THIRTY-TWO NODES HEXAHEDRONAL ELEMENT SUBROUTINE FOR MULTI-PURPOSE PROGRAM MEF

by

Anan Sukaneeyouth

September 1986

Thesis Advisor: Gilles Cantin

Approved for public release; distribution is unlimited.
**Title:** Thirty-Two Nodes Hexahedral Element Subroutine for Multi-Purpose Program MEF

**Personal Authors:** Anan Sukaneeyouth, LT, Thailand Navy

**Type of Report:** Engineer's Thesis

**Date of Report:** 1986 September

**Abstract:**

A general finite element program of moderate complexity called MEF is organized to contain a library of one, two, and three-dimensional elements for the solution of problems from a wide variety of disciplines. A cubic, thirty-two node, three dimensional isoparametric element was developed. With such an element very complex structures could be solved with a very coarse mesh.
Approved for public release; distribution is unlimited.

Thirty-Two Nodes Hexahedronal Element Subroutine for Multi-Purpose Program MEF

by

Anan Sukaneeyouth
Lieutenant, Royal Thai Navy
B.S., Royal Thai Naval Academy, 1978

Submitted in partial fulfillment of the requirements for the degrees of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING
and
MECHANICAL ENGINEER

from the

NAVAL POSTGRADUATE SCHOOL
September 1986

Author: Anan Sukaneeyouth

Approved by: Gilles Cantin, Thesis Advisor

A.J. Healey, Chairman,
Department of Mechanical Engineering

John N. Dyer,
Dean of Science and Engineering
ABSTRACT

A general finite element program of moderate complexity called MEF is organized to contain a library of one, two, and three-dimensional elements for the solution of problems from a wide variety of disciplines. A cubic, thirty-two node, three dimensional isoparametric element was developed. With such an element very complex structures could be solved with a very coarse mesh.
TABLE OF CONTENTS

I. INTRODUCTION ................................................................. 8
   A. A GENERAL-PURPOSE FINITE ELEMENT PROGRAM ................. 8
      1. Problem Definition (data base): ............................... 8
      2. Element Computations: ........................................... 8
      3. Assembly Operations: ............................................ 8
      4. Solution: ........................................................... 8
      5. Result: ............................................................. 8
   B. MEF PROGRAM .................................................................. 9

II. DESCRIPTION OF THE COMPUTER PROGRAM ......................... 11
   A. REFERENCE ELEMENT ..................................................... 11
   B. SUBROUTINE ELEM03 ...................................................... 12
      1. Operation common to all element of the same type: ........ 12
      2. Operation for the computation of matrix [k] of each
         element: ............................................................. 12
      3. Operation required to compute mass matrix [m]: ............ 12
      4. Operation required to compute consistent load vectors
         (f): ..................................................................... 12
      5. Operation required to compute the residue vector {r}: ....... 13
      6. Operation required to compute gradients \( \delta u \) at points
         of integral: .......................................................... 13
   C. CODING. ........................................................................... 13
      1. Evaluate coordinates, weights, functions \( N \) and their
         derivatives ............................................................. 13
      2. Computation of stiffness matrix \([k]\) of each element. .... 14
      3. Computation The Mass Matrix \([m]\) ................................. 18
      4. Computation of consistent load vector \((f)\). .................. 19
      5. Computation the residue array. .................................... 20
      6. Computation of the gradients and stresses at the points
         of integration. ...................................................... 20
III. THE SOLUTION OF A SIMPLE PROBLEM AND DISCUSSION OF RESULT ........................................... 24
   A. ENTRY AND EXECUTION FUNCTIONAL BLOCKS ........ 24
   B. STRUCTURE MODELING .................................... 24
   C. DISCUSSION OF RESULTS .................................. 26

APPENDIX: ELEM03 SUBPROGRAM LISTING .............................. 29

LIST OF REFERENCES .................................................. 37

INITIAL DISTRIBUTION LIST ........................................ 38
LIST OF TABLES

I. THE DISPLACEMENT FOR THE ONE ELEMENT ............... 28
II. THE DISPLACEMENT FOR THE EIGHT ELEMENTS ............. 28
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Reference Element</td>
<td>11</td>
</tr>
<tr>
<td>2.2</td>
<td>Block Diagram for the Shape Function and their Derivative</td>
<td>14</td>
</tr>
<tr>
<td>2.3</td>
<td>Block Diagram of Computation of Stiffness Matrix</td>
<td>15</td>
</tr>
<tr>
<td>2.4</td>
<td>The Jacobian Matrix</td>
<td>16</td>
</tr>
<tr>
<td>2.5</td>
<td>The Inverse of Jacobian Matrix</td>
<td>17</td>
</tr>
<tr>
<td>2.6</td>
<td>The derivative of shape function w.r.t. x, y, z</td>
<td>17</td>
</tr>
<tr>
<td>2.7</td>
<td>The Array VBE</td>
<td>17</td>
</tr>
<tr>
<td>2.8</td>
<td>The Array VDE</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>Block Diagram for the Mass Matrix [m]</td>
<td>18</td>
</tr>
<tr>
<td>2.10</td>
<td>The Shape Function Matrix [N]</td>
<td>19</td>
</tr>
<tr>
<td>2.11</td>
<td>The Product of Matrices [N]^T and [N]</td>
<td>20</td>
</tr>
<tr>
<td>2.12</td>
<td>Block Diagram for Consistent Load Vector [f]</td>
<td>21</td>
</tr>
<tr>
<td>2.13</td>
<td>Block Diagram for the Residue Array</td>
<td>21</td>
</tr>
<tr>
<td>2.14</td>
<td>The Strain-Node Value Relation</td>
<td>22</td>
</tr>
<tr>
<td>2.15</td>
<td>The Stress-Strain Relation</td>
<td>22</td>
</tr>
<tr>
<td>2.16</td>
<td>Evaluation of the Coordinate of the Integration Points on the Real Element</td>
<td>23</td>
</tr>
<tr>
<td>3.1</td>
<td>The Model used for the Problem</td>
<td>25</td>
</tr>
<tr>
<td>3.2</td>
<td>The consistent load at the end of the beam</td>
<td>26</td>
</tr>
</tbody>
</table>
I. INTRODUCTION

A. A GENERAL-PURPOSE FINITE ELEMENT PROGRAM

A general-purpose finite element program should be able to solve a variety of problems from the number of disciplines: linear and non-linear, static and dynamic problem of elasticity, fluid mechanics, heat transfer, etc. and can solve problems of large size involving a variety of elements.

A general program is going to be voluminous and complex. It is, however, desirable that:

- its logic be easily understood;
- one or many of its parts be easily modifiable;
- it offers possibilities to tailor its facilities for the solution particular classes of problems.

The program should have a modular structure, with the modules made as independent from one another as is practicable. The following modular operations are recognized:

1. Problem Definition (data base):
   - node coordinates and element connectivities;
   - nodal element properties;
   - boundary conditions.

2. Element Computations:
   - integration points and associated weights;
   - interpolation functions and their derivatives;
   - Jacobian matrix, inverse, and determinant;
   - element matrices and vectors: \([k], [m], \{f\}, \text{etc.}\)

3. Assembly Operations:
   - assemblage of master matrices and vectors, \([K], [M], \text{etc.}\)

4. Solution:
   - factorization of master matrices and solution of equations.

5. Result:
   - output of nodal variables and other calculated quantities: gradients, reactions, etc.
Subroutines implementing the various operations described above are contained in all finite element codes. The flow of information between these operations is problem dependent; linear, non-linear, static, and dynamic problems all require logic of their own.

B. MEF PROGRAM

A program of medium complexity, called MEF, implementing the techniques of a general-purpose program that can solve a large variety of boundary value problems of mathematical physics. It is written in FORTRAN IV and can be easily adaptable to various computers.

The main program controls the flow of all information through the functional blocks by transferring control to a subroutine called BLNNNNN when the block calling card NNNN is encountered in the input file. The subroutine BLNNNNN then performs preliminary functions such as logical unit identification, and reading of control parameters for the creation of various files and tables. The subroutine then calls subroutine EXNNNNN. In all cases, subroutine BLNNNNN provides appropriate default parameters which will be overridden by user values if specified. Subroutine EXNNNNN then performs the major operations of the block by calling on the needed subroutines in the MEF library. The above protocol holds for all blocks except STOP, COMT, and IMAG. All the functions of COMT and IMAG are performed by subroutine BLNNNNN, and the function of block STOP is performed by the main program.

The executable functional blocks contained in the MEF are:

<table>
<thead>
<tr>
<th>BLOCK</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOLR</td>
<td>Assemblage of distributed load.</td>
</tr>
<tr>
<td>LINM</td>
<td>Solution of linear problem with global matrix in core.</td>
</tr>
<tr>
<td>LIND</td>
<td>Solution of linear problem with global matrix out of core.</td>
</tr>
<tr>
<td>NLIN</td>
<td>Solution of stationary non-linear problem.</td>
</tr>
<tr>
<td>TEMP</td>
<td>Solution of unsteady problem (linear or non-linear).</td>
</tr>
<tr>
<td>VALP</td>
<td>Eigenvalues and eigenvectors.</td>
</tr>
</tbody>
</table>

The various blocks designed for execution of the various computations have similar structures since they have to:

- construct element and load matrices;
- assemble global matrices and vectors;
- factorize and solve the system of equations;
• output the results.

Using the constructed elements and load matrices, the subroutine element library ELEMLB is called. This library contains subroutines that define the individual element types. The ELEMO3 subroutine, a thirty-two node, three dimensional isoparametric element was developed and added to the element library. This new element allowed the solution of linear elastic structures composed of homogeneous and isotropic materials.

Multiple sample problems were developed to fully exercise use of this new element. An indepth investigation was then conducted to determine the limit of computational ability using the newly defined element to represent physical phenomenons.
II. DESCRIPTION OF THE COMPUTER PROGRAM

A. REFERENCE ELEMENT

To simplify the analytical expression for elements of complex shapes an element of reference is introduced. Such an element is defined in an abstract non-dimensional space with a very simple geometrical shape. The geometry of the reference element is then mapped into the geometry of the real element using geometrical transformation expression.

A thirty-two node, three dimensional cubic element was introduced as the reference element. The element has eight corner nodes and twenty four mid-side nodes dividing each edge in three equal parts as shown in Figure 2.1. Using this reference element we created the fundamental matrices and vectors in subroutine called ELEM03, NI03, D03 and B03 to be used in element library subroutine ELEMLB of the MEF program.

![Reference Element Diagram](image)

Figure 2.1 Reference Element.
B. SUBROUTINE ELEM03

Multi-purpose program MEF is organized to contain a library of one, two, and three-dimensional elements for the solution of problems from a wide variety of disciplines. Problems from the mechanics of solids and fluids, heat transfer, etc. have been solved. For each element type \( mn \), subroutine ELEM\( mn \) controls the computation of all matrices and vectors.

In element type 03, subroutine ELEM03 is organized to create the fundamental matrices and vectors that must be numerically integrated using methods of Numerical Integration described in and the computations can be carried out by the following steps.

1. Operation common to all element of the same type:
   - compute the weight \( w_r \) and the coordinate of integration points;
   - construct the functions \( N \) (interpolation functions), the function \( \bar{N} \) (geometrical interpolation functions) and their derivatives with respect to \( \xi, \eta, \zeta \) at the points of integration.

2. Operation for the computation of matrix \([k]\) of each element:
   - initialize the matrix \([k]\);
   - for each point of integration \( \xi_r \):
     - construct the Jacobian matrix \([J]\) from the derivatives with respect to \( \xi, \eta, \zeta \) of function \( N \) and the nodal coordinates of the element;
     - construct the inverse of \([J]\) and its determinant;
     - construct the derivatives of functions \( N \) with respect to \( x, y, z \) starting from the derivatives with respect of \( \xi, \eta, \zeta \);
     - construct the matrices \([B]\) and \([D]\);
     - accumulate into \([k]\) the values of \([B][D][B]\det(J)w_r\) calculated for each integration point.

3. Operation required to compute mass matrix \([m]\):
   - initialize \([m]\)
   - for each integration point \( \xi_r \):
     - compute Jacobian matrix and its determinant;
     - accumulate the values of \((N) < N > \det(J)w_r\) into matrix \([m]\).

4. Operation required to compute consistent load vectors \([f]\):
   - initialize \([f]\);
   - for each integration point \( \xi_r \):
     - compute the Jacobian matrix and its determinant;
• accumulate the values of \( (Nf) \text{det}(J)w_r \) into \( \{f\} \).

5. Operation required to compute the residue vector \( \{r\} \):
   - using the value of \( \{f\} \) from 4;
   - for each integration point \( \xi_r \):
     • compute matrices \([B], [D], [J]\) as in 2 above;
     • accumulate the product: \( \{f\} - [B][D][B]\{u_n\} w_r \text{det}(J) \) into \( \{r\} \).

6. Operation required to compute gradients \( \partial u \) at points of integral:
   - for each integration point \( \xi_r \):
     • construct matrix \([B]\) as in 2 above;
     • compute and print gradient \( \{\partial u\} = [B]\{u_n\} \).

The subroutine ELEM03 executes one operation at a time depending on the value of ICODE. Control variable ICODE specifies which element operation is desired, and expression of this variable as follows:

ICODE = 1 initialization of the characteristic parameters of an element
(number of nodes, number of degrees of liberty).

ICODE = 2 operations required by a given reference element which are
independent of the real geometry; construction of interpolation function \( N \) and their derivative with
respect to \( \xi \) at the points of integration.

ICODE = 3 construction of matrix \([k]\) in array VKE.

ICODE = 4 construction of tangent matrix needed for non-linear
problems in array VKE.

ICODE = 5 construction of mass matrix \([m]\) in array VKE.

ICODE = 6 computation of residual vector \( \{r\} \) in array VFE.

ICODE = 7 computation load vector \( \{f\} \) in array VFE.

ICODE = 8 computation and printing of gradients \( \{\partial u\} \).

C. CODING.

1. Evaluate coordinates, weights, functions \( N \) and their derivatives
Integration formula for numerical integration is the following form.

\[
\int_{\xi_1}^{\xi_g} = \sum_{i=1}^{ng} w_i \xi(\xi_1) \quad \text{(eqn 2.1)}
\]

where
- \( \xi_1 \) are the coordinates of integration point 1 in \( \xi, \eta, \zeta \) system
- \( w_1 \) coordinate corresponding to weight \( w_1 \).
- \( w_i \) are the weights corresponding to integration point number;
- IPG are the total number of integration points.

A choice of 2, 3 or 4 integration points by dimension can be made in subroutine GAUSS [Ref. 1:p. 265], giving respectively 8, 27 and 64 integration points, the weights corresponding to the integration points and their coordinates. They are in the array called IPG, VCPG and VKPG respectively.

Subroutine N103 create the array VNI that contains the shape function \( N_i \) and their derivatives with respect to \( \xi_i \) ( \( \xi, \eta, \zeta \) system coordinate ) as Figure 2.2.

![Figure 2.2 Block Diagram for the Shape Function and their Derivative.](image)

2. Computation of stiffness matrix \([k]\) of each element.

The explicit equation of element stiffness matrix is following:

\[
[k] = \int \int \int [B]^T[D][B]\det(J)d\xi d\eta d\zeta \\
\text{(eqn 2.2)}
\]

where

- \([B]\) is the linear strain matrix;
- \([B]^T\) is the transpose matrix of \([B]\);
- \([D]\) is the matrix of elastic constants for an isotropic material.

The stiffness matrix that has the block diagram as Figure 2.3 is the integration of product of the linear strain transpose matrix, the element property matrix and the linear strain matrix over the volume of the reference element.
For each integration point $\xi_r$

Subroutine JACOB

$\left[ J \right], \left[ J^T \right], \text{det}(J)$

Subroutine DNXID

$\frac{\partial N_i}{\partial x}, \frac{\partial N_i}{\partial y}, \frac{\partial N_i}{\partial z}$

Subroutine D03

$[B]$,

Subroutine D03

$[D]$,

Subroutine BTDB

$[k] = [B]^T[D][B]\text{det}(J)$

---

Figure 2.3  Block Diagram of Computation of Stiffness Matrix.

In coding, we employ numerical integration formular having the following generic form:

$$[k] = \sum_{i=1}^{\text{IPG}} w_i[k^0(\xi_i)]$$  \hspace{1cm} (eqn 2.3)

where

- IPG are the total number of integration points;
- $w_i$ are the weighting coefficients corresponding to each integration point;
- $\xi_i$ are the coordinate of the IPG integration points;
- $[k^0]$ is the stiffness matrix at each integration point as shown in equation 2.4.
\[ [k^0] = [B]^t[D][B]\text{det}(J) \] (eqn 2.4)

Subroutine JACOB [Ref. 1:p. 63], compute the Jacobian matrix, its inverse and its determinant. The Jacobian matrix as Figure 2.4 is obtained as the product of two matrices, one containing the derivatives of the geometrical transformation functions with respect to the space of the reference element, and the other containing the real coordinates of the geometrical nodes of the element.

\[ \langle x\ y\ z \rangle = \langle N(\xi)\rangle [(x_n)\ (y_n)\ (z_n)] \] (eqn 2.5)

\( \{x_n\}, \{y_n\}, \{z_n\} \) being the geometrical node coordinates. The Jacobian matrix is figure 2.4.

\[
[J] = \begin{vmatrix}
\frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi} \\
\frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \\
\frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} & \frac{\partial z}{\partial \zeta}
\end{vmatrix}
\]

Figure 2.4 The Jacobian Matrix.

The inverse of Jacobian matrix \([J]^{-1}\) as Figure 2.5 and its determinant are computed by the method described on [Ref. 1:p. 44],

Subroutine DNIDX [Ref. 1:p. 64], computes the derivatives of the shape function \(N_i\) with respect to the coordinate system of the real element using the product on Figure 2.6.

Subroutine B03 creates the matrix as Figure 2.7 that contains the strain components at all direction of each node in the element using output components of subroutine DNIDX and rearrange the new array called VBE.

Subroutine D03 compute the stress-strain matrix (VDE) as Figure 2.8 [Ref. 2:p. 110] for isotropic materials ( \(E = \text{Young's Modulus}, \nu = \text{Pisson's Ratio}\) ).
Subroutine BTDB construct the stiffness matrix by adding the product of the transpose matrix VBE, the stress-strain matrix VDE and the matrix VBE of every integration points.

\[ [k] = [B]^T[D][B]w_i \cdot \text{det}(J) \]  
\[(\text{eqn 2.6})\]

3. Computation The Mass Matrix \([m]\).

The element mass matrix has the block diagram as Figure 2.9 and the explicit equation as following:

\[ [m] = \int \int \int [N]^T[N] \cdot \text{det}(J) \, d\xi \, d\eta \, d\zeta \]  
\[(\text{eqn 2.7})\]

and numerical integration form is:
\[ [\mathbf{s}] = \begin{bmatrix}
\delta_{1,x} & 0 & 0 & \cdots & 0 \\
0 & \delta_{1,y} & 0 & \cdots & 0 \\
0 & 0 & \delta_{2,x} & \cdots & 0 \\
\cdots & \cdots & \cdots & \cdots & \cdots \\
0 & 0 & 0 & \cdots & \delta_{32,x} \\
0 & 0 & 0 & \cdots & \delta_{32,y} \\
\end{bmatrix} \]

Figure 2.7 The Array VBE.

\[
[D] = \frac{E(1-\nu)}{(1-\nu)(1-2\nu)} \begin{bmatrix}
1 & \frac{\nu}{1-\nu} & \frac{\nu}{1-\nu} & 0 & 0 & 0 \\
\frac{\nu}{1-\nu} & 1 & \frac{\nu}{1-\nu} & 0 & 0 & 0 \\
\frac{\nu}{1-\nu} & \frac{\nu}{1-\nu} & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1-2\nu}{2(1-\nu)} & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1-2\nu}{2(1-\nu)} & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{1-2\nu}{2(1-\nu)} \\
\end{bmatrix}
\]

Figure 2.8 The Array VDE.

\[ [m] = \sum_{i=1}^{\text{nodes}} w_i [N]^t[N] \text{det}(J) \quad (\text{eqn 2.8}) \]

- $[N]$ is the shape function matrix Figure 2.10 that was created by subroutine N103;
- $[N]^t$ is the transpose matrix of $[N]$;
- $w_i$ are the weighting coefficients corresponding to each integration point;

Obviously look at the product of matrices $[N]^t$and$[N]$ as Figure 2.11 is a symmetric matrix. By convention, the mass matrix $[m]$ contains upper half components and diagonal components of this matrix.
For each integration point $\xi_r$

Subroutine JACOB
- $[J]$, $[\dot{J}]$, $\det(J)$

Accumulate the values

$[m] = w_i[N][N]\det(J)$

Figure 2.9 Block Diagram for the Mass Matrix $[m]$. 

$[N] = \begin{bmatrix}
N_1 & 0 & 0 & N_2 & 0 & 0 & \ldots & \ldots & N_{32} & 0 & 0 \\
0 & N_1 & 0 & 0 & N_2 & 0 & \ldots & \ldots & 0 & N_{32} & 0 \\
0 & 0 & N_1 & 0 & 0 & N_2 & \ldots & \ldots & 0 & 0 & N_{32}
\end{bmatrix}$

Figure 2.10 The Shape Function Matrix $[N]$. 

4. Computation of consistent load vector $\{f\}$.

The explicit formula for the load vector $\{f\}$ is:

$$\{f\} = \int_{\xi_1}^{\xi_2} \int_{\eta_1}^{\eta_2} \begin{bmatrix} f_{vx} \\ f_{vy} \\ f_{vz} \end{bmatrix} \det(J) \, d\xi \, d\eta \, d\xi$$

(eqns 2.9)

has the block diagram as Figure 2.12 and numerical integration form is:

$$\{f\} = \sum_{i=1}^{np} w_i [N]^{[f]} \begin{bmatrix} f_{vx} \\ f_{vy} \\ f_{vz} \end{bmatrix} \det(J)$$

(eqns 2.10)
\[ [N|N] = \begin{bmatrix}
N_1^1 & 0 & 0 & N_1^2 & 0 & 0 & N_1^3 & 0 & 0 \\
0 & N_2^1 & 0 & 0 & N_2^2 & 0 & 0 & N_2^3 & 0 \\
0 & 0 & N_3^1 & 0 & 0 & N_3^2 & 0 & 0 & N_3^3 \\
N_4^1 & 0 & 0 & N_4^2 & 0 & 0 & N_4^3 & 0 & 0 \\
0 & N_5^1 & 0 & 0 & N_5^2 & 0 & 0 & N_5^3 & 0 \\
0 & 0 & N_6^1 & 0 & 0 & N_6^2 & 0 & 0 & N_6^3 \\
N_7^1 & 0 & 0 & N_7^2 & 0 & 0 & N_7^3 & 0 & 0 \\
0 & N_8^1 & 0 & 0 & N_8^2 & 0 & 0 & N_8^3 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\end{bmatrix} \]

Figure 2.11 The Product of Matrices \([N]^T\) and \([N]\).

\(f_x, f_y, f_z\) are the force per unit volume in direction \(x, y, z\).

5. Computation the residue array.

The element residual array has the block diagram as Figure 2.13 and defined by:

\[
\{r\} = \{f\} - \{k\}\{u_n\} \quad \text{(eqn 2.11)}
\]

\(\text{where}\)

- \(\{r\}\) is the residue vector;
- \(\{f\}\) is the element load vector;
- \(\{k\}\) is the element stiffness matrix;
- \(\{u_n\}\) is the nodal values vector.

6. Computation of the gradients and stresses at the points of integration.

For each integration point:

- compute strain as Figure 2.14. [Ref. 3:p. 31]

or compacted form

\[
\{c\} = [B]\{u_i\} \quad \text{(eqn 2.12)}
\]
Figure 2.12 Block Diagram for Consistent Load Vector \([\mathbf{f}]\).

Figure 2.13 Block Diagram for the Residue Array.

- compute stress as Figure 2.15. [Ref. 3:p. 30]

or compacted form
\[
\begin{align*}
\begin{bmatrix}
\sigma_x \\
\sigma_y \\
\sigma_z \\
\tau_{xy} \\
\tau_{yz} \\
\tau_{zx}
\end{bmatrix} &= \frac{E(1-\nu)}{(1-\nu)(1-2\nu)}
\begin{bmatrix}
1 & \nu & \nu & 0 & 0 & 0 \\
\nu & 1-\nu & \nu & 0 & 0 & 0 \\
\nu & \nu & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1-2\nu}{2(1-\nu)} & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1-2\nu}{2(1-\nu)} & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{1-2\nu}{2(1-\nu)}
\end{bmatrix}
\begin{bmatrix}
\varepsilon_x \\
\varepsilon_y \\
\varepsilon_z \\
\gamma_{xy} \\
\gamma_{yz} \\
\gamma_{zx}
\end{bmatrix}
\end{align*}
\]

Figure 2.15 The Stress-Strain Relation.

\[
\{\sigma_i\} = [D]\{\varepsilon_i\}
\]  
(eqn 2.13)

- compute coordinate of each integration points on the real element as figure 2.16.

or compacted form

\[
\{x_i\} = [N]\{x_{ni}\}
\]  
(eqn 2.14)

Using \(x_n, y_n, z_n\), where \(N = N(\xi_i, \eta_i, \zeta_i)\) and \(\xi_i, \eta_i, \zeta_i\) are the gauss points, the coordinate of thirty two nodes on the real element that existed in the array VCORE. All the integration points are evaluated in the reference element. Using the product of the transfer matrix (the shape function matrix \(N\)) and the array VCORE, we can evaluate the coordinate of the integration points on the real element.
Figure 2.16 Evaluation of the Coordinate of the Integration Points on the Real Element.

Note that subroutine ELEM03 executes one operation at a time depending on the value of ICODE. For preserving the memory locations, some of the operations have been performed using the same array name as VKE in both the stiffness matrix \([k]\) and the mass matrix \([m]\). The same thing has been done with array VFE used for the residual vector \(\{r\}\) and the load vector \(\{f\}\).
III. THE SOLUTION OF A SIMPLE PROBLEM AND DISCUSSION OF RESULT

In this chapter the preparation of input data for the computer program are described. A simple problem was developed and the investigation was conducted to determine the ability of this cubic element.

A. ENTRY AND EXECUTION FUNCTIONAL BLOCKS

MEF has specialized functional blocks for the entry, verification and organization of the data required to define a problem. Block COOR reads the nodal coordinates and the number of degrees of freedom of each node, it also provides automatic node generation. Block COND reads the boundary condition. Block PREL reads element properties if required for element type being used. Block SOLC reads the concentrated loads. Block ELEM reads the element connectivities; it also reads element group information when more than one element type is used, if elements have different properties. This block provides automatic element generation.

Other function blocks of MEF for the execution of particular finite element computations use the data base constructed by entry blocks and augment it by their results. Block LINM assembles and solves a linear system of equations residing in-core. Block LIND is similar to the block LINM but the system of equations resides out-of-core in a mass storage device. And block STOP terminates execution of the problem.

MEF provides various levels of output. The quantity of output desired from a given block is controlled by a parameter on the block calling card, described in detail [Ref. 1:pp. 440-447], which ranges from 0 (the assumed value) to 4. The default value provides all the information needed to verify the input stream and obtain the desired answers while the value 1 thru 4 provide various level of verbosity.

B. STRUCTURE MODELING

The first step in applying a finite element solution to the problem is the selection of an appropriate mesh. In many case an extrapolation of the results will be required and hence more than one mesh will have to be selected. The mesh size solution does not follow any predetermined rules and will, in general, depend on the nature of the problem and the judgement of the analyst. An arbitrary numbering convention is
adopted for the nodal points of the entire structure (in distinction with the numbering convention for the element nodes shown in Figure 3.1. A second numbering scheme is also needed for the elements of a mesh.

In order to show the function of the program, as well as some observations affecting the use of it, the solution of a simple problem is presented in this section. The problem selected is that usually presented in classical texts of strength of materials as a cantilever beam of uniform cross section subjected to a concentrated load at the end of the beam. The model used for this problem is shown in Figure 3.1. Nodes 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 and 12 (here arbitrarily numbered) are constrained to zero displacement in all directions. The concentrated load at the end of the cantilever beam is considered as the consistent load and are shown in Figure 3.2.

Four different meshes which each mesh has the thickness 9, 0.9, 0.09, 0.009 inches and has the concentrated loads 1200, 1.2, 1.2e-3 and 1.2e-6 pounds were employed in order to show the variation of results with mesh size. Numerical results for the maximum displacement at the tip of the beam are shown in Tables I and II.
C. DISCUSSION OF RESULTS

Elementary beam theory gives a displacement of 0.0631 inches, however this beam theory neglects shear deflection and three dimensional elasticity does not do so. Observation of the results presented here reveal that mesh refinement lead to improved results which eventually will converge to a certain value. The thirty two nodal points brick is subject to numerical bad conditionary when used with an adverse slenderness ratio. This ratio being defined as the ratio of the maximum element dimension over the minimum dimension. For example in the numerical examples treated the slenderness ratio for the one and eight elements representation of the four beams analysis are shown in the Tables I and II. Results are acceptable for the first three beams in each case. However a computation of approximate values of zero, first invariant of the element stiffness as well as condition number are produced to investigate the results.

As the slenderness ratio increases the numerical of the conditioning of the stiffness matrices become so bad that no significant digits can be expected out of the solution for displacements. The same conclusion is arrived with a mesh of eight elements. In such a case the slenderness ratio varies from 1.6 to 1666.6. For the thickness of 0.09 the slenderness ratio was adequate. To reduce the ratio, more element must be used and tests with 20 and 40 elements were conducted. The twenty elements mesh gives of 0.6 to 666.6. In the use of a mesh with fourty elements the largest dimension become 6.0 and must therefore be reduced to 3.0 as 3.0 is the value
of 120/40. Runs with such a mesh were then performed to confirm our results. On the basis of all these runs and the value of the condition number of these matrices it seems that a slenderness ratio of approximately 150 could still be used to give results with 6 significant digits in the answers, however further examples must be treated before a conclusion could be reached.

These are observations essentially made on the results of a simple problem. Thus no firm rules regarding the use of the newly element can be established and further experimentation with different problems has to be conducted for this purpose.
### TABLE I
THE DISPLACEMENT FOR THE ONE ELEMENT

<table>
<thead>
<tr>
<th>Thickness</th>
<th>Slenderness Ratio</th>
<th>Zero</th>
<th>Σ$s_{ij}$</th>
<th>Σ$λ_{ij}$</th>
<th>$λ_{max}$</th>
<th>$λ_{min}$</th>
<th>Condition Number</th>
<th>Displacement</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>13.3</td>
<td>0.24e-5</td>
<td>-0.54e-5</td>
<td>7.27e11</td>
<td>3.72e10</td>
<td>3.14e+4</td>
<td>1.18e06</td>
<td>-0.0581</td>
</tr>
<tr>
<td>0.9</td>
<td>133.3</td>
<td>0.76e-5</td>
<td>-0.16e-4</td>
<td>7.11e12</td>
<td>1.64e11</td>
<td>7.42e+1</td>
<td>2.22e09</td>
<td>-0.0533</td>
</tr>
<tr>
<td>0.09</td>
<td>1333.3</td>
<td>0.74e-4</td>
<td>0.16e-2</td>
<td>7.11e13</td>
<td>1.64e12</td>
<td>7.42e-2</td>
<td>2.22e13</td>
<td>-0.0686</td>
</tr>
<tr>
<td>0.009</td>
<td>13333.3</td>
<td>0.74e-3</td>
<td>0.56e-6</td>
<td>7.11e14</td>
<td>1.64e13</td>
<td>7.42e-3</td>
<td>4.59e15</td>
<td>*</td>
</tr>
</tbody>
</table>

### TABLE II
THE DISPLACEMENT FOR THE EIGHT ELEMENTS

<table>
<thead>
<tr>
<th>Thickness</th>
<th>Slenderness Ratio</th>
<th>Zero</th>
<th>Σ$s_{ij}$</th>
<th>Σ$λ_{ij}$</th>
<th>$λ_{max}$</th>
<th>$λ_{min}$</th>
<th>Condition Number</th>
<th>Displacement</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>1.66</td>
<td>0.33e-6</td>
<td>0.23e-5</td>
<td>3.21e10</td>
<td>4.66e09</td>
<td>1.86e06</td>
<td>2.54e03</td>
<td>-0.0623</td>
</tr>
<tr>
<td>0.9</td>
<td>16.6</td>
<td>0.96e-6</td>
<td>0.12e-4</td>
<td>9.12e10</td>
<td>2.06e10</td>
<td>1.60e04</td>
<td>1.29e06</td>
<td>-0.0614</td>
</tr>
<tr>
<td>0.09</td>
<td>166.6</td>
<td>0.93e-5</td>
<td>0.97e-4</td>
<td>8.89e11</td>
<td>2.06e11</td>
<td>1.65e01</td>
<td>1.29e10</td>
<td>-0.0597</td>
</tr>
<tr>
<td>0.009</td>
<td>1666.6</td>
<td>0.93e-4</td>
<td>0.17e-2</td>
<td>8.88e12</td>
<td>2.06e12</td>
<td>1.36e-2</td>
<td>1.32e14</td>
<td>*</td>
</tr>
</tbody>
</table>
APPENDIX

ELEM03 SUBPROGRAM LISTING

The listing of subprogram ELEM03 is provided below. It includes four subroutines: N103, DO3, BO3 and BTDB respectively.

```fortran
SUBROUTINE ELEMO3(VCORE, VPRNE, VPREE, VDLE, VKE, VFE)
C******************************************************************************
C 32 NODES HEXAHEDRON ELEMENT FOR 3 DIMENSIONAL ELASTICITY
EVALUATE ELEMENT INFORMATIONS ACCORDING TO ICODE VALUE
ICODE=1 ELEMENT PARAMETERS
ICODE=2 INTERPOLATION FUNCTIONS AND GAUSS COEFFICIENTS
ICODE=3 STIFFNESS MATRIX
ICODE=4 MASS MATRIX
ICODE=5 RESIDUALS
ICODE=6 EVALUATE AND PRINT STRESSES
 ELEMENT PROPERTIES
VPREE(1) YOUNG'S MODULUS
VPREE(2) POISSON'S COEFFICIENT
VPREE(3) SPECIFIC MASS
******************************************************************************
IMPLICIT REAL*8(A-H,O-Z)
COMMON /COOR/NDIM
COMMON /ASSE/NSYM
COMMON /RGDT/IPE, IPEI, IPEE, ICIE, IPRNE, IPREE, INEL, IDEG, IPGICODE, IPGO
COMMON /ES/MR MP
DIMENSION VCORE(1), VPRNE(1), VPREE(1), VDLE(1), VKE(1), VFE(1)
C------ CHARACTERISTIC DIMENSIONS OF THE ELEMENT
DIMENSION VCPG(IPG), VKPG(NDIM*IPG), VDE1(IMATD*2)
DIMENSION VCPS(27), VKPS(81), VDE1(36)
DIMENSION VBDE(IMATD*2), VDE1(IMATD*2), VJ(NDIM**2), VJ(NDIM**2)
DIMENSION VNI(NIN*NDIM), VNI(NDIM**2), INEL*IPG), IPGKED(NDIM)
DIMENSION VNIX(96), VNIX(3456), IPGKED(3)
C------ DIMENSION OF MATRIX D, NUMBER OF G.P.
DATA IMATD/6/, IPGKED/3.3.3/
DATA ZERO/0.000/, DEUX/2.00/, X05/0.500/, RADN/.572957795130823D2/
DATA EPS/1.0D-6/
DATA NNNNNN/0/
C------ CHOOSE FUNCTION TO BE EXECUTED
GOTO (100,200,300,400,500,600,700,800), ICODE
100 IDLEO=96
INELO=32
IPGO=27
RETURN
C******************************************************************************
C EVALUATE COORDINATES,WEIGHTS,FUNCTIONS N AND THEIR DERIVATIVES AT G.P.
******************************************************************************
200 CALL GAUSS(IPGKED, NDIM, VKPG, VCPG, IPG)
IF(M.LT.2) GOTO 220
```
WRITE(*,2000) IPG

2000 FORMAT(/15,' GAUSS POINTS',/10X,'VCPG',25X,'VKPG')

DO 210 IG=1,IPG
I=10-NDIM+1
WRITE(*,2010) VCPG(IG),(VKPG(I),I=I0,I1)

210 FORMAT(1X,F13.9,5X,3F13.9)
DO 220 IPG=1,IPG

220 FORMAT(1X,'FUNCTION AND DERIVATIVES'/(1X,8E12.5))
RETURN

C******************************************************************************
C** EVALUATE ELEMENT STIFFNESS MATRIX
C******************************************************************************

C------------ INITIALIZE VKE
300 IF(NSYM.NE.0) NINI=9216
do 310 I=1,NINI
310 VKE(I)=ZERO

C------------ FORM MATRIX D
CALL DO3(VPREE,VDE)

C------------ LOOP OVER THE G.P.
320 DO 330 I=1,NIPG

C------------ EVALUATE THE JACOBIAN ITS INVERSE AND ITS DETERMINANT
CALL JACOB(VNI(I),VCORE,NDIM,INEL,VJ,VJ1,DETJ)

C------------ PERFORM D*COEF
C=VCPG(IG)*DETJ
320 DO 360 I=1,NIPG
360 DO 370 I=1,NDIM,1
370 DO 380 J=1,NDIM,1
380 D=VCPG(IG)*DETJ*VPREE(LL)
360 D=VDE1(I)=VDE1(I)*C
370 D=VDE1(I)=VDE1(I)*C
380 D=VDE1(I)=VDE1(I)*C
390 DO 400 J=1,INEL
400 I=1+INEL
410 CALL DNDX(VNI(I),VJ,NDIM,INEL,VNI)
420 CALL B03(VNI,INEL,VBE)
430 CALL BTOB(VKE,VBE;VDE1,IDLE,IMATD,NSYM)
440 I=1+INEL
330 RETURN
RETURN

C******************************************************************************
C** EVALUATE THE MASS MATRIX
C******************************************************************************

500 IF(NSYM.NE.0) NINI=9216
do 510 I=1,NINI
510 VKE(I)=ZERO

C---- LOOP OVER THE G.P.
520 IDIM=NDIM-1
530 IDEC=(NDIM+1)*INEL
I=1+INEL
540 DO 550 IG=1,IPG
CALL JACOB(VNI(I1),VCORE,NDIM,INEL,VJ,VJ1,DETJ)
550 LL=3
560 D=VCPG(IG)*DETJ*VPREE(LL)

C---- ACCUMULATE MASS TERMS

C
C

30
VESJOI=VKE(JO)+C

DO 520 II=1,IDIM
VKE(II)=VKE(JI)+C
520 JI=JI+ID+3

JO=JO+NDIM
540 IDL=IDL+NDIM
II=II+IDENTL
RETURN

*******************************************************************************

C EVALUATE THE ELEMENT RESIDUAL
*******************************************************************************
C
C FORM MATRIX D
C
C 600 CALL DO3(VPREE,VDE)
C
C INITIALIZE THE RESIDUAL VECTOR
C
DO 610 ID=1,IDIM
VFE(ID)=ZERO
610
C LOOP OVER THE G.P.
C
II=1+INEL
DO 640 IG=1,IPG
C EVALUATE THE JACOBIAN
CALL JACOB(VNI(I),VNOR,NDIM,INEL,VJ,VJ1,DETJ)
C EVALUATE FUNCTIONS D(NI)/D(X)
CALL DNIX(VNI(I),VJ1,NDIM,INEL,VNIX)
C EVALUATE STRAINS AND STRESSES
EPSX=ZERO
EPSY=ZERO
EPSZ=ZERO
GAMXY=ZERO
GAMYZ=ZERO
GAMZX=ZERO

DO 620 IN=1,INEL
UN=VDE(ID)
VN=VDE(ID+1)
WN=VDE(ID+2)
C1=VNIX(IN)
IN1=IN+INEL
C2=VNIX(IN1)
IN2=IN1+INEL
C3=VNIX(IN2)
EPSX=EPSX+C1*UN
EPSY=EPSY+C2*VN
EPSZ=EPSZ+C3*WN
GAMXY=GAMXY+C1*VN+C2*UN
GAMYZ=GAMYZ+C2*WN+C3*VN
GAMZX=GAMZX+C1*WN+C3*UN
620

C FORM THE RESIDUAL

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C
DO 630 IN=1, INEL
C1=VNIX(IN)
IN1=IN+1
C2=VNIX(IN1)
IN2=IN1+1
C3=VNIX(IN2)
VFE(ID)=VFE(ID)+C1*SIGX+C2*TAUXY+C3*TAUZX
VFE(ID+1)=VFE(ID+1)+C2*SIGY+C1*TAUXY+C3*TAUZY
VFE(ID+2)=VFE(ID+2)+C3*SIGZ+C2*TAUZY+C1*TAUZX
630 ID=ID+3
RETURN
C******************************************************************************
C EVALUATE BODY FORCES, FX,FY,FZ PER UNIT VOLUME
C******************************************************************************
700 FX=ZERO
FY=ZERO
FZ=-VPREE(LL)
CPEL=EVALUATE BODY FORCES, FX,FY,FZ PER UNIT VOLUME
DO 710 I=1,PEL
VFE(I)=ZERO
11=1
IDECL=(NDIM+1)*INEL
DO 730 IG=1,IPG
CALL JACOB(VNI(I1+INEL),VCORE,NDIM,INEL,VJ,VJ1,DETJ)
DX=UCPG(IG)*DETJ
D1=DX*FX
D2=DX*FY
D3=DX*FZ
12=I1+3
710 I1=I1+IDECL
RETURN
C******************************************************************************
C EVALUATE AND PRINT STRESS AT G.P.
C******************************************************************************
800 WRITE(MP,2080) IEL
2080 FORMAT(/' STRESSES IN ELEMENT',I5,' P.G.',6X,'X',I1X,'Y',I1X,
9X,'SIGX',8X,'SIGY',8X,'SIGZ',7X,'TAUXY',7X,'TAUZY',7X,'TAUZX'/)
C------- FORM THE MATRIX D
C CALL D03(VPREE,VDE)
C------- LOOP OVER THE G.P.
C I1=1+INEL
C I2=0
DO 820 IG=1,IPG
C------- EVALUATE THE JACOBIAN
CALL JACOB(VNI(I1),VCORE,NDIM,INEL,VJ,VJ1,DETJ)
C------- EVALUATE FUNCTIONS D(NI)/D(X)
CALL DNIDX(VNI(I1),VJ1,NDIM,INEL,VNIX)
C------- COMPUTE STRAINS AND COORDINATE AT G.P.
EPSX=ZERO
EPSY=ZERO
EPSZ=ZERO
GAMXY=ZERO
32
**GAMZ = ZERO**

**GX = ZERO**

**X = ZERO**

**Y = ZERO**

**Z = ZERO**

**DO 810 IN = 1, INEL**

**UN = V0LE(iD)**

**VN = V0LE(iD+1)**

**WN = V0LE(iD+2)**

**XN = VCORE(iD)**

**ZN = VCORE(iD+2)**

**C1 = VNIX(IN)**

**IN1 = IN + INEL**

**C2 = VNIX(IN1)**

**IN2 = IN1 + INEL**

**C3 = VNIX(IN2)**

**C4 = VNIX(IN1)**

**EPSX = EPSX + C1*UN**

**EPSZ = EPSZ + C5*WN**

**GAMXY = GAMXY + C1*VN + C2*UN**

**GAMYZ = GAMYZ + C2*WN + C3*VN**

**GAMZX = GAMZX + C1*WN + C3*UN**

**X = X + C4*XN**

**Y = Y + C4*YN**

**Z = Z + C4*ZN**

**810 ID = ID + 3**

------------ COMPUTE THE STRESSES ------------

**SIGX = VDE(1)*EPSX + VDE(2)*EPSY + VDE(2)*EPSZ**

**SIGZ = VDE(2)*EPSX + VDE(1)*EPSY + VDE(1)*EPSZ**

**TAUXY = VDE(22)*GAMXY**

**TAUZ = VDE(22)*GAMZX**

**WRITE(MP. 2050) ID, X, Y, Z, SIGX, SIGY, SIGZ, TAUXY, TAUZ, TAUZX**

**820 **

**RETURN**

END

---

**TO EVALUATE THE SHAPE FUNCTIONS N AND THEIR DERIVATIVES W.R.T. KSI, ETA, DZETA**

**INPUT**

**VPKG(N) = COORDINATES OF POINTS IN KSI, ETA, DZETA**

**OUTPUT**

**VNI = THE SHAPE FUNCTION N AND THE DERIVATIVE OF SHAPE FUNCTION W.R.T. KSI, ETA, DZETA**

---

**IMPLICIT REAL*8 (A-H, O-Z)**

**DIMENSION VNI(3456), CORRF(32, 3), VPKG(81), VN(32, 3), RN(32)**

**DATA ((CORRF(I, J), I = 1, 3), J = 1, 3) = (1, 3)**

---

The rest of the code snippet is not transcribed due to its length and complexity, but it continues with the evaluation of shape functions and their derivatives.
C
WRITE(*,99) I,(CORRF(I,J))=J=1,3
C
FORMAT(2X,2F2.6)
DO J=1,3
C
X=VKPG(NN)
Y=VKPG(NN+1)
Z=VKPG(NN+2)
PRINT* X,Y,Z
C
---------- AT CONNER
---------- NODE #: 1,4,7,10,21,24,27,30
C
DO 100 I=1,2
II=I+1
DO 100 J=II,IT,3
X1=CORRF(J,1)
Y1=CORRF(J,2)
Z1=CORRF(J,3)
CF1=81.0
100 RN(J)=CF1*(1.0+X*X1)*(1.0+Y*Y1)*(1.0+Z*Z1)*(-CF2+X*X+Y*Y+Z*Z)
VN(J,1)=CF1*(1.0+X*X1)*(1.0+Y*Y1)*(1.0+Z*Z1)*(-CF2+X*X+Y*Y+Z*Z)
*+2.00*Y)
VN(J,2)=CF1*(1.0+X*X1)*(1.0+Y*Y1)*(1.0+Z*Z1)*(-CF2+X*X+Y*Y+Z*Z)
*+2.00*Y)
VN(J,3)=CF1*(1.0+X*X1)*(1.0+Y*Y1)*(1.0+Z*Z1)*(-CF2+X*X+Y*Y+Z*Z)
C
---------- AT MIDSIDE
---------- NODE #: 2,3,8,9,22,23,28,29
C
DO 200 K=1,2
IK=20*(K-1)
DO 200 I=I+1
II=I+1
DO 200 J=II,IT
X1=CORRF(J,1)
Y1=CORRF(J,2)
Z1=CORRF(J,3)
CF1=81.00
DO 64.00
CF2=1.00/9.00
DO 9.00
RN(J)=CF1*(1.00+X*X1)*(1.00+Y*Y1)*(1.00+Z*Z1)*(-CF2+X*X+Y*Y+Z*Z)
VN(J,1)=CF1*(1.00+X*X1)*(1.00+Y*Y1)*(1.00+Z*Z1)*(-CF2+X*X+Y*Y+Z*Z)
*+2.00*Y)
VN(J,2)=CF1*(1.00+X*X1)*(1.00+Y*Y1)*(1.00+Z*Z1)*(-CF2+X*X+Y*Y+Z*Z)
*+2.00*Y)
VN(J,3)=CF1*(1.00+X*X1)*(1.00+Y*Y1)*(1.00+Z*Z1)*(-CF2+X*X+Y*Y+Z*Z)
C
---------- AT MIDSIDE
---------- NODE #: 5,6,11,12,25,26,31,32
C
DO 300 K=1,2
IK=20*(K-1)
DO 300 I=1,2
II=I+1
DO 300 J=II,IT
X1=CORRF(J,1)
Y1=CORRF(J,2)
Z1=CORRF(J,3)
RN(J)=CF1*(1.00+X*X1)*(1.00+Y*Y1)*(1.00+Z*Z1)*(-CF2+X*X+Y*Y+Z*Z)
VN(J,1)=CF1*(1.00+X*X1)*(1.00+Y*Y1)*(1.00+Z*Z1)*(-CF2+X*X+Y*Y+Z*Z)
*+2.00*Y)
VN(J,2)=CF1*(1.00+X*X1)*(1.00+Y*Y1)*(1.00+Z*Z1)*(-CF2+X*X+Y*Y+Z*Z)
*+2.00*Y)
VN(J,3)=CF1*(1.00+X*X1)*(1.00+Y*Y1)*(1.00+Z*Z1)*(-CF2+X*X+Y*Y+Z*Z)
C
---------- AT MIDSIDE
---------- NODE #: 13,14,15,16,17,18,19,20
C
---------- AT MIDSIDE
---------- NODE #: 1,4,7,10,21,24,27,30
C
---------- AT MIDSIDE
---------- NODE #: 2,3,8,9,22,23,28,29
C
---------- AT MIDSIDE
---------- NODE #: 5,6,11,12,25,26,31,32
C
---------- AT MIDSIDE
---------- NODE #: 13,14,15,16,17,18,19,20
C
DO 400 J=13,20
X(J)=CORRF(J,1)
Y(J)=CORRF(J,2)
Z(J)=CORRF(J,3)
RN(J)=CF1*(1.DO+X*X1)*(1.DO+Y*Y1)*(1.DO-Z*Z)*(CF2+Z*Z1)
VN(J,1)=CF1*X1*(1.DO+Y*Y1)*(1.DO-Z*Z)*(CF2+Z*Z1)
VN(J,2)=CF1*Y1*(1.DO+X*X1)*(1.DO-Z*Z)*(CF2+Z*Z1)
VN(J,3)=CF1*(1.DO+X*X1)*(1.DO+Y*Y1)*(Z1-2.DO-Z*CF2-3.DO*Z*Z1)
DO 400 J=1,32
VN(J,0)=RN(J)
410 JU=J+1
DO 420 K=1,3
DO 430 L=1,32
VN(J,J)=VN(I,K)
JU=J+1
420 CONTINUE
10 CONTINUE
RETURN
END

SUBROUTINE DO3(VPREE,VDE)
C********************************************
TO FORM MATRIX D ( 3 DIMENSIONAL ELASTICITY)
INPUT
VPREE = ELEMENT PROPERTY
    VPREE(1) = YOUNG'S MODULUS
    VPREE(2) = POISSON'S RATIO
OUTPUT
VDE = MATRIX D
C********************************************
IMPLICIT REAL*8 (A-H O-Z)
DIMENSION VPREE(1),VDE(36)
DATA ZERO/0.000/,UN/1.000/,DEUX/2.000/
E=VPREE(1)
LL=2
C1=E*(UN-X)/(UN-DU+X*(UN-DEUX*X))
C2=C1*(UN-DU+X)/(DEUX*(UN-X))
C3=C1*(UN-DU+X)*(DEUX*(UN-X))
DO 10 J=1,36
10 VDE(J)=ZERO
VDE(1)=C1
VDE(2)=C2
VDE(3)=C3
VDE(4)=C3
VDE(5)=C3
VDE(6)=C3
VDE(7)=C3
VDE(8)=C3
VDE(9)=C3
VDE(10)=C3
VDE(11)=C3
VDE(12)=C3
VDE(13)=C3
RETURN
END

SUBROUTINE DO3(VNIX,INEL,VBE)
C********************************************
TO FORM MATRIX B ( 3 DIMENSIONAL ELASTICITY)
INPUT
VNIX = DERIVATIVES OF SHAPE FUNCTION W.R.T. X,Y,Z
OUTPUT
VBE = MATRIX B
C********************************************
IMPLICIT REAL*8 (A-H O-Z)
DIMENSION VNIX(3,1),VBE(61)
DO 10 I=1,6
DO 10 J=1,36
10 VBE(I,J)=0.000
FORMATION OF MATRIX B

```
C DO 20 I=1,32
I1=3*I-2
I2=I1+1
I3=I2+1
IK=1
CI=VIX(1,KK)
C2=VIX(1,(KK+1))
C3=VIX(1,(KK+2))
VBE(1,1)=C1
VBE(2,2)=C2
VBE(3,3)=C3
VBE(4,1)=C2
VBE(4,2)=C1
VBE(5,2)=C3
VBE(6,1)=C3
VBE(6,3)=C1
20 RETURN

SUBROUTINE BTDB(VKE, VBE, VDE, IDLE, IMATD, NSYM)
C TO ADD THE PRODUCT B(i).D.B TO THE ELEMENT MATRIX K
C INPUT
VBE = MATRIX B
VDE = MATRIX D
C OUTPUT
VKE = ELEMENT MATRIX K
C
IMPLICIT REAL*8 (A-H,O-Z)
DIMENSION VKE(1), VBE(IMATD,1), VDE(IMATD,1), T(576)
DATA ZERO/0.000/
IJ=1
IMAX=IDLE
DO 40 J=1,IDLE
DO 20 I1=1,IMATD
C=ZERO
DO 10 J1=I1,IMATD
10 C=C+VDE(I1,J1)*VBE(J1,J)
20 T(J1)=C
IF(NSYM.EQ.O) IMAX=J
DO 40 I=1,IMAX
C=ZERO
DO 30 J1=I,IMATD
30 C=C+VBE(J1,I)*T(J1)
VKE(I,J)=VKE(I,J)+C
40 I=I+1
RETURN
END
```
LIST OF REFERENCES


<table>
<thead>
<tr>
<th>No.</th>
<th>Distribution Details</th>
<th>Copies</th>
</tr>
</thead>
</table>
| 1.  | Defense Technical Information Center  
     Cameron Station  
     Alexandria, Virginia  22304-6145 | 2 |
| 2.  | Library, Code 0142  
     Naval Postgraduate School  
     Monterey, California  93943-5002 | 2 |
| 3.  | Department Chairman, Code 69  
     Department Of Mechanical Engineering  
     Naval Postgraduate School  
     Monterey, California  93943-5000 | 1 |
| 4.  | Professor Gilles Cantin, Code 69Ci  
     Department Of Mechanical Engineering  
     Naval Postgraduate School  
     Monterey, California  93943-5000 | 4 |
| 5.  | Library  
     Royal Thai Naval Academy  
     Parknam, Samutplakan, Thailand | 1 |
| 6.  | LT Anan Sukaneeyouth  
     Nor-96, Naval Fleet Village  
     Sattahip, Chonburi, Thailand | 3 |
END

3-87

Dtic