/ E(3),PR(3),G(3),MF(100),X(200,2)
00818:       COMMON /ABST/ A(7000),B(500),F(500)
00819:       COMMON /CBOUN/ SB(3364)
00820:       DIMENSION D(3),XE(58),UE(58),FE(58),STR(3),SIG(3)
00821:      +,SG(4),TG(4),SHF(4,3),S(8,8)
00822:C
00823:C
00824:C     LOCALIZE MATERIAL PROPERTIES,NODAL COORDINATES
00825:C     AND NODAL DISPLACEMENTS FOR EACH ELEMENT
00826:C
00827:1      FORMAT(//,'ELEMENT NO.',3X,'COORDINATES',20X,'STRESSES')
00828:C
00829:       REWIND(9)
00830:C
00831:       READ(8,*)R
00832:       READ(8,*)NE1,NE2
00833:       SG(1)=-R
00834:       SG(2)=R
00835:       SG(3)=R
00836:       SG(4)=-R
00837:       TG(1)=-R
00838:       TG(2)=-R
00839:       TG(3)=R
00840:       TG(4)=R
00841:C
00842:       DO 12 I=1,NEL
00843:C
00844:       NN12=NEN(I)*NEN(I)*4
00845:       IF(NEN(I).GT.4)READ(9,*)(SB(K),K=1,NN12)
00846:       IF(IK.EQ.1) THEN
00847:       READ(8,*)NE1,NE2
00848:       IK=0
00849:       ELSE
00850:       CONTINUE
00851:       END IF
00852:C
00853:       IF(NE2.EQ.I) IK=1
00854:       M=MF(I)
00855:       D(1)=E(M)
```

```
00855:        D(2)=PR(M)
00857:        D(3)=G(M)
00858:        NIN=NEN(I)
00859:C
00860:        DO 2 J=1,NIN
00861:        JJ=NN(I,J)
00862:        J2=2*(J-1)
00863:        XE(J)=X(JJ,1)
00864:        XE(J+NIN)=X(JJ,2)
00865:C
00866:        DO 2 K=1,NDOF
00867:        J2=J2+1
00868:        KK=NTBC(JJ,K)
00869:        IF(KK.LT.-NSD) THEN
00870:          UE(J2)=0.0
00871:          ELSE IF(KK.GT.0) THEN
00872:          UE(J2)=F(KK)
00873:          ELSE
00874:          UE(J2)=F(NEQ+IABS(KK))
00875:        END IF
00876:2       CONTINUE
00877:C
00878:C     CALCULATE REACTIONS
00879:C
00880:        M=0
00881:        DO 3 J=1,8
00882:        DO 3 K=1,8
00883:3       S(K,J)=0.0
00884:C
00885:        DO 6 J=1,NEN(I)
00886:        JJ=NN(I,J)
00887:        J2=2*(J-1)
00888:        DO 6 K=1,NDOF
00889:        J2=J2+1
00890:        KK=NTBC(JJ,K)
00891:        IF(KK.GT.0) GO TO 6
00892:        IF(M.EQ.0.AND.NIN.EQ.4)CALL ELEMENT(D,XE,S,NEN(I),NDOF,3)
00893:        M=1
00894:        MC=NEQ+NSD+IABS(KK)
00895:        IF(NIN.EQ.4) THEN
00896:        DO 4 L=1,8
00897:4       F(MC)=F(MC)+S(J2,L)*UE(L)
00898:C
00899:        ELSE
00900:        DO 5 L=1,2*NIN
00901:5       F(MC)=F(MC)+SB(2*NIN*(L-1)+J2)*UE(L)
00902:        END IF
00903:6       CONTINUE
00904:C
00905:C     CALCULATE  STRAINS AND STRESSES
00906:C
00907:        IF(NIN.GT.4) THEN
00908:        II=2*NIN
00909:C
00910:C     CALCULATE FE(I), THE VECTOR OF NODAL FORCES IN THE S. ELEMENTS
00911:C
00912:        DO 7 L=1,II
00913:        FE(L)=0.0
00914:        DO 7 K=1,II
00915:7       FE(L)=FE(L)+SB(II*(K-1)+L)*UE(K)
00916:C
00917:C     BOUN(...,2) CALCULATES STRESSES AND DISPLACEMENTS  IN
00918:C     THE S. ELEMENTS
00919:C
00920:        CALL BOUN(NIN,II,D,XE,SB,UE,FE,2)
00921:        IF(I.NE.NEL) WRITE(10,1)
```

```
00922:C
00923:       ELSE
00924:       IF(NE1.EQ.0) GO TO 12
00925:       IF(I.LT.NE1.OR.I.GT.NE2) GO TO 12
00926:       IF(I.EQ.1)WRITE(10,1)
00927:       LL=4
00928:       IF(NIN.EQ.3.OR.R.EQ.0.0) LL=1
00929:       DO 10 L=1,LL
00930:C
00931:C     CALCULATE SHAPE FUNCTIONS AND THEIR DERIVATIVES
00932:C
00933:       CALL SHAPE(SG(L),TG(L),XE,SHP,XSJ,NDOF,NIN)
00934:       DO 8 K=1,3
00935:8      STR(K)=0.0
00936:       XC=0.0
00937:       YC=0.0
00938:       DO 9 J=1,NEN(I)
00939:       J2=2*J
00940:       J1=J2-1
00941:       XC=XC+SHP(J,3)*XE(J)
00942:       YC=YC+SHP(J,3)*XE(J+NIN)
00943:       STR(1)=STR(1)+SHP(J,1)*UE(J1)
00944:       STR(2)=STR(2)+SHP(J,2)*UE(J2)
00945:9      STR(3)=STR(3)+SHP(J,1)*UE(J2)+SHP(J,2)*UE(J1)
00946:C
00947:       SIG(1)=D(1)*STR(1)+D(2)*STR(2)
00948:       SIG(2)=D(2)*STR(1)+D(1)*STR(2)
00949:       SIG(3)=D(3)*STR(3)
00950:C
00951:10     WRITE(10,11)I,XC,YC,SIG(1),SIG(2),SIG(3)
00952:11     FORMAT(/,I5,5(2X,F12.6))
00953:       END IF
00954:12      CONTINUE
00955:       CLOSE(UNIT=9)
00956:C
00957:C     OUTPUT OF REACTIONS
00958:C
00959:       WRITE(10,13)
00960:13     FORMAT(//,3X,'NODE NO.',3X,'DIRECTION',3X,'REACTION',/)
00961:       DO 15 I=1,NNO
00962:       DO 15 K=1,NDOF
00963:       KK=NTBC(I,K)
00964:       IF(KK.GT.0) GO TO 15
00965:       MC=NEQ+NSD+IABS(KK)
00966:       R=F(MC)
00967:       WRITE(10,14)I,K,R
00968:14     FORMAT(3X,I5,6X,I5,3X,F12.6,/)
00969:15     CONTINUE
00970:C
00971:       RETURN
00972:       END
00973:C
00974:C*******************************************************************************
00975:C
00976:C     SUBROUTINE BOUN(N,NN,D,XE,UC,SAV,F,IB)
00977:C
00978:C*******************************************************************************
00979:C
00980:       SUBROUTINE BOUN(N,NN,D,XE,UC,SAV,F,IB)
00981:C
00982:       DIMENSION UR(61,61)
00983:       DIMENSION R(6),W(6),W1(6),W2(6)
00984:       DIMENSION UC(NN,NN)
00985:       DIMENSION XE(N,2)
00986:      +,IPIV(61),D(3)
00987:      +,XF(1),YF(1),SAV(58),F(58)
```

```
00988:     +,D(21)
00989:C
00990:       PR=1.0-2.0*D(3)/D(1)
00991:       E=D(1)*(1.0-PR*PR)
00992:       P1=ACOS(-1.0)
00993:       PR2=2.*PR
00994:       PR3=3.*PR
00995:       P7=-(3.-PR)/2.
00996:       NNP1=NN+1
00997:       NNP2=NN+2
00998:       NNP3=NN+3
00999:C
01000:C  POINTS AND WEIGHTS FOR NUMERICAL INTEGRATION
01001:C
01002:       LL=6
01003:       WRITE(10,1)LL
01004:1      FORMAT(//,'POINTS OF INTEGRATION IN SUPERELEMENT',3X,I5)
01005:C
01006:       R(1)=-0.93246951420315
01007:       R(2)=-0.66120938646626
01008:       R(3)=-0.23861918608319
01009:       R(4)=-R(3)
01010:       R(5)=-R(2)
01011:       R(6)=-R(1)
01012:       W(1)=0.17132449237917
01013:       W(2)=0.36076157304813
01014:       W(3)=0.46791393457269
01015:       W(4)=W(3)
01016:       W(5)=W(2)
01017:       W(6)=W(1)
01018:       DO 2 L=1,LL
01019:       W1(L)=1.-R(L)
01020:2      W2(L)=1.+R(L)
01021:       IF(IB.EQ.2) GO TO 38
01022:C
01023:C  INITIALIZE MATRICES
01024:C
01025:       DO 3 I=1,NNP3
01026:       DO 3 J=1,NNP3
01027:       UR(J,I)=0.0
01028:       IF(I.GT.NN.OR.J.GT.NN) GO TO 3
01029:       UC(J,I)=0.0
01030:3       CONTINUE
01031:C
01032:       DO 4 I=1,N
01033:       UR(2*I-1,NNP1)=1.0
01034:       UR(2*I  ,NNP2)=1.0
01035:       UR(NNP1,2*I-1)=1.0
01036:4      UR(NNP2,2*I  )=1.0
01037:C
01038:       DO 5 I=2,N
01039:       UR(2*I-1,NNP3)=XE(1,2)-XE(I,2)
01040:       UR(2*I  ,NNP3)=XE(I,1)-XE(1,1)
01041:       UR(NNP3,2*I-1)=XE(1,2)-XE(I,2)
01042:5      UR(NNP3,2*I  )=XE(I,1)-XE(1,1)
01043:C
01044:C
01045:C  LOOP ON COLUMNS
01046:C
01047:       DO 7 J=1,N
01048:       JJ=2*J
01049:       IF(J.EQ.1) THEN
01050:         JM1=N
01051:         JJM2=NN
01052:       ELSE
01053:         JM1=J-1
```

```
01054:          JJM2=JJ-2
01055:        END IF
01056:        IF(J.EQ.N) THEN
01057:          JP1=1
01058:          JJP1=1
01059:        ELSE
01060:          JP1=J+1
01061:          JJP1=JJ+1
01062:        END IF
01063:C
01064:        UC(JJ-1,JJM2)=UC(JJ-1,JJM2)+2.-PR2
01065:        UC(JJ  ,JJM2-1)=UC(JJ  ,JJM2-1)-2.+PR2
01066:        UC(JJ-1,JJP1+1)=UC(JJ-1,JJP1+1)-2.+PR2
01067:        UC(JJ  ,JJP1)=UC(JJ  ,JJP1)+2.-PR2
01068:C
01069:        XM=XE(J,1)-XE(JM1,1)
01070:        YM=XE(J,2)-XE(JM1,2)
01071:        SF=SQRT(XM*XM+YM*YM)
01072:        CTF=YM/SF
01073:        STF=-XM/SF
01074:        XM=XE(JP1,1)-XE(J,1)
01075:        YM=XE(JP1,2)-XE(J,2)
01076:        SS=SQRT(XM*XM+YM*YM)
01077:        CTS=YM/SS
01078:        STS=-XM/SS
01079:        ARG=CTS*CTF+STS*STF
01080:        IF(ARG.GT.1.0) THEN
01081:        OM=PI*4.
01082:        ELSE
01083:         OM=(PI-ACOS(ARG))*4.
01084:        END IF
01085:C
01086:        AA=(2.+PR2)*(STS*CTS-STF*CTF)
01087:        BB=(2.+PR2)*(CTS*CTS-CTF*CTF)
01088:        CC=(2.-PR2)*LOG(SF/SS)
01089:C
01090:        UC(JJ-1,JJ-1)=OM+AA
01091:        UC(JJ,JJ-1)=-BB-CC
01092:        UC(JJ-1,JJ)=-BB+CC
01093:        UC(JJ,JJ)=OM-AA
01094:C
01095:        AA=1.-LOG(SF)
01096:        BB=STF*STF
01097:        CC=-CTF*STF
01098:        DD=CTF*CTF
01099:C
01100:        UR(JJ-1,JJ-1)=AA*(3.-PR)+BB*(1.+PR)
01101:        UR(JJ,JJ-1)=CC*(1.+PR)
01102:        UR(JJ-1,JJ)=UR(JJ,JJ-1)
01103:        UR(JJ,JJ)=AA*(3.-PR)+DD*(1.+PR)
01104:C
01105:        AA=1.-LOG(SS)
01106:        BB=STS*STS
01107:        CC=-CTS*STS
01108:        DD=CTS*CTS
01109:C
01110:        UR(JJ-1,JJP1)=AA*(3.-PR)+BB*(1.+PR)
01111:        UR(JJ,JJP1)=CC*(1.+PR)
01112:        UR(JJ-1,JJP1+1)=UR(JJ,JJP1)
01113:        UR(JJ,JJP1+1)=AA*(3.-PR)+DD*(1.+PR)
01114:C
01115:        XM=(XE(J,1)+XE(JM1,1))/2.
01116:        YM=(XE(J,2)+XE(JM1,2))/2.0
01117:        B1=XE(J,1)-XM
01118:        B2=XE(J,2)-YM
01119:        BB=B1*B1+B2*B2
```

```
01120:       P8=B2*(2.+PR2)
01121:       P9=B1*(2.+PR2)
01122:       P10=B2*(1.-PR)
01123:       P11=B1*(3.+PR)
01124:       P12=B1*(1.+PR3)
01125:       P13=B2*(3.+PR)
01126:       P14=B2*(1.+PR3)
01127:       P15=B1*(1.-PR)
01128:C
01129:C    LOOP ON ROWS
01130:C
01131:       DO 7 I=1,N
01132:       IF(I.EQ.J.OR.I.EQ.JM1) GO TO 7
01133:       II=2*I
01134:       A1=XE(I,1)-XM
01135:       A2=XE(I,2)-YM
01136:       AA=A1*A1+A2*A2
01137:       AB=2.*(A1*B1+A2*B2)
01138:C
01139:C    LOOP ON POINTS OF INTEGRATION
01140:C
01141:       DO 6 L=1,LL
01142:       RR=AA-AB*R(L)+BB*R(L)*R(L)
01143:       A1B1=A1-B1*R(L)
01144:       UF3=A1B1/RR
01145:       UF6=A1B1*UF3
01146:       UF1=UF6*UF3
01147:       A1B1=A2-B2*R(L)
01148:       UF7=A1B1*UF3
01149:       UF4=A1B1/RR
01150:       UF8=A1B1*UF4
01151:       UF2=UF4*UF8
01152:       UF5=LOG(RR)
01153:C
01154:       EE=(P8*UF1+P9*UF2+P10*UF3-P11*UF4)*W(L)
01155:       UC(II-1,JJM2-1)=UC(II-1,JJM2-1)+W1(L)*EE
01156:       UC(II-1,JJ-1)=UC(II-1,JJ-1)+W2(L)*EE
01157:C
01158:       EE=(P9*UF1-P8*UF2-P12*UF3+P13*UF4)*W(L)
01159:       UC(II-1,JJM2)=UC(II-1,JJM2)+W1(L)*EE
01160:       UC(II-1,JJ)=UC(II-1,JJ)+W2(L)*EE
01161:C
01162:       EE=(P9*UF1-P8*UF2-P11*UF3+P14*UF4)*W(L)
01163:       UC(II,JJM2-1)=UC(II,JJM2-1)+W1(L)*EE
01164:       UC(II,JJ-1)=UC(II,JJ-1)+W2(L)*EE
01165:C
01166:       EE=(-P8*UF1-P9*UF2+P13*UF3-P15*UF4)*W(L)
01167:       UC(II,JJM2)=UC(II,JJM2)+EE*W1(L)
01168:       UC(II,JJ)=UC(II,JJ)+W2(L)*EE
01169:C
01170:       UR(II-1,JJ-1)=UR(II-1,JJ-1)+(P7*UF5+(1.+PR)*UF6)*W(L)/2.0
01171:       UR(II-1,JJ)=UR(II-1,JJ)+(1.+PR)*UF7*W(L)/2.0
01172:       UR(II,JJ-1)=UR(II-1,JJ)
01173:       UR(II,JJ)=UR(II,JJ)+(P7*UF5+(1.+PR)*UF8)*W(L)/2.0
01174:6      CONTINUE
01175:7      CONTINUE
01176:C
01177:C     MODIFY UR
01178:C
01179:       DO 10 I=1,N
01180:       II=2*I
01181:       IIM1=II-1
01182:       A=1.0
01183:       DO 8 K=2,N
01184:       KK=2*K
01185:       KKM1=KK-1
```

```
01166:       UR(IIM1,1)=UR(IIM1,1)+UR(IIM1,KKM1)*A
01167:       UR(II  ,1)=UR(II  ,1)+UR(II  ,KKM1)*A
01168:       UR(IIM1,2)=UR(IIM1,2)+UR(IIM1,KK  )*A
01169:       UR(II  ,2)=UR(II  ,2)+UR(II  ,KK  )*A
01170:8      A=-A
01171:       DO 9 K=2,N
01172:       KK=2*K
01173:       UR(IIM1,KK-1)=-UR(IIM1,KK-3)+2.0*UR(IIM1,KK-1)
01174:       UR(II  ,KK-1)=-UR(II  ,KK-3)+2.0*UR(II  ,KK-1)
01175:       UR(IIM1,KK  )=-UR(IIM1,KK-2)+2.0*UR(IIM1,KK  )
01176:9      UR(II  ,KK  )=-UR(II  ,KK-2)+2.0*UR(II  ,KK  )
01177:10     CONTINUE
01178:C
01179:C    THE OUTER LOOP USES ELEMENTARY ROW OPERATIONS TO TRANSFORM
01180:C    UR TO TRIANGULAR FORM.
01181:C
01182:       DO 16 I=1,NNP2
01183:C
01184:C    SEARCH FOR LARGEST ENTRY IN COLUMN I, ROWS I THROUGH NNP3.
01185:C    IPIV(I) IS THE ROW INDEX OF THE LARGEST ENTRY.
01186:C
01187:       PIVOT=0.0
01188:       DO 11 J=I,NNP3
01189:       TEMP=ABS(UR(J,I))
01190:       IF(PIVOT.GE.TEMP) GO TO 11
01191:       PIVOT=TEMP
01192:       IPIV(I)=J
01193:11     CONTINUE
01194:       IF(PIVOT.EQ.0.0) WRITE(10,*)'  ERROR ',I,J
01195:       IF(IPIV(I).EQ.I) GO TO 13
01196:C
01197:C    INTERCHANGE ROW I AND ROW IPIV(I).
01198:C
01199:       DO 12 K=I,NNP3
01200:       TEMP=UR(I,K)
01201:       UR(I,K)=UR(IPIV(I),K)
01202:       UR(IPIV(I),K)=TEMP
01203:12     CONTINUE
01204:C
01205:C    ZERO ENTRIES (I+1,I),(I+2,I),....,(NNP3,I) IN MATRIX UR
01206:C
01207:13     IP1=I+1
01208:       DO 15 K=IP1,NNP3
01209:       UR(K,I)=-UR(K,I)/UR(I,I)
01210:       Q=UR(K,I)
01211:       DO 14 J=IP1,NNP3
01212:       UR(K,J)=UR(K,J)+UR(I,J)*Q
01213:14     CONTINUE
01214:15     CONTINUE
01215:16     CONTINUE
01216:       IF(UR(NNP3,NNP3).EQ.0.0) WRITE(10,*)'   ERROR  NO. 2'
01217:C
01218:C    MODIFY COLUMNS OF UC . BACKSOLVE.
01219:C
01220:       DO 21 K=1,NN
01221:C
01222:       DO 40 I=1,NN
01223:40     B(I)=UC(I,K)
01224:       B(NNP1)=0.0
01225:       B(NNP2)=0.0
01226:       B(NNP3)=0.0
01227:C
01228:       DO 18 I=1,NNP2
01229:       IF(IPIV(I).EQ.I) GO TO 17
01230:       TEM=B(IPIV(I))
01231:       B(IPIV(I))=B(I)
```

```
01252:       B(I)=TEM
01253:17     DO 18 L=I+1,NNP3
01254:18     B(L)=B(L)+B(I)*UR(L,I)
01255:C
01256:       B(NNP3)=B(NNP3)/UR(NNP3,NNP3)
01257:       DO 20 I=1,NNP2
01258:       Q=0.0
01259:       DO 19 L=1,I
01260:       Q=Q+UR(NNP3-I,NNP3-L+1)*B(NNP3-L+1)
01261:19     CONTINUE
01262:       B(NNP3-I)=(B(NNP3-I)-Q)/UR(NNP3-I,NNP3-I)
01263:20     CONTINUE
01264:C
01265:       DO 41 I=1,NN
01266:41     UC(I,K)=B(I)
01267:21     CONTINUE
01268:C
01269:C     SYMMETRY IN UC
01270:C
01271:       DO 22 I=1,NN
01272:       DO 22 K=I,NN
01273:       UC(I,K)=D(3)*(UC(I,K)+UC(K,I))/2.
01274:22     UC(K,I)=UC(I,K)
01275:       IF(IB.EQ.1) RETURN
01276:C
01277:C     CALCULATIONS FOR FIELD POINTS
01278:C
01279:38     WRITE(10,39)
01280:39     FORMAT(/,5X,'RESULTS FOR FIELD POINTS',//,6X,'COORDINATES',
01281:      +15X,'DISPLACEMENTS',13X,'STRESSES',//)
01282:30     READ(8,*)IA,XF(1),YF(1)
01283:       IF(IA.NE.1) RETURN
01284:       DO 31 I=1,NN
01285:       DO 31 K=1,5
01286:       UC(K,I)=0.0
01287:31     UR(K,I)=0.0
01288:       DO 32 J=1,N
01289:       JJ=2*J
01290:       IF(J.EQ.1) THEN
01291:         JJM2=NN
01292:         JJM3=NN-1
01293:         JM1=N
01294:       ELSE
01295:         JJM2=JJ-2
01296:         JJM3=JJ-3
01297:         JM1=J-1
01298:       END IF
01299:C
01300:       XM=(XE(J,1)+XE(JM1,1))/2.0
01301:       YM=(XE(J,2)+XE(JM1,2))/2.0
01302:       B1=XE(J,1)-XM
01303:       B2=XE(J,2)-YM
01304:       BB=B1*B1+B2*B2
01305:       P8=B2*(2.+PR2)
01306:       P9=B1*(2.+PR2)
01307:       P10=B2*(1.-PR)
01308:       P11=B1*(3.+PR)
01309:       P12=B1*(1.+PR3)
01310:       P13=B2*(3.+PR)
01311:       P14=B2*(1.+PR3)
01312:       P15=B1*(1.-PR)
01313:C
01314:       A1=XF(1)-XM
01315:       A2=YF(1)-YM
01316:       AA=A1*A1+A2*A2
01317:       AB=2.*(A1*B1+A2*B2)
```

```
01318:C
01319:C   LOOP ON POINTS OF INTEGRATION
01320:C
01321:       DO 32 L=1,LL
01322:       RR=AA-AB*R(L)+BB*R(L)*R(L)
01323:       A1B1=A1-B1*R(L)
01324:       UF3 =A1B1/RR
01325:       UF6 =A1B1*UF3
01326:       UF1 =UF6*UF3
01327:       A1B1=A2-B2*R(L)
01328:       UF7 =A1B1*UF3
01329:       UF4 =A1B1/RR
01330:       UF8 =A1B1*UF4
01331:       UF2 =UF4*UF8
01332:       UF5 =LOG(RR)
01333:       UF9=UF3*UF3
01334:       UF10=UF4*UF4
01335:       UF11=1.0/RR
01336:       UF12=UF6*UF8*UF11
01337:       UF13=UF6*UF11
01338:       UF14=UF8*UF11
01339:C
01340:       EE=(P8*UF1+P9*UF2+P10*UF3-P11*UF4)*W(L)
01341:       UC(1,JJM2-1)=UC(1,JJM2-1)+W1(L)*EE
01342:       UC(1,JJ-1)=UC(1,JJ-1)+W2(L)*EE
01343:C
01344:       EE=(P9*UF1-P8*UF2-P12*UF3+P13*UF4)*W(L)
01345:       UC(1,JJM2)=UC(1,JJM2)+W1(L)*EE
01346:       UC(1,JJ)=UC(1,JJ)+W2(L)*EE
01347:C
01348:       EE=(P9*UF1-P8*UF2-P11*UF3+P14*UF4)*W(L)
01349:       UC(2,JJM2-1)=UC(2,JJM2-1)+W1(L)*EE
01350:       UC(2,JJ-1)=UC(2,JJ-1)+W2(L)*EE
01351:C
01352:       EE=(-P8*UF1-P9*UF2+P13*UF3-P15*UF4)*W(L)
01353:       UC(2,JJM2)=UC(2,JJM2)+EE*W1(L)
01354:       UC(2,JJ)=UC(2,JJ)+W2(L)*EE
01355:C
01356:       UR(1,JJ-1)=UR(1,JJ-1)+(P7*UF5+(1.+PR)*UF6)*W(L)/2.0
01357:       UR(1,JJ)=UR(1,JJ)+(1.+PR)*UF7*W(L)/2.0
01358:       UR(2,JJ-1)=UR(1,JJ)
01359:       UR(2,JJ)=UR(2,JJ)+(P7*UF5+(1.+PR)*UF8)*W(L)/2.0
01360:C
01361:       EE=(2.0+PR2)*UF1
01362:       UR(3,JJ-1)=UR(3,JJ-1)-(EE+(1.0-PR)*UF3)*W(L)
01363:       UR(4,JJ-1)=UR(4,JJ-1)+(EE-(1.0+3.0*PR)*UF3)*W(L)
01364:       UR(5,JJ  )=UR(5,JJ  )+(EE-(3.0+PR)*UF3)*W(L)
01365:C
01366:       EE=(2.0+PR2)*UF2
01367:       UR(3,JJ  )=UR(3,JJ  )+(EE-(1.0+3.0*PR)*UF4)*W(L)
01368:       UR(4,JJ  )=UR(4,JJ  )-(EE+(1.0-PR)*UF4)*W(L)
01369:       UR(5,JJ-1)=UR(5,JJ-1)+(EE-(3.0+PR)*UF4)*W(L)
01370:C
01371:C
01372:       EE=(2.*B2*(UF11+4.*UF13-8.*UF1*UF3)-4.*B1*UF7*(UF11-4.*UF9))
01373:      +*W(L)
01374:       UC(3,JJM3)=UC(3,JJM3)+EE*W1(L)
01375:       UC(3,JJ-1)=UC(3,JJ-1)+EE*W2(L)
01376:C
01377:       EE=(4.*B2*UF7*(UF11-4.0*UF9)-2.*B1*(UF11-8.0*UF12))*W(L)
01378:       UC(5,JJM3)=UC(5,JJM3)+EE*W1(L)
01379:       UC(5,JJ-1)=UC(5,JJ-1)+EE*W2(L)
01380:       UC(3,JJM2)=UC(3,JJM2)+EE*W1(L)
01381:       UC(3,JJ  )=UC(3,JJ  )+EE*W2(L)
01382:C
01383:       EE=((UF11-8.0*UF12)*2.*B2-UF7*(UF11-4.0*UF10)*4.*B1)*W(L)
```

```
01384:       UC(4,JJM3)=UC(4,JJM3)+EE*W1(L)
01385:       UC(4,JJ-1)=UC(4,JJ-1)+EE*W2(L)
01386:       UC(5,JJM2)=UC(5,JJM2)+EE*W1(L)
01387:       UC(5,JJ  )=UC(5,JJ  )+EE*W2(L)
01388:C
01389:C
01390:       EE=(4.*B2*UF7*(UF11-4.*UF10)-2.*B1*(UF11+4.*UF14-8.*UF2*UF4))
01391:      +*W(L)
01392:       UC(4,JJM2)=UC(4,JJM2)+EE*W1(L)
01393:       UC(4,JJ  )=UC(4,JJ  )+EE*W2(L)
01394:C
01395:32      CONTINUE
01396:C
01397:C     MODIFY UR
01398:C
01399:       A=1.0
01400:       DO 34 K=2,N
01401:       KK=2*K
01402:       KKM1=KK-1
01403:       DO 33 L=1,5
01404:       UR(L,1)=UR(L,1)+UR(L,KKM1)*A
01405:33     UR(L,2)=UR(L,2)+UR(L,KK  )*A
01406:34     A=-A
01407:       DO 35 K=2,N
01408:       KK=2*K
01409:       KKM1=KK-1
01410:       DO 35 L=1,5
01411:       UR(L,KKM1)=-UR(L,KK-3)+2.0*UR(L,KKM1)
01412:35     UR(L,KK  )=-UR(L,KK-2)+2.0*UR(L,KK  )
01413:C
01414:       DIS1=0.0
01415:       DIS2=0.0
01416:       SIG1=0.0
01417:       SIG2=0.0
01418:       SIG3=0.0
01419:C
01420:       DO 36 I=1,NN
01421:       DIS1=DIS1-UC(1,I)*SAV(I)+UR(1,I)*F(I)
01422:       DIS2=DIS2-UC(2,I)*SAV(I)+UR(2,I)*F(I)
01423:       SIG1=SIG1-(1.0+PR)*UC(3,I)*SAV(I)+UR(3,I)*F(I)
01424:       SIG2=SIG2-(1.0+PR)*UC(4,I)*SAV(I)+UR(4,I)*F(I)
01425:36     SIG3=SIG3-(1.0+PR)*UC(5,I)*SAV(I)+UR(5,I)*F(I)
01426:       DIS1=DIS1/(8.0*PI)
01427:       DIS2=DIS2/(8.0*PI)
01428:       SIG1=SIG1/(8.0*PI)
01429:       SIG2=SIG2/(8.0*PI)
01430:       SIG3=SIG3/(8.0*PI)
01431:       WRITE(10,37)XF(1),YF(1),DIS1,DIS2,SIG1,SIG2,SIG3
01432:37     FORMAT(7(3X,F10.5),/)
01433:C
01434:       IF(IA.EQ.1) GO TO 30
01435:       RETURN
01436:       END
01437:C
BOTTOM
$
```

BIBLIOGRAPHY

1. Reddy, J.M., An Introduction to the Finite Element Method, McGraw Hill Book Co., New York, (1984).

2. Huebner, K.H., Thornton, E.A., The Finite Element Method for Engineers, John Wiley, Sons, New York, (1982).

3. Zienkiewics, O.C., The Finite Element Method, 3rd edition, McGraw Hill Book Co., New York, (1977).

4. Banerjee, P.K., Boundary Element Methods in Engineering Science, McGraw Hill Book Co., London, (1981).

5. Brebbia, C.A. and Walker, S., Boundary Element Techniques in Engineering, Newnes-Butterworths, London, (1980).

6. Beer, G., Finite Element, Boundary Element and Coupled Analysis of Unbounded Problems in Elastostatics, International Journal for Numerical Methods in Engineering, 19, (1983).

7. Mustoe, G.G.W., Volait, F. and Zienkiewics, O.C., A Symmetric Direct Boundary Integral Equation Method for Two Dimensional Elastostatics, Res Mechanica, 4, (1982).

8. Zienkiewics, O.C., Kelly, D.W. and Bettess, P., The Coupling of the Finite Element Method and Boundary Solution Procedures, International Journal for Numerical Methods in Engineering, 11, (1977).

9. Beer, G., BEFE - a combined boundary element finite element computer program, Adv. Eng. Software, John Wiley, Sons, New York, (1982).

10. Lapidus, L. and Pinder, G., Numerical Solution of Partial Differential Equations in Science and Engineering, John Wiley, Sons, New York, (1982).

11. Gupta, A.K., Efficient Numerical Integration of element stiffness matrices, International Journal for Numerical Methods in Engineering, 19, (1983).

12. Johnson, L.W. and Riess, R.D., Numerical Analysis, Addison-Wesley Publishing Company, Reading, Massachusetts, (1982).

13. Timoshenko, S.P. and Goodier, J.N., Theory of Elasticity, McGraw Hill Book Co., New York, (1953).

14. Savin, G.N., Stress Concentration Around Holes, Pergamon Press, New York, (1961).

15. Peterson, R.E., Stress Concentration Design Factors, John Wiley, Sons, New York, (1953).

16. Crotty, J.M., A Block Equation Solver for large Unsymmetric Matrices arising in the Boundary Integral Equation Method, International Journal for Numerical Methods in Engineering, 18, (1982).

17. Moscardini, A.O., Lewis, B.A. and Cross, M., AGTHOM-Automatic Generation of Triangular and Higher Order Meshes, International Journal for Numerical Methods in Engineering, 19, (1983).

18. Nguyen-Van-Phai, Automatic Mesh Generator with the Thetrahedrum Elements, International Journal for Numerical Methods in Engineering, 18, (1982).

19. Perucchio, R., Ingraffea, A.R. and Abel, J.F., Interactive Computer Graphic Preprocessing for three-dimentional Finite Element Analysis, International Journal for Numerical Methods in Engineering, 18, (1982).