LINEAR AND NONLINEAR ANALYSIS OF SPACE TRUSSES
RELATIVE TO CYLINDRICAL COORDINATES

by

Jeffrey A. Perrier

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of
MASTER OF SCIENCE
in
Civil Engineering

APPROVED:

S. M. Holzer, Chairman

A. E. Somers, Jr. R. M. Barker

December, 1982
Blacksburg, Virginia
ACKNOWLEDGEMENTS

The author would like to gratefully acknowledge Dr. S. M. Holzer for his guidance, comments, and suggestions, and for his time in proof-reading parts of the thesis.

Appreciation is given to Dr. L. T. Watson for his helpful suggestions, and to Dr. A. E. Somers, Jr. and Dr. R. M. Barker for their helpful comments and for serving on the committee.

The author gratefully acknowledges the financial support provided by the Department of Civil Engineering and the use of the computer facilities of Virginia Polytechnic Institute and State University.

Comments and assistance from fellow graduate students, particularly Edgar E. Lamma, III, G. Stuart Matthis, II, and Douglas G. Fitzpatrick, are also appreciated.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>ii</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>iv</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>v</td>
</tr>
<tr>
<td>1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>2. STRUCTURAL MODELS</td>
<td>3</td>
</tr>
<tr>
<td>2.1 Cylindrical Coordinate Systems</td>
<td>3</td>
</tr>
<tr>
<td>2.2 Radial Symmetry</td>
<td>3</td>
</tr>
<tr>
<td>2.3 Linear Element Model</td>
<td>7</td>
</tr>
<tr>
<td>2.4 Linear System Model</td>
<td>18</td>
</tr>
<tr>
<td>2.5 Nonlinear Element Model</td>
<td>19</td>
</tr>
<tr>
<td>2.6 Nonlinear System Model</td>
<td>30</td>
</tr>
<tr>
<td>2.7 Stability of Equilibrium</td>
<td>30</td>
</tr>
<tr>
<td>2.8 Solutions of Systems of Linear Algebraic Equations</td>
<td>31</td>
</tr>
<tr>
<td>3. DISCUSSION OF RESULTS</td>
<td>34</td>
</tr>
<tr>
<td>3.1 Linear Analysis</td>
<td>34</td>
</tr>
<tr>
<td>3.1.1 Structure 1</td>
<td>34</td>
</tr>
<tr>
<td>3.1.2 Structure 2</td>
<td>36</td>
</tr>
<tr>
<td>3.1.3 Comparison of Effort</td>
<td>36</td>
</tr>
<tr>
<td>3.2 Nonlinear Analysis</td>
<td>42</td>
</tr>
<tr>
<td>3.2.1 Structure 1</td>
<td>42</td>
</tr>
<tr>
<td>3.2.2 Structure 2</td>
<td>47</td>
</tr>
<tr>
<td>4. CONCLUSION AND RECOMMENDATIONS</td>
<td>53</td>
</tr>
<tr>
<td>APPENDIX A - REFERENCES</td>
<td>57</td>
</tr>
<tr>
<td>APPENDIX B - LINEAR PROGRAM</td>
<td>60</td>
</tr>
<tr>
<td>APPENDIX C - NONLINEAR PROGRAM</td>
<td>89</td>
</tr>
<tr>
<td>VITA</td>
<td>179</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td></td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Cylindrical Coordinate System</td>
<td>4</td>
</tr>
<tr>
<td>2.2</td>
<td>&quot;Pie Slice&quot; of a Structure</td>
<td>6</td>
</tr>
<tr>
<td>2.3</td>
<td>Linear Truss Element</td>
<td>8</td>
</tr>
<tr>
<td>2.4</td>
<td>Linear Model Coordinate Transformations</td>
<td>12</td>
</tr>
<tr>
<td>2.5</td>
<td>Nonlinear Truss Element</td>
<td>20</td>
</tr>
<tr>
<td>2.6</td>
<td>Nonlinear Model Coordinate Transformations</td>
<td>27</td>
</tr>
<tr>
<td>3.1</td>
<td>Structure 1</td>
<td>35</td>
</tr>
<tr>
<td>3.2</td>
<td>&quot;Pie Slices&quot; of Structure 1</td>
<td>37</td>
</tr>
<tr>
<td>3.3</td>
<td>Structure 2</td>
<td>38</td>
</tr>
<tr>
<td>3.4</td>
<td>Percent of Effort vs. Percent of Structure</td>
<td>41</td>
</tr>
<tr>
<td>3.5</td>
<td>Nonlinear Load vs. Vertical Displacement of Compression Ring Curve for Structure 1 Showing Locations of Bifurcation Points</td>
<td>44</td>
</tr>
<tr>
<td>3.6</td>
<td>Buckled Configuration of Structure 1</td>
<td>46</td>
</tr>
<tr>
<td>3.7</td>
<td>Percent of Effort vs. Percent of Structure</td>
<td>49</td>
</tr>
<tr>
<td>3.8</td>
<td>&quot;Pie Slice&quot; of Structure 2</td>
<td>51</td>
</tr>
<tr>
<td>Table</td>
<td>Computational Effort Required in Linear Analysis</td>
<td>Page</td>
</tr>
<tr>
<td>-------</td>
<td>-------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>3.1</td>
<td>Computational Effort Required in Linear Analysis</td>
<td>40</td>
</tr>
<tr>
<td>3.2</td>
<td>Computational Effort Required in Nonlinear Analysis</td>
<td>48</td>
</tr>
</tbody>
</table>
CHAPTER 1
INTRODUCTION

The purpose of this study is to formulate linear and nonlinear three-dimensional truss element models in cylindrical coordinates, to implement these element models using existing solution techniques, and to investigate the utility of such element models in the analysis of two radially symmetric lamella domes. The linear model is used to perform a joint displacement and element force analysis while the nonlinear model is used to trace the nonlinear load vs. displacement path, to locate bifurcation and limit points, and, frequently, to branch off of the fundamental equilibrium path onto bifurcation paths.

The modified Riks/Wempner method [8,15] for tracing nonlinear equilibrium paths is used in the nonlinear model because of its ability to trace equilibrium paths through limit points into the post-critical range and because of its good convergence characteristics.

The reason for formulating structural models in cylindrical coordinates is three-fold.

1. In the analysis of essentially round structures, generation of input data and interpretation of results is much easier when joint forces, constraints, and displacements are expressed as radial, tangential, or vertical rather than in terms of three arbitrary global directions.
2. Under certain conditions, a cylindrical coordinate system allows imposition of the appropriate boundary conditions to use a "pie slice" of a radially symmetric structure to represent the entire structure.

3. When existing methods for branching onto bifurcation paths fail, noting which bifurcation points are suppressed by which "pie slice" configurations provides, at least, qualitative information regarding geometries associated with bifurcation buckling modes.

The only paper located by the author in which a similar model formulation is used is a dissertation by Jagannathan [9] who performs a nonlinear analysis on radially symmetric structures using "pie slices" but does not include the details of his element model formulation.
CHAPTER 2
STRUCTURAL MODELS

In this chapter, the linear and nonlinear truss models used in this study are developed. The cylindrical coordinate system is described and radial symmetry, stability of equilibrium, and solution of systems of linear algebraic equations are discussed.

2.1 Cylindrical Coordinate Systems

The system model is assembled in such a way that the joint forces, displacements, and constraints are expressed relative to distinct joint coordinate systems which together form an overall cylindrical, rather than cartesian, coordinate system.

Fig. 2.1 is the plan view of a reticulated dome showing the orientations of the joint coordinate systems. The overall global coordinate system, shown in dashed lines, is still required to specify the joint coordinates and is used to calculate the element deformations. Since all coordinate systems used in this study are right-handed, both the global and joint 3-directions are down.

2.2 Radial Symmetry

The degree of radial symmetry exhibited by a structure can be designated \( C_n \) after Glockner [4] where \( n \) is effectively the number of identical "pie slices" into which the structure can be divided.
Figure 2.1: Cylindrical Coordinate System
Some structures exhibit several degrees of radial symmetry. For example, a structure which can be divided into four identical parts can also be divided into two identical parts. Therefore, such a structure would exhibit both $C_4$ and $C_2$ radial symmetry. The dome shown in Fig. 2.1 exhibits $C_6$, $C_3$, and $C_2$ radial symmetry. Fig. 2.2 illustrates a "pie slice" equal to one sixth of the dome in Fig 2.1 which could be used to represent the entire structure provided the loading condition also exhibited $C_6$ symmetry. Slices equal to one third of one half of the structure could also be used if the loads exhibited $C_3$ or $C_2$ symmetry respectively. For a "pie slice" to be used to represent the entire structure, both the loads and the structure must exhibit the same degree of symmetry and the "pie slice" itself must exhibit bisymmetry. This latter condition must be met to insure that the tangential constraints applied to the "pie slice" do not restrict any "twisting" of the entire structure which could result from non-bisymmetric slices.

Note that, in Fig. 2.2, since its orientation is arbitrary, the joint coordinate system at the joint lying on the radial axis of symmetry is taken to coincide with the global coordinate axes. Joints 1, 2, and 3 are not constrained against motion in the 3-direction and the original constraint conditions at joint 4 are maintained. For a rigorous discussion of the rules governing division of structures into symmetric parts, see Glockner [4] and Holzer [7]. Let it suffice to say here that 1) the cross-sectional areas of members lying on one cutting plane are reduced by one half, 2) the cross-sectional areas of members lying on the intersection of two
Figure 2.2: "Pie Slice" of a Structure
cutting planes (i.e. the radial axis of symmetry) are divided by the number of slices into which the structure was divided (the subscript, \( n \), in the expression for the degree of radial symmetry, \( C_n \)), 3) the magnitudes of loads applied to joints lying on one cutting plane are reduced by one half, and 4) the magnitudes of loads applied to joints lying on the intersection of two cutting planes are divided by the number of slices into which the structure was divided.

2.3 **Linear Element Model**

The linear truss element model, Fig. 2.3, is from Holzer [5,7] and can be expressed in local element coordinates as

\[
\mathbf{f} = \mathbf{k} \mathbf{d}
\]

(2.1)

where

\[
\begin{align*}
\mathbf{f} &= \begin{bmatrix} f_a \\ f_b \end{bmatrix} \\
\mathbf{k} &= \gamma \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \\
\mathbf{d} &= \begin{bmatrix} d_a \\ d_b \end{bmatrix}
\end{align*}
\]

(2.1a, 2.1b, 2.1d)

\[
\gamma = \frac{AE}{L}
\]

(2.1c)

\( A \) = the cross-sectional area of the element

\( E \) = the modulus of elasticity of the element material

\( L \) = the element length

The coordinate transformation process is a specialization of
Figure 2.3: Linear Truss Element

Since the joint coordinate systems may be different at each end of the element (Figs. 2.1 and 2.2), two transformation matrices, $\lambda$ and $\hat{\lambda}$ (which accomplish the transformation from local to cylindrical coordinates at the $a$- and $b$-ends of the element respectively), are required and are defined such that

\[
\begin{align*}
d_a &= \lambda_D a \\
d_b &= \hat{\lambda}_D b \\
F_a &= \lambda^T f_a \\
F_b &= \hat{\lambda}^T f_b
\end{align*}
\]  

(2.2)

where $d_a$, $f_a$ and $d_b$, $f_b$ are the element end displacements and forces at the $a$- and $b$-ends respectively relative to local element coordinates, and $D_a$, $F_a$ and $D_b$, $F_b$ are the element end displacements and forces at the $a$- and $b$-ends respectively relative to cylindrical coordinates.

Eqs. 2.2 can be expressed more compactly as

\[
\begin{align*}
d &= \Lambda D \\
F &= \Lambda^T f
\end{align*}
\]  

(2.3)

where
Eqs. 2.1 and 2.3 can then be combined to form the linear truss element model in cylindrical coordinates

\[ F = KD \] (2.4)

where

\[ K = \Lambda^T k \Lambda \] (2.4a)

The transformation matrices, \( \lambda \) and \( \hat{\lambda} \), are defined as

\[ \lambda = \begin{bmatrix} c_1 & c_2 & c_3 \end{bmatrix} \] (2.5)

\[ \hat{\lambda} = \begin{bmatrix} \hat{c}_1 & \hat{c}_2 & \hat{c}_3 \end{bmatrix} \]

where \( c_i \) and \( \hat{c}_i \), \( i = 1,2,3 \), are the direction cosines of the element local 1-axis relative to the joint i-axis at the a- and b-ends respectively.

The direction cosines of the element local 1-axis relative to the global coordinate axes, \( \sigma_i \), \( i = 1,2,3 \), are defined as

\[ \sigma_i = \frac{L_i}{L}, \quad i = 1,2,3 \] (2.6)
where $L_i$ is the component of the element length vector in the global $i$-direction.

Referring to Fig. 2.4, which is the projection of an arbitrary element onto the global 1,2-plane, expressions for $c_i$ and $\hat{c}_i$, $i = 1, 2, 3$, are easily formulated. The dashed lines represent the global coordinate system, the solid lines represent the joint coordinate system, and the arrowhead on the element indicates the direction of the element local $l$-axis. Detailed formulations are given only for the expressions for $c_i$, $i = 1, 2, 3$.

The direction cosine of the element local $l$-axis relative to the joint $l$-axis is the ratio of the component of the element length vector in the joint $l$-direction to the total element length.

\[
C_1 = \frac{L_{12}}{L} \cos \phi_1 = \frac{L_{12}}{L} \sin \phi_2 \tag{2.7}
\]

where

\[
L_{12} = \left[ L_1^2 + L_2^2 \right]^{\frac{1}{2}} \tag{2.7a}
\]

Noting that

\[
\phi_2 = \theta + \psi \tag{2.8}
\]

and employing the angle sum relation for the sine function yields

\[
c_1 = \frac{L_{12}}{L} \left[ \sin \theta \cos \psi + \cos \theta \sin \psi \right]
\]

\[
= \frac{L_{12}}{L} \left[ \frac{X_{2a}}{R_a} \left( \frac{L_2}{L_{12}} \right) + \frac{X_{1a}}{R_a} \left( \frac{L_1}{L_{12}} \right) \right]
\]
Figure 2.4: Linear Model Coordinate Transformations
Inserting Eq. 2.6 into the appropriate locations of Eq. 2.9 and rearranging terms finally yields

\[
\begin{align*}
\mathbf{c}_{1} &= \frac{X_{1a}}{R_a} \mathbf{c}_{1} + \frac{X_{2a}}{R_a} \mathbf{c}_{2} \\
\mathbf{c}_{2} &= \frac{L_{12}}{L} \cos \phi_{2}
\end{align*}
\]

Similarly, by definition

\[
\mathbf{c}_{2} = \frac{L_{12}}{L} \cos \phi_{2}
\]

and employing the angle sum relation for the cosine function yields

\[
\mathbf{c}_{2} = -\frac{X_{2a}}{R_a} \mathbf{c}_{1} + \frac{X_{1a}}{R_a} \mathbf{c}_{2}
\]

Since the joint and global 3-axes coincide,

\[
\mathbf{c}_{3} = \mathbf{c}_{3}
\]

for all elements.

The computation of the joint direction cosines at the a-end can be represented in matrix form as

\[
\lambda^T = \begin{bmatrix}
\mathbf{c}_{1} \\
\mathbf{c}_{2} \\
\mathbf{c}_{3}
\end{bmatrix} = \frac{1}{R_a} \begin{bmatrix}
X_{1a} & X_{2a} & 0 \\
-X_{2a} & X_{1a} & 0 \\
0 & 0 & R_a
\end{bmatrix}
\]
where

\[ R_a = \left[ X_{1a}^2 + X_{2a}^2 \right]^{1/2} \]  

(2.14a)

Similarly, for the b-end

\[ \frac{c_1}{R_b} = \frac{X_{1b}}{R_b} \sigma_1 + \frac{X_{2b}}{R_b} \sigma_2 \]  

(2.15)

\[ \frac{c_2}{R_b} = \frac{-X_{2b}}{R_b} \sigma_1 + \frac{X_{1b}}{R_b} \sigma_2 \]  

(2.16)

\[ c_3 = c_3 \]  

(2.17)

or, in matrix form,

\[ T = \begin{bmatrix} \hat{c}_1 \\ \hat{c}_2 \\ \hat{c}_3 \end{bmatrix} = \frac{1}{R_b} \begin{bmatrix} X_{1b} & X_{2b} & 0 \\ -X_{2b} & X_{1b} & 0 \\ 0 & 0 & R_b \end{bmatrix} \]  

(2.18)

where

\[ R_b = \left[ X_{1b}^2 + X_{2b}^2 \right]^{1/2} \]  

(2.18a)

Inserting Eqs. 2.1b and 2.3c into Eq. 2.4a yields

\[ K = \gamma \begin{bmatrix} \lambda^T \lambda & -\lambda^T \hat{T} \\ -\hat{T}^T \lambda & \hat{T}^T \hat{T} \end{bmatrix} \]  

(2.19)

Finally, inserting Eqs 2.5, 2.13, and 2.17 into Eq. 2.19 yields the cylindrical element stiffness matrix.
In radially symmetric structures, certain regularly oriented members are frequently encountered for which significant simplifications can be made in the cylindrical element stiffness matrix.

The first type of element considered is the radial element which is defined as any element which is parallel to or the extension of which intersects the radial axis of symmetry except for those elements which have precisely one end point on the axis. Elements which lie entirely on the radial axis of symmetry, however, are considered to be radial elements. For radial elements, the joint coordinate systems at both ends coincide so that
\[ \hat{c}_1 = c_1 \]  

(2.21)

and there is no tangential component of the element length vector so that

\[ \hat{c}_2 = c_2 = 0 \]  

(2.22)

Elements with precisely one end point on the radial axis of symmetry do not qualify as radial elements because the orientation of the joint coordinate system at the joint on the axis is arbitrary and, therefore, the two joint coordinate systems do not necessarily coincide.

Inserting Eqs. 2.21 and 2.22 into Eq. 2.20 yields the cylindrical stiffness matrix for a radial element

\[
K = \begin{bmatrix}
  g_1 & 0 & g_3 & -g_1 & 0 & -g_3 \\
  0 & 0 & 0 & 0 & 0 & 0 \\
  g_{10} & -g_3 & 0 & -g_{10} \\
  g_1 & 0 & g_3 & 0 & 0 & 0 \\
  \text{sym.} & & & & & \\
  0 & 0 & 0 & 0 & g_{10}
\end{bmatrix}
\]  

(2.23)

where the g functions are defined in Eq. 2.20.

The second commonly encountered element is the ring element which is defined as an element which has both end points equidistant from and lying in a plane perpendicular to the radial axis of symmetry. For a ring element, the joint coordinate systems are oriented relative to each other such that
\[ \hat{c}_1 = -c_1 \quad (2.24) \]
\[ \hat{c}_2 = c_2 \quad (2.25) \]
\[ c_3 = 0 \quad (2.26) \]

Inserting Eqs. 2.24 - 2.26 into Eq. 2.20 yields the cylindrical stiffness matrix for a ring element

\[
K = \begin{bmatrix}
g_1 & g_2 & 0 & g_1 & -g_2 & 0 \\
g_6 & 0 & g_2 & -g_6 & 0 \\
0 & 0 & 0 & 0 & 0 \\
g_1 & -g_2 & 0 & 0 & 0 \\
\text{sym.} & 0 & 0 & 0 & 0
\end{bmatrix} \quad (2.27)
\]

where the \( g \) functions are defined in Eq. 2.20.

The third type of element, which is not as frequently encountered as the previous two, is the skewed ring element. This element is the same as a regular ring element except that the two end points do not lie in a plane perpendicular to the radial axis of symmetry. In this case, Eq. 2.26 does not hold. Inserting Eqs. 2.24 and 2.25 into Eq. 2.20 yields the cylindrical stiffness matrix for a skewed ring element.
where the g functions are defined in Eq. 2.20.

2.4 Linear System Model

The element models are then assembled to form the system model after Holzer [5,7] using the code-number technique to add the appropriate entries of the individual element stiffness matrices into the appropriate locations of the system stiffness matrix, K. This then yields the system model

\[ K q = Q \]  \hspace{1cm} (2.29)

where q is the generalized displacement vector and Q is the generalized load vector. Q is specified relative to cylindrical coordinates and, when solved for, q will also be in cylindrical coordinates.

The element forces are then easily calculated. Employing the code-number technique, D for each element is obtained from q. d is then calculated via Eq. 2.3. Finally, f is obtained from Eq. 2.1.

The joint forces and reactions are just as easily calculated. F is obtained via Eq. 2.3, then, using the member incidence information stored in the computer, the total force at each joint is compiled. These joint forces and reactions are also in cylindrical
coordinates.

2.5 Nonlinear Element Model

The geometrically nonlinear truss element model used in this study, Fig. 2.5a, is directly formulated relative to global coordinates using the principle of virtual work after Holzer [6], Holzer et al. [8], Lamma [12], and Vu [15]. Since large displacements are considered, the virtual work equations must be formulated relative to the deformed element configuration.

The virtual work associated with a virtual displacement is the sum of the external and internal work which must vanish at an equilibrium configuration

\[ \delta W = \delta W_e + \delta W_i = 0 \]  

(2.30)

The internal work can be expressed as negative the change in internal strain energy, U, which yields, from Eq. 2.30

\[ \delta W = \delta W_e - \delta U = 0 \]  

(2.31)

Referring to Fig. 1.5b, the element displacement can be broken into two components—a rigid body translation and a rotation with axial deflection about the a-end. Since, for a system in equilibrium, no work is required to accomplish a rigid body translation, the generalized displacement vector can be taken as \( u \) where

\[ u = D_b - D_a \]  

(2.32)

and the virtual displacements need only consist of variations at
Figure 2.5: Nonlinear Truss Element
the b-end, $\delta u_i$, $i = 1, 2, 3$. The external work can therefore be expressed as

$$\delta W_e = \sum_{i=1}^{3} P_i \delta u_i$$

(2.33)

and the variation in internal strain energy as

$$\delta U = \sum_{i=1}^{3} \frac{\partial U}{\partial u_i} \delta u_i$$

(2.34)

where $P_i$ is the force in the $i$-direction at the b-end required to equilibrate the deformed state. The total virtual work associated with a virtual displacement can then be expressed as

$$\delta W = \sum_{i=1}^{3} \left( P_i - \frac{\partial U}{\partial u_i} \right) \delta u_i = 0$$

(2.35)

Since virtual displacements are arbitrary, for Eq. 2.35 to be true, the quantity in parentheses must equal zero for all $i$, yielding

$$P_i = \frac{\partial U}{\partial u_i}, \quad i = 1, 2, 3$$

(2.36)

To trace a nonlinear equilibrium path requires that, given one equilibrium point, a load-displacement increment be taken yielding an approximation to the next equilibrium point. An iterative procedure is then used to resolve the force imbalances arising from the approximation and to converge on a new equilibrium point. The procedure is then repeated until a desired portion of the equilibrium
has been traced. To formulate the incremental load-displacement relation, a Taylor series expansion of \( P_i \) at a displacement \( u + \Delta u \) is taken, yielding

\[
P_i(u + \Delta u) = P_i(u) + \sum_{j=1}^{3} \frac{\partial P_i}{\partial u_j} \Delta u_j + \ldots
\]  

(2.37)

Rearranging terms and neglecting those with higher order derivatives yields an approximate expression for the incremental force, \( \Delta P_i \), associated with an incremental displacement, \( \Delta u \),

\[
\Delta P_i = P_i(u + \Delta u) - P_i(u) = \sum_{j=1}^{3} \frac{\partial P_i}{\partial u_j} \Delta u_j
\]  

(2.38)

Inserting Eq. 2.36 into Eq. 2.38 yields

\[
\Delta P_i = \sum_{j=1}^{3} \frac{\partial^2 u}{\partial u_i \partial u_j} \Delta u_j , \quad i = 1, 2, 3
\]  

(2.39)

Eq. 2.39 can be expressed in matrix form as

\[
\Delta P = \mathbf{K} \Delta u
\]  

(2.40)

where

\[
\Delta P = \begin{bmatrix} \Delta P_1 \\ \Delta P_2 \\ \Delta P_3 \end{bmatrix}
\]  

(2.40a)
Formulating an expression for $U$ from the geometric and material properties of the element and assuming that Hooke's Law applies yields the terms of $\hat{K}$ for a general element:

$$\hat{K} = \begin{bmatrix} h_1 & h_2 & h_4 \\ h_2 & h_3 & h_5 \\ h_4 & h_5 & h_6 \end{bmatrix}$$

where

$$h_1 = \gamma [c_1^* \frac{e}{L} (1 - c_1^*)]$$
$$h_2 = \gamma [c_1^* c_2^* (1 - \frac{e}{L*})]$$
$$h_3 = \gamma [c_2^* \frac{e}{L} (1 - c_2^*)]$$
$$h_4 = \gamma [c_1^* c_3^* (1 - \frac{e}{L*})]$$
$$h_5 = \gamma [c_2^* c_3^* (1 - \frac{e}{L*})]$$
$$h_6 = \gamma [c_3^* \frac{e}{L} (1 - c_3^*)]$$

$$\gamma = \frac{AE}{L}$$

$L = $ undeformed element length
L* = deformed element length

e = L* - L = element elongation

\( c_i^* = \frac{L_i^*}{L_i} \), \( i = 1,2,3 \)

= direction cosine of the element local l-axis in the deformed state relative to the global i-axis

L_i* = component of the deformed element length vector in the global i-direction

To formulate the incremental force-displacement relation for the entire element, the transformation matrix, C, is defined as

\[ C = [-I \ I] \quad (2.42) \]

where I is the identity matrix of order 3. The relationship between \( \Delta u \) and \( \Delta D \) (an incremental version of Eq. 2.32) may then be expressed as

\[ \Delta u = C \Delta D \quad (2.43) \]

where

\[ \Delta D = \begin{bmatrix} \Delta D_a \\ \Delta D_b \end{bmatrix} \quad (2.43a) \]

To preserve equilibrium of the deformed element, the relationship between \( \Delta P \) and the incremental total element force vector, \( \Delta F \), may be expressed as

\[ \Delta F = C^T \Delta P \quad (2.44) \]
Inserting Eqs. 2.43 and 2.44 into Eq. 2.40 yields

\[ \Delta F = K \Delta D \]  \hspace{1cm} (2.45)

where

\[ K = C^T K_C = \begin{bmatrix} \hat{K} & -\hat{K} \\ -\hat{K} & \hat{K} \end{bmatrix} \]  \hspace{1cm} (2.45a)

\( K \) is called the \textit{global element tangent stiffness matrix} and will be used to trace the geometrically nonlinear equilibrium path of space trusses.

Since the element tangent stiffness matrix was formulated directly relative to global coordinates, the coordinate transformation used in the nonlinear model must be from global to cylindrical coordinates rather than from local to cylindrical as was used in the formulation of the linear model. To accomplish this transformation, the transformation matrix, \( \Lambda \), is defined such that

\[ \Delta D = \Lambda \Delta D \]  \hspace{1cm} (2.46)

\[ \Delta F = \Lambda^T \Delta F \]

where \( \Delta D \) and \( \Delta F \) are the element nodal displacement and force vectors relative to cylindrical coordinates. Note that \( \Lambda \) in this case differs from that used in the formulation of the linear model. \( \Lambda \) can be partitioned thus
where $\lambda$ and $\hat{\lambda}$ perform the transformation between global and cylindrical coordinates at the a- and b-ends of the element respectively. $\lambda$ and $\hat{\lambda}$ are defined as

$$
\Lambda = \begin{bmatrix}
\lambda & 0 \\
0 & \hat{\lambda}
\end{bmatrix} \quad (2.47)
$$

$$
\lambda = \begin{bmatrix}
c_1 & c_2 & 0 \\
-c_2 & c_1 & 0 \\
0 & 0 & 1
\end{bmatrix}
$$

$$
\hat{\lambda} = \begin{bmatrix}
\hat{c}_1 & \hat{c}_2 & 0 \\
-c_2 & c_1 & 0 \\
0 & 0 & 1
\end{bmatrix} \quad (2.48)
$$

where $c_i$ and $\hat{c}_i$ are the direction cosines of the joint 1-axis at the a- and b-ends of the element, respectively, relative to the global i-axis.

Referring to Fig. 2.6, which is the projection of an arbitrary joint coordinate system onto the global 1,2-plane, expressions for $c_i$ and $\hat{c}_i$, $i = 1,2$, are easily formulated. The formulation which follows assumes that the joint coordinate system shown lies at the a-end of the element being considered. By definition

$$
c_1 = \cos \phi_1 = \frac{X_1}{R} \quad (2.49)
$$

$$
c_2 = \cos \phi_2 = \frac{X_2}{R}
$$
Figure 2.6: Nonlinear Model Coordinate Transformations
where

\[ R = \left[ X_1^2 + X_2^2 \right]^{1/2} \tag{2.49a} \]

When the joint coordinate system lies at the b-end, the formulation is the same except that \( \hat{c} \) is substituted for \( c \) in Eq. 2.49. When the direction cosines are actually calculated in the computer, even though more than one element may be incident at a particular joint, the calculation is done only once for each joint and the general direction cosines are stored in vectors of dimension equal to the number of joints. These cosines are then accessed through the member incidence information stored in the computer as either \( c \) or \( \hat{c} \) and used in the individual element coordinate transformations.

Inserting Eqs. 2.46 into Eq. 2.45 yields the incremental force-displacement relation relative to cylindrical coordinates

\[ \Delta F = K \Delta D \tag{2.50} \]

where \( \Delta F \) and \( \Delta D \) are the cylindrical incremental element force and displacement vectors respectively and

\[ K = \Lambda K A^T \tag{2.50a} \]

K is the cylindrical element tangent stiffness matrix. Incorporating Eqs. 2.41, 2.45a, 2.47, and 2.48 into Eq. 2.50a yields
$K = \begin{bmatrix}
H_1 & H_2 & H_3 & H_4 & H_5 & -H_3 \\
H_6 & H_7 & H_8 & H_9 & -H_7 \\
H_{10} & H_{11} & H_{12} & -H_{10} \\
H_{13} & H_{14} & -H_{11} \\
sym. & & & & & H_{10} 
\end{bmatrix}$  \hspace{1cm} (2.51)

where

\begin{align*}
H_1 &= c_1^2 h_1 + 2c_1 c_2 h_2 + c_2^2 h_3 \\
H_2 &= -c_1 c_2 h_1 + (c_1^2 - c_2^2) h_2 + c_1 c_2 h_3 \\
H_3 &= c_1 h_4 + c_2 h_5 \\
H_4 &= -c_1 \hat{c}_1 h_1 - (c_1 \hat{c}_2 + c_2 \hat{c}_1) h_2 - c_2 \hat{c}_2 h_3 \\
H_5 &= c_1 \hat{c}_2 h_1 - (c_1 \hat{c}_1 - c_2 \hat{c}_2) h_2 - c_2 \hat{c}_1 h_3 \\
H_6 &= c_2^2 h_1 - 2c_1 c_2 h_2 + c_1^2 h_3 \\
H_7 &= -c_2 h_4 + c_1 h_5 \\
H_8 &= c_2 \hat{c}_1 h_1 + (c_2 \hat{c}_2 - c_1 \hat{c}_1) h_2 - c_1 \hat{c}_2 h_3 \\
H_9 &= -c_2 \hat{c}_2 h_1 + (c_2 \hat{c}_1 + c_1 \hat{c}_2) h_2 - c_1 \hat{c}_1 h_3 \\
H_{10} &= h_6 \\
H_{11} &= -c_1 h_4 - \hat{c}_2 h_5 \\
H_{12} &= \hat{c}_2 h_4 - \hat{c}_1 h_5
\end{align*}
\[ H_{13} = \hat{c}_1 c_1 h_1 + 2 \hat{c}_1 \hat{c}_2 h_2 + \hat{c}_2 c_2 h_3 \]
\[ H_{14} = -\hat{c}_1 c_1 h_1 + (\hat{c}_1^2 - \hat{c}_2^2) h_2 + \hat{c}_1 \hat{c}_2 h_3 \]
\[ H_{15} = \hat{c}_2 c_1 h_1 - 2 \hat{c}_1 c_2 h_2 + \hat{c}_1 c_2 h_3 \]

and \( h_i, i = 1, 2, \ldots, 6, \) are defined in Eq. 2.41.

2.6 Nonlinear System Model

The cylindrical system tangent stiffness matrix is assembled from the individual cylindrical element tangent stiffness matrices using the code-number technique after Holzer [5,7]. A series of points on the nonlinear equilibrium path can then be generated after Holzer et al. [8] and Vu [15] using one of several methods. These methods are 1) the Newton/Raphson method, 2) the Riks/Wempner method, and 3) the modified Riks/Wempner method. Although the program listed in Appendix C is capable of utilizing all three of these methods, since the modified Riks/Wempner method without updating has been shown to be the most efficient scheme (see Holzer et al. [8] and Vu [15]), it has been used exclusively in this study.

2.7 Stability of Equilibrium

For a conservative n-degree-of-freedom system, the total potential energy function can be defined after Holzer et al. [8] as

\[ V(q, \lambda) = \Omega(q, \lambda) + U(q) \quad (2.52) \]
where $q$ is the generalized displacement vector, $\lambda$ is the loading parameter, $\Omega$ is the potential energy of the external loads, and $U$ is the internal strain energy. If the loads are configuration independent, the stability of equilibrium can be determined from the second variation of $V$. Specifically

$$\delta^2 V(q, \lambda) = \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\partial^2 U}{\partial q_i \partial q_j} \delta q_i \delta q_j$$

(2.53)

which can be expressed in matrix form as

$$\delta^2 V = \delta q^T K \delta q$$

(2.54)

where $K$ is the cylindrical system tangent stiffness matrix. Equilibrium is stable when $\delta^2 V > 0$, critical when $\delta^2 V = 0$, and unstable when $\delta^2 V < 0$. These three conditions correspond to $K$ being positive definite, positive semidefinite, and indefinite, negative semidefinite, or negative definite respectively. Correspondingly, equilibrium is stable when all eigenvalues of $K$ are positive, critical when at least one eigenvalue is zero, and unstable when at least one eigenvalue is negative. A critical point can be either a bifurcation point or a limit point. When, at a critical point, more than one eigenvalue vanishes, the critical point is called "coincident". Otherwise, it is called "simple".

2.8 Solution of Systems of Linear Algebraic Equations

If a symmetric matrix, $K$, is positive definite, there exists a unique factorization (see Vu [15])
where \( \mathbf{L} \) is a unit lower triangular matrix and \( \mathbf{D} \) is a positive diagonal matrix. The half-band width of \( \mathbf{L} \) is the same as the half-band width of \( \mathbf{K} \), therefore, no additional storage is required to perform this decomposition. A system of linear algebraic equations of the form

\[
\mathbf{Kq} = \mathbf{Q}
\]

(2.56)
can then be solved using a three-step procedure as follows:

1. Solve \( \mathbf{Lx} = \mathbf{Q} \) for \( \mathbf{x} \)
2. Solve \( \mathbf{Dy} = \mathbf{x} \) for \( \mathbf{y} \)
3. Solve \( \mathbf{L}^T\mathbf{q} = \mathbf{y} \) for \( \mathbf{q} \)

Note that all three steps can be executed with very little computational effort and that the \( \text{LDL}^T \) decomposition is not destroyed in the process, allowing later solution of subsequent systems of equations having the same coefficient matrix. This is especially useful in the solution of linear problems where the results of several loading conditions are desired and in the solution of nonlinear problems where the tangent stiffness matrix is not updated with each iteration.

When tracing a nonlinear equilibrium path into the unstable region, however, the tangent stiffness matrix is not always positive definite. In this case, the \( \text{LDL}^T \) decomposition discussed previously may not exist or may be numerically unstable. Therefore, an unconditionally stable decomposition of \( \mathbf{K} \) is required. In this study,
the LINPACK equation solver \cite{3} for symmetric indefinite matrices was used. This solver decomposes $K$ to the form

$$K = UDU^T$$

(2.57)

where $D$ is a block diagonal matrix with blocks of order 1 or 2 and $U$ is the product of elementary unit upper triangular and permutation matrices. Regretfully, LINPACK does not have a routine capable of exploiting both bandedness and symmetry of an indefinite matrix simultaneously. Therefore, with larger problems, the equation solver used with the linear model was also used with the nonlinear model unless difficulties arose which could be attributed to instability of the $LDLT$ decomposition.
CHAPTER 3
DISCUSSION OF RESULTS

Linear and nonlinear analyses were performed on each of two trusses. The results of the nonlinear analysis were compared to those in refs. 12 and 15. The results of the linear analysis were compared to those obtained using a program previously written and verified by the author. This earlier program operated solely in global coordinates and has been incorporated into the linear analysis program as an option.

When analyzing "pie slices" using either the linear or nonlinear program, the member cross sections and joint loads were altered, where necessary, per sec. 2.2.

3.1 Linear Analysis

The computer program used in the linear analysis was written by the author and is capable of operating in either global or cylindrical coordinates. A program listing and users' guide can be found in Appendix B.

3.1.1 Structure 1

The first structure analyzed was a 24-bar lamella dome (see Fig. 3.1). All element cross-sectional areas were taken as 1.0 cm$^2$ and the modulus of elasticity as 2,110,000 g/cm$^2$. The loading condition consisted of a 1000 g downward load at each free joint.
Figure 3.1: Structure 1
The structure and loading condition exhibit \( C_6 \) radial symmetry and, therefore, five "pie slices" equal to \( \frac{1}{6}, \frac{2}{6}, ..., \frac{5}{6} \) of the structure (see Fig. 3.2) were analyzed.

All "pie slice" analyses yielded results identical to those obtained from analysis of the entire structure.

3.1.2 Structure 2

The second structure analyzed was a 70-bar lamella dome with 70 degrees of freedom (see Fig. 3.3). All joints lie on a spherical cap of 1200 in. radius with the first ring 10° from the radial axis of symmetry and the second ring 20°. All element cross-sectional areas were taken as 1.0 in.\(^2\) and the modulus of elasticity as 30,000 ksi. All joints on the outer ring were constrained in the 3-direction, joint 16 was constrained in the 1- and 2-directions, and joint 31 was constrained in the 2-direction. The loading condition consisted of downward 1.0 kip loads at joints 8, 9, 12, 13, 15, 16, 17, 19, 20, 23, and 24.

The structure and loading condition exhibit \( C_{10} \) radial symmetry and, therefore, nine "pie slices" equal to \( \frac{1}{10}, \frac{2}{10}, ..., \frac{9}{10} \) of the structure were analyzed.

As with the 24-bar dome, all "pie slice" analyses yielded results identical to those obtained from analysis of the entire structure.

3.1.3 Comparison of Effort

A comparison of the relative effort required to analyze a "pie slice" of a structure vs. the effort required to analyze the
Figure 3.2: "Pie Slices" of Structure 1
Figure 3.3: Structure 2
entire structure was made for structures 1 and 2. Acknowledging that certain FORTRAN statements require more execution time than others, the number of statements executed in performing an analysis was, nevertheless, used as the criterion for comparison of computational effort. For a structure exhibiting $C_n$ symmetry, a "pie slice", $n$ of which would form the whole structure, was considered to represent $\frac{1}{n} \times 100$ percent of the structure. Consequently, $m$ of these "pie slices" combined to form a larger "pie slice" were considered to represent $\frac{m}{n} \times 100$ percent of the structure. Since, when analyzing a "pie slice", the joint lying on the radial axis of symmetry must be constrained in the 1- and 2-directions, for purposes of comparison of effort, this joint was likewise constrained when analyzing the entire structure.

The number of statements executed for the various portions of structures 1 and 2 are displayed in Table 3.1. The percent of computational effort is defined as the ratio of the number of statements executed in the analysis of a "pie slice" to the number executed in the analysis of the entire structure times one hundred. The half-band width and the number of degrees of freedom (D.O.F.) are included for reference. A "wave front" scheme [5] was used in numbering the joints. When analyzing a "pie slice", the "wave front" was taken to move in the direction of the axis of bisymmetry.

Fig. 3.4 is a plot of the data in Table 3.1. As can be seen, for the two structures analyzed, the percent reduction in the amount of effort required was roughly equal to the percent reduction in the size of the structure.
### TABLE 3.1
COMPUTATIONAL EFFORT REQUIRED IN LINEAR ANALYSIS

<table>
<thead>
<tr>
<th>% of Structure</th>
<th>No. of D.O.F.</th>
<th>Half-Band Width</th>
<th>Statements Executed</th>
<th>% of Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>19</td>
<td>10</td>
<td>9965</td>
<td>100.0</td>
</tr>
<tr>
<td>83.3</td>
<td>17</td>
<td>10</td>
<td>8541</td>
<td>85.7</td>
</tr>
<tr>
<td>66.7</td>
<td>14</td>
<td>10</td>
<td>6684</td>
<td>67.1</td>
</tr>
<tr>
<td>50</td>
<td>11</td>
<td>9</td>
<td>4842</td>
<td>48.6</td>
</tr>
<tr>
<td>33.3</td>
<td>8</td>
<td>8</td>
<td>3380</td>
<td>33.9</td>
</tr>
<tr>
<td>16.7</td>
<td>5</td>
<td>5</td>
<td>1641</td>
<td>16.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>% of Structure</th>
<th>No. of D.O.F.</th>
<th>Half-Band Width</th>
<th>Statements Executed</th>
<th>% of Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>70</td>
<td>22</td>
<td>64211</td>
<td>100.0</td>
</tr>
<tr>
<td>90</td>
<td>65</td>
<td>22</td>
<td>59207</td>
<td>92.2</td>
</tr>
<tr>
<td>80</td>
<td>58</td>
<td>22</td>
<td>51409</td>
<td>80.1</td>
</tr>
<tr>
<td>70</td>
<td>51</td>
<td>22</td>
<td>44681</td>
<td>69.6</td>
</tr>
<tr>
<td>60</td>
<td>44</td>
<td>22</td>
<td>37792</td>
<td>58.9</td>
</tr>
<tr>
<td>50</td>
<td>37</td>
<td>22</td>
<td>31637</td>
<td>49.3</td>
</tr>
<tr>
<td>40</td>
<td>30</td>
<td>19</td>
<td>22295</td>
<td>34.7</td>
</tr>
<tr>
<td>30</td>
<td>23</td>
<td>18</td>
<td>15582</td>
<td>24.3</td>
</tr>
<tr>
<td>20</td>
<td>16</td>
<td>11</td>
<td>7694</td>
<td>12.0</td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>8</td>
<td>3484</td>
<td>5.4</td>
</tr>
</tbody>
</table>
Figure 3.4: Percent of Effort vs. Percent of Structure
3.2 Nonlinear Analysis

The computer program used in the nonlinear analysis is an extension by the author of a program written by Lamma [12]. The program is capable of tracing the nonlinear fundamental equilibrium path of an arbitrary truss structure with arbitrary loads, converging on bifurcation and limit points, and, in many cases, branching off of the fundamental equilibrium path onto a bifurcation path. These varying tasks may be performed in either global or cylindrical coordinates. Either of two equation solvers may be used in tracing equilibrium paths. The first is a very economical solver after Cook [2] which exploits both bandedness and symmetry of the tangent stiffness matrix, and the second is the less efficient LINPACK solution routine for symmetric indefinite systems of equations. A program listing and users' guide can be found in Appendix C.

3.2.1 Structure 1

The first structure analyzed was the 19 degree of freedom, 24-bar lamella dome analyzed linearly in sec. 3.1.1 (see Fig. 3.1). This structure was chosen because it has been extensively analyzed in previous papers (see refs. 12 and 15) and a well-established basis for comparison of results exists.

All members have the same cross-sectional area and modulus of elasticity. The loading condition, Q, is expressed as

\[ Q = \lambda \tilde{Q} \]  

(3.1)

where \( \tilde{Q} \) is the generalized loading vector and \( \lambda \) is a scaling factor.
When incrementing the load while tracing the equilibrium path, \( \bar{Q} \) is held constant while \( \lambda \) is incremented. Load magnitudes for both structures analyzed in this study are expressed nondimensionally in terms of \( \bar{\lambda} \) where

\[
\bar{\lambda} = \frac{\lambda \times 10^4}{AE}
\]  

(3.2)

The generalized loading condition consisted of downward unit loads at joints 3, 4, 5, 9, 10, and 11.

Tracing the fundamental nonlinear equilibrium path of the entire structure located three bifurcation points before reaching the first limit point. These three points occurred at values of \( \bar{\lambda} \) equal to 7.592, 9.359, and 15.520 which are verified within a reasonable computational error by ref. 12 (see Fig 3.5). Branching onto the bifurcation paths at the first and second bifurcation points revealed that the buckling mode exhibited \( C_3 \) radial symmetry at the first bifurcation point and \( C_2 \) radial symmetry at the second. The program was unable to converge on a bifurcation path at the third bifurcation point. The buckling configuration at the first bifurcation point consisted of a buckling of the compression ring with joints 3, 9, and 10 moving together as one group, and joints 4, 5, and 11 as another. This differs from the buckling mode described by Lamma [12] for this bifurcation point. The buckling configuration at the second bifurcation point consisted of a "tilting" of the compression ring around the global 1-axis.

Tracing the fundamental equilibrium path of a "pie slice" equal to one sixth of structure 1 located only one bifurcation
Figure 3.5: Nonlinear load vs. vertical displacement of compression ring curve for structure 1 showing locations of bifurcation points.
point before reaching the limit point. This bifurcation point duplicates the first bifurcation point found by analysis of the entire structure. Branching onto the bifurcation path of the "pie slice" also duplicated the buckling mode revealed by analysis of the whole structure, with the single ring element "tilting" to one side. When extrapolating this buckling configuration around the entire structure, alternating "pie slices" tip in opposite directions to produce the overall "zig-zag" buckled configuration for the compression ring (see Fig. 3.6).

Tracing the fundamental equilibrium path of a "pie slice" equal to two sixths of the structure located two bifurcation points before reaching the limit point. The first bifurcation point, as expected, again duplicates the first bifurcation point found by analysis of the entire structure. The second bifurcation point, however, occurred at $\lambda = 11.800$ where none had been encountered in the two previous analyses. Although the program was unable to converge on the associated bifurcation path, the author strongly suspects that the buckling configuration consists of the entire "pie slice" tipping to one side. Since an odd number of these "pie slices" would be required to form the entire structure, this buckling mode would, therefore, be inhibited in an analysis of the entire structure.

A comparison of the computational effort required to analyze "pie slices" vs. the effort to analyze the entire structure was also made with the nonlinear model. However, due to high computing costs and difficulties which arose in the analysis of structure 2,
Figure 3.6: Buckled Configuration of Structure 1
the comparison was made only with the 24-bar dome. The fundamental equilibrium paths of the entire structure and of five "pie slices" equal to \( \frac{1}{6}, \frac{2}{6}, \ldots, \frac{5}{6} \) of the structure were traced until the first bifurcation point was passed at \( \bar{\lambda} = 7.592 \). The initial load increment, \( \Delta \bar{\lambda}_o \), was set at 0.800 and the tangent vector was scaled per refs. 8, 12, and 15 for four iterations. The determinant of the undeformed tangent stiffness matrix was provided before execution began. The results of this comparison are displayed in Table 3.2 and Fig. 3.7. The LINPACK equation solver was used to trace the equilibrium paths, so the half-band widths and the joint numbering scheme are of no consequence. Again, the percent reduction in the computational effort required was roughly equal to the percent reduction in the size of the structure. Note, however, that the data points may suggest a slightly parabolic relation between effort and size of structure. This is probably because, in a nonlinear analysis, a greater portion of the total computational effort is expended in solving systems of simultaneous equations than is likewise expended in a linear analysis, and that there is generally a second order relation between the size of a system of equations and the number of operations required to solve it [10].

3.2.2 Structure 2

The second structure analyzed was the 70 degree of freedom, 70-bar lamella dome analyzed in sec. 3.1.2 (see Fig. 3.3). All members have the same cross-sectional area and modulus of elasticity. The loading condition, \( Q \), is as expressed in Eq. 3.1 and load mag-
TABLE 3.2

COMPUTATIONAL EFFORT REQUIRED IN NONLINEAR ANALYSIS

<table>
<thead>
<tr>
<th>% of Structure</th>
<th>No. of D.O.F.</th>
<th>Statements Executed</th>
<th>% of Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>19</td>
<td>431684</td>
<td>100.0</td>
</tr>
<tr>
<td>83.3</td>
<td>17</td>
<td>353177</td>
<td>81.8</td>
</tr>
<tr>
<td>66.7</td>
<td>14</td>
<td>275593</td>
<td>63.8</td>
</tr>
<tr>
<td>50</td>
<td>11</td>
<td>198041</td>
<td>45.9</td>
</tr>
<tr>
<td>33.3</td>
<td>8</td>
<td>128609</td>
<td>29.8</td>
</tr>
<tr>
<td>16.7</td>
<td>5</td>
<td>69633</td>
<td>16.1</td>
</tr>
</tbody>
</table>
Figure 3.7: Percent of Effort vs. Percent of Structure
nitudes are again expressed in terms of $\tilde{\lambda}$, which is defined in Eq. 3.2.

The generalized loading condition consisted of downward unit loads at joints 8, 9, 12, 13, 15, 17, 19, 20, 23, and 24.

Tracing the fundamental equilibrium path of the entire structure encountered five bifurcation points before reaching the first limit point at $\tilde{\lambda} = 6.686$. The first bifurcation point occurred at $\tilde{\lambda} = 2.932$. Due to problems with the solution algorithm in converging on the remaining bifurcation points and to the high computing costs associated with a structure of this size, the remaining bifurcation were not determined to a high degree of accuracy.

Analyzing a "pie slice" equal to one tenth of the structure yielded no bifurcation points before reaching the first limit point.

Analyzing a "pie slice" equal to two tenths of the structure (see Fig. 3.8) yielded two bifurcation points before reaching the first limit point. The first bifurcation point duplicated the first bifurcation point found by analysis of the entire structure. The second occurred at a value of $\tilde{\lambda} \approx 4.000$ where none had been previously encountered. Again, the author strongly suspects that the associated buckling mode consists of the entire "pie slice" tipping to one side, which would be impossible in an analysis of the entire structure.

The program was unable to converge on the bifurcation path at the first bifurcation point. However, suspecting that, as with the 24-bar dome, the associated buckling mode consisted of a buckling of the compression ring in a "zig-zag" fashion, a $0.001\%$
Figure 3.8: "Pie Slice" of Structure 2
imperfection in the vertical coordinate of joint 6 was introduced (actually, since the joint coordinates were entered in spherical coordinates, joint 6 was placed on a spherical cap of radius 1199.988 in. The resulting imperfection in the radial coordinate was negligible). Analysis of this imperfect slice yielded a limit point at $\bar{\lambda} = 2.884$, thus confirming the suspicion. However, the author is unable to offer any explanation as to why the one-tenth "pie slice" did not model this buckling mode as did the one-sixth "pie slice" with the 24-bar dome.

Since no single bifurcation point was modelled by all "pie slices" of the structure, a comparison of effort was not made with the 70-bar dome.
CHAPTER 4

CONCLUSION AND RECOMMENDATIONS

In this study, the element and system models for linear and geometrically nonlinear three-dimensional trusses were developed and implemented relative to cylindrical coordinates. These models were each used in the analysis of two radially symmetric domes with radially symmetric loading conditions.

The most important benefits derived from using these models are as follows:

1. Analysis of "pie slices" of a structure required considerably less effort than did analysis of the entire structure in both computer time and in operator time to develop data files.

2. Noting which bifurcation points were inhibited by which "pie slice" configurations provided qualitative information concerning buckled configurations when "branching" methods failed, and provided guidance in introducing imperfections to verify suspected buckling modes.

The primary drawback to using these models was that if a buckling mode cannot be modelled by a particular "pie slice" configuration, the bifurcation point corresponding to that mode will be suppressed in the "pie slice" analysis. Therefore, at least a rough analysis of the entire structure is usually required to insure
that all of the bifurcation points have been determined and to permit identification of extraneous bifurcation points introduced by the "pie slice".

The author would like to offer several recommendations concerning the further application of the structural models developed in this study.

1. Investigate further the suppression of bifurcation points by "pie slice" configurations. Primarily, try to offer some explanation for why some buckling modes which exhibit $C_{2n}$ radial symmetry can be modelled with a "pie slice" equal to one $n^{th}$ of the structure (as with structure 1) and others require a "pie slice" equal to one $2n^{th}$ of the structure (as with structure 2).

2. Investigate the possibility of expressing arbitrary loads in terms of combinations of radially symmetric loads. In a nonlinear analysis, determine the effect this would have on the locations of bifurcation and limit points. If, indeed, such an expression is possible, investigate the economics of doing so.

The author would also like to offer several suggestions concerning further use of the nonlinear solution technique regardless of which model formulation we employed.

1. A frequent problem encountered was that, when using the modified Riks/Wempner method, while solving the resulting quadratic equation (see refs. 8 and 15) the value under the radical would become negative, resulting in an error. The actual physical meaning of this should be investigated
so that ways to avoid the problem might be suggested. This would make the modified Riks/Wempner method more consistently reliable.

2. Locate or write a solution routine for banded symmetric indefinite matrices with accompanying routines for calculating the determinant and number of negative eigenvalues of the system tangent stiffness matrix. Implement and investigate this routine.

3. The present method for converging on bifurcation points (see ref. 12) consists of noting, via the number of negative eigenvalues, when a bifurcation point has been passed and then reducing the length and changing the sign of the tangent vector until the bifurcation point has been located with sufficient accuracy. However, reducing the length of a vector in n + 1 dimensional space (where n is the number of D.O.F.) does not insure that the resulting load magnitude does not fall in a region of the equilibrium path that has already been determined not to contain the bifurcation point. Therefore, the method often does not converge rapidly and, occasionally, with very tight convergence tolerances, for all practical purposes, does not converge at all. The author, instead, suggests that the Newton/Raphson method (see ref. 6) might perform better in converging on bifurcation points. Decreasing the size of $\Delta \lambda$ rather than $\Delta s$ systematically would provide absolute control of one of the parameters, thus insuring that the
method will converge on the bifurcation point directly and to any desired degree of accuracy. Although less computationally efficient *per iteration*, the author believes that the Newton/Raphson method would provide overall greater efficiency in locating bifurcation points.

4. More often than not, the present branching technique is not capable of converging on a bifurcation path. More research should be conducted in an effort to improve the branching capabilities.
APPENDIX A

REFERENCES


APPENDIX B

LINEAR PROGRAM
THIS PROGRAM PERFORMS A LINEAR STRESS AND DISPLACEMENT ANALYSIS ON SPACE TRUSSES. IT CAN OPERATE IN BOTH GLOBAL AND CYLINDRICAL COORDINATES AND ACCEPTS LOADS AND JOINT CONSTRAINTS IN WHICHEVER COORDINATE SYSTEM IS BEING EMPLOYED. THE CYLINDRICAL COORDINATE SYSTEM IS INTENDED FOR, BUT NOT RESTRICTED TO, RADially SYMMETRIC STRUCTURES AND ALLOWS FOR IMPOSITION OF THE APPROPRIATE BOUNDARY CONDITIONS TO ANALYZE A REPRESENTATIVE "PIE SLICE" OF THE STRUCTURE.

THE GLOBAL COORDINATE AXES ARE A RIGHT-HANDED SYSTEM. WHEN WORKING WITH A RADIAII SYMMETRIC STRUCTURE IN CYLINDRICAL COORDINATES, THE GLOBAL 3-AXIS MUST BE PLACED ALONG THE AXIS OF RADIAL SYMMETRY WITH THE 1-DIRECTION BEING RADIIALLY OUT.

JOINT COORDINATES MAY BE ENTERED IN TERMS OF CARTESIAN (X1, X2, X3) OR SPHERICAL (R, THETA, PHI) COORDINATES. WHEN ENTERED IN SPHERICAL COORDINATES, THE CARTESIAN COORDINATES ARE CALCULATED THUS:

\[
\begin{align*}
X1 &= R \times \cos(\text{THETA}) \times \sin(\text{PHI}) \\
X2 &= R \times \sin(\text{THETA}) \times \sin(\text{PHI}) \\
X3 &= R \times (1. - \cos(\text{PHI}))
\end{align*}
\]

WHEN WORKING IN CYLINDRICAL COORDINATES, FOUR TYPES OF ELEMENTS ARE RECOGNIZED AND MUST BE IDENTIFIED BY THE USER PRIOR TO DATA ENTRY. THESE TYPES ARE:

1) RADIAL ELEMENT—AN ELEMENT WHICH LIES ON A LINE INTERSECTING THE AXIS OF RADIAL SYMMETRY OR WHICH IS PARALLEL TO OR LIES ON THAT AXIS EXCEPT FOR THOSE ELEMENTS WHICH HAVE PRECISELY ONE END POINT ON THE AXIS.

2) RING ELEMENT—AN ELEMENT WHICH HAS BOTH END POINTS EQUIDISTANT FROM AND LYING IN A PLANE PERPENDICULAR TO THE RADIAL AXIS OF SYMMETRY.
3) SKEW RING ELEMENT—AN ELEMENT WHICH HAS BOTH END POINTS EQUIDISTANT FROM BUT NOT LYING IN A PLANE PERPENDICULAR TO THE RADIAL AXIS OF SYMMETRY.

4) SKEW ELEMENT—AN ELEMENT WHICH DOES NOT MEET ANY OF THE CRITERIA ABOVE, INCLUDING OTHERWISE RADIAL ELEMENTS WHICH HAVE PRECISELY ONE END POINT ON THE AXIS OF RADIAL SYMMETRY.

DATA INPUT IS AS FOLLOWS (ALL DATA ARE UNFORMATTED):

1) NUMBER OF ELEMENTS, NUMBER OF JOINTS, NUMBER OF LOADING CONDITIONS, A FLAG INDICATING WHETHER THE JOINT COORDINATES ARE IN SPHERICAL OR CARTESIAN COORDINATES (1 = SPHERICAL, 2 = CARTESIAN), AND A FLAG INDICATING WHETHER THE PROGRAM IS OPERATING IN GLOBAL OR CYLINDRICAL COORDINATES (1 = GLOBAL, 2 = CYLINDRICAL) (5 INTEGER).

2) JOINT COORDINATES FOR EACH JOINT IN TERMS OF GLOBAL OR SPHERICAL COORDINATES. X1, X2, X3; OR R, THETA, PHI. ANGLES ARE IN DEGREES.

   R = RADIUS OF CURVATURE
   THETA = ANGLE FROM THE GLOBAL 1-AXIS
   PHI = ANGLE FROM THE GLOBAL 3-AXIS

3) FOR EACH ELEMENT, THE JOINT NUMBERS AT ITS ENDS (2 INTEGER).

4) WHEN WORKING IN GLOBAL COORDINATES--
   FOR EACH ELEMENT, ITS CROSS SECTIONAL AREA AND THE MODULUS OF ELASTICITY OF ITS MATERIAL (2 REAL).

   WHEN WORKING IN CYLINDRICAL COORDINATES--
   FOR EACH ELEMENT, ITS ID NUMBER, CROSS SECTIONAL AREA, AND THE MOD--
ULUS OF ELASTICITY OF ITS MATERIAL (1 INTEGER, 2 REAL).

ID=1 RADIAL ELEMENT
ID=2 RING ELEMENT
ID=3 SKEW RING ELEMENT
ID=4 SKEW ELEMENT

5) THE JOINT NUMBER AND DIRECTION OF EACH APPLIED JOINT CONSTRAINT
RELATIVE TO WHICHEVER COORDINATE SYSTEM IS BEING EMPLOYED. A DIRECTION OF 4 INDICATES THAT ALL THREE DIRECTIONS ARE CONSTRAINED FOR THAT JOINT (I.E. THE JOINT IS PINNED) (2 INTEGER) AFTER THE LAST CONSTRAINT, ENTER ZEROS.

6) THE JOINT NUMBER, DIRECTION, AND MAGNITUDE OF EACH EXTERNAL JOINT LOAD RELATIVE TO WHICHEVER COORDINATE SYSTEM IS BEING EMPLOYED (2 INTEGER, 1 REAL). AFTER THE LAST JOINT LOAD, ENTER ZEROS.

***************************************************************************
SYNOPSIS OF DATA INPUT
***************************************************************************

1) NE, NJ, NACT, JOINTS, ICOORD (5 INTEGER)

2) FOR EACH JOINT --
   X1, X2, X3 (3 REAL)
OR
   R, THETA, PHI (3 REAL)
   (ALL ANGLES ARE IN DEGREES)

3) FOR EACH ELEMENT -- MINC(I,1), MINC(I,2) (2 INTEGER)
4) FOR EACH ELEMENT --

WHEN WORKING IN GLOBAL COORDINATES --
A, E (2 REAL)

WHEN WORKING IN CYLINDRICAL COORDINATES --
ID, A, E (1 INTEGER, 2 REAL)

5) FOR EACH JOINT CONSTRAINT --

JNUM, JDIR (2 INTEGER)
(AFTER THE LAST CONSTRAINT, ENTER ZEROS)

6) FOR EACH EXTERNAL JOINT LOAD --

JNUM, JDIR, FORCE (2 INTEGER, 1 REAL)
(AFTER THE LAST JOINT FORCE, ENTER ZEROS)

*********************************************************************************************
DEipherS OF OUTPUT PARAMETERS
*********************************************************************************************

AREA: THE CROSS SECTIONAL AREA OF THE MEMBER

C1: THE DIRECTION COSINE OF THE MEMBER RELATIVE TO THE JOINT
1-AXIS AT THE A-END (THE END INCIDENT TO THE JOINT WHOSE
NUMBER LIES IN THE "FROM" COLUMN IN THE MEMBER PROPERITES LIST-
ING)

C2: THE DIRECTION COSINE OF THE MEMBER RELATIVE TO THE JOINT
2-AXIS AT THE A-END
C1H  THE DIRECTION COSINE OF THE MEMBER RELATIVE TO THE JOINT 1-AXIS AT THE B-END (THE END INCIDENT TO THE JOINT WHOSE NUMBER LIES IN THE "TO" COLUMN IN THE MEMBER PROPERTIES LISTING)

C2H  THE DIRECTION COSINE OF THE MEMBER RELATIVE TO THE JOINT 2-AXIS AT THE B-END

CG1  THE DIRECTION COSINE OF THE MEMBER RELATIVE TO THE GLOBAL 1-AXIS

CG2  THE DIRECTION COSINE OF THE MEMBER RELATIVE TO THE GLOBAL 2-AXIS

CG3  THE DIRECTION COSINE OF THE MEMBER RELATIVE TO THE GLOBAL 3-AXIS

E    THE MODULUS OF ELASTICITY OF THE MEMBER MATERIAL

ID   THE ID NUMBER OF THE MEMBER

IHBW THE HALF BAND WIDTH OF THE SYSTEM STIFFNESS MATRIX

JCODE THE LISTING IN ORDER OF JOINT NUMBER OF THE SYSTEM GENERAL DISPLACEMENT NUMBER CORRESPONDING TO THE 1-, 2-, AND 3-DIRECTION IN TERMS OF WHICHEVER COORDINATE SYSTEM IS BEING EMPLOYED. A ZERO ENTRY INDICATES THAT THAT DIRECTION IS CONSTRAINED.

LENGTH THE LENGTH OF THE MEMBER

MCODE THE LISTING IN ORDER OF ELEMENT NUMBER OF THE SYSTEM GENERAL DISPLACEMENT NUMBER CORRESPONDING TO THE 1-, 2-, AND 3-DIR-
SECTION IN TERMS OF WHICHEVER COORDINATE SYSTEM IS BEING EMPLOYED. A ZERO ENTRY INDICATES THAT THAT DIRECTION IS CONSTRAINED.

MINC THE MATRIX CONTAINING THE JOINT NUMBERS AT THE ENDS OF THE ELEMENTS.

NACT NUMBER OF LOADING CONDITIONS

NE NUMBER OF ELEMENTS

NEQS THE NUMBER OF DEGREES OF FREEDOM OF THE ENTIRE STRUCTURE

NJ NUMBER OF JOINTS

Q THE DISPLACEMENT AND LOAD VECTOR—WHEN THE DISPLACEMENTS ARE CALCULATED, THE LOAD VECTOR IS DESTROYED.

SS THE SYSTEM STIFFNESS MATRIX OF THE STRUCTURE

X1 JOINT COORDINATE IN THE 1-DIRECTION RELATIVE TO THE GLOBAL COORDINATE AXES

X2 JOINT COORDINATE IN THE 2-DIRECTION RELATIVE TO THE GLOBAL COORDINATE AXES

X3 JOINT COORDINATE IN THE 3-DIRECTION RELATIVE TO THE GLOBAL COORDINATE AXES

NOTE: FOR RADIAL ELEMENTS (ID=1), C1H=C1 AND C2H=C2=0.

FOR RING ELEMENTS (ID=2), CG3=0., C1H=−C1, AND C2H=C2.

FOR SKEW RING ELEMENTS (ID=3), C1H=−C1 AND C2H=C2.
SUBROUTINE DESCRIPTIONS

ACTION  THIS SUBROUTINE GENERATES THE EXTERNAL FORCE VECTOR FOR THE
         PARTICULAR LOADING CONDITION IN QUESTION.

DATA    THIS SUBROUTINE READS AND STORES ALL INFORMATION CONCERNING
         THE STRUCTURE EXCEPT FOR THE LOADING CONDITIONS. IT ALSO
         CONSTRUCTS JCODE AND PLACES ZEROS IN THE POSITIONS CORRESPONDING
         TO CONSTRAINTS BUT DOES NOT GENERATE THE NON-ZERO TERMS.

FORCE   THIS SUBROUTINE COMPUTES THE ELEMENT AND JOINT FORCES FOR
         THE STRUCTURE.

LOKATE  THIS SUBROUTINE GENERATES THE OUTPUT PARAMETERS WHICH
         DEFINE THE LOCATION OF THE GENERALIZED DISPLACEMENTS.

PROCES  THIS SUBROUTINE COMPLETES THE GENERATION OF JCODE AND GEN-
         ERATES ALL OF THE PARAMETERS REQUIRED TO PERFORM THE ANALYSIS
         BUT WHICH ARE NOT READ DIRECTLY INTO THE COMPUTER.

SOLVE   THIS SUBROUTINE SOLVES THE BANDED SYSTEM OF LINEAR ALGEBRAIC
         EQUATIONS FOR THE GENERAL DISPLACEMENTS.

STIFF   THIS SUBROUTINE GENERATES THE BANDED SYSTEM STIFFNESS MATRIX
         FOR THE ENTIRE STRUCTURE.
C*******************************************************************************
C MAIN PROGRAM
C*******************************************************************************
DIMENSION SS(19,19)
DIMENSION Q(19), LOCATE(19,2)
DIMENSION A(24), E(24), XL(24), GAMMA(24), CGL(24), CG2(24),
CG3(24), C1(24), C2(24), C1H(24), C2H(24), MCODE(24,6),
2MINC(24, 2), ID(24)
DIMENSION P(13,3), X1(13), X2(13), X3(13), JCODE(13,3)

C READ AND WRITE CONTROL PARAMETERS AND FLAGS.
READ(1,*) NE, NJ, NACT, JOINTS, ICOORD
WRITE(2,15) NE, NJ, NACT
15 FORMAT(///'NE=', 13, 4X,'NJ=', 13, 4X,'NACT=', 13://${}
CALL DATA(X1, X2, X3, A, E, JCODE, MINC, ID, ICOORD, JOINTS, NJ, NE)
CALL PROCES(X1, X2, X3, CGL, CG2, CG3, C1, C2, C1H, C2H, GAMMA, XL, MCODE, 
JCODE, MINC, ID, NEQS, NJ, NE, IHBW, A, E, ICOORD)
CALL LOCATE(JCODE, LOCATE, NJ, NEQS)
DO 30 I=1, NEQS  
DO 30 J=1, IHBW  
30 SS(I, J)=0.
CALL STIFF(SS, CGL, CG2, CG3, C1, C2, C1H, C2H, GAMMA, MCODE, ID, NEQS, 
IHBW, ICOORD, NE)
NCOUNT=NACT
100 LCNUN=NACT-NCOUNT+1
WRITE(2,20) LCNUN  
20 FORMAT(///' * * * LOADING CONDITION * * *', 13, 2X,'* * *')
CALL ACTION(Q, JCODE, NEQS, NJ)
CALL SOLVE(SS, Q, LOCATE, NACT, NEQS, IHBW, NCOUNT)
CALL FORCE(CGL, CG2, CG3, C1, C2, C1H, C2H, Q, GAMMA, MCODE, ID, NJ, NE, 
ICOORD, NEQS, P, MINC)
NCOUNT = NCOUNT - 1
IF(NCOUNT.GT.0)GOTO 100
STOP
SUBROUTINE ACTION

SUBROUTINE ACTION(Q,JCODE,NEQS,NJ)
  DIMENSION Q(NEQS),JCODE(NJ,3)
  DO 1 I=1,NEQS
    Q(I)=0.
 1 C

READ AND WRITE THE EXTERNAL JOINT FORCES.

WRITE(2,4)
  4 FORMAT('/// EXTERNAL JOINT FORCES')
WRITE(2,5)
  5 FORMAT('JNUM',3X,'JDIR',6X,'FORCE')

READ(1,*JNUM,JDIR,FORCE

TEST FOR LAST CARD AND PLACE THE EXTERNAL JOINT FORCES IN THE FORCE VECTOR.

IF(JNUM.EQ.0)GOTO 50
  K=JCODE(JNUM,JDIR)
  Q(K)=FORCE
WRITE(2,20)JNUM,JDIR,FORCE
GOTO 45
  20 FORMAT(217,F13.2)
  50 RETURN
SUBROUTINE OATACX1,X2,X3,A,E,JCODE,MINC,1D,ICOORD,JOINTS,NJ,NE)
DIMENSION X1(NJ),X2(NJ),X3(NJ),A(NE),E(NE),JCODE(NJ,3),MINC(NE,2),
1D(NE)
GOTO(21,22),ICOORD
21 WRITE(2,23)
GOTO 25
22 WRITE(2,24)
23 FORMAT(' THE DISPLACEMENTS AND JOINT FORCES ARE COMPUTED RELATIVE
1TO GLOBAL COORDINATES.//)
24 FORMAT(' THE DISPLACEMENTS AND JOINT FORCES ARE COMPUTED RELATIVE
1TO CYLINDRICAL/,,' COORDINATES.//)
READ AND WRITE JOINT COORDINATES.
WRITE(2,20)
20 FORMAT(' JOINT COORDINATES')
WRITE(2,100)
100 FORMAT(' JOINT ',5X,'X1',8X,'X2',8X,'X3')
GOTO(1,2),JOINTS
1 DO 41 I=1,NJ
READ(1,*)R,THETA,PHI
THETA =THETA*.01745329
PHI=PHI*.01745329
X1(I)=R*COS(THETA)*SIN(PHI)
X2(I)=R*SIN(THETA)*SIN(PHI)
X3(I)=R*(1.-COS(PHI))
41 WRITE(2,101)I,X1(I),X2(I),X3(I)
GOTO 90
2 DO 40 I=1,NJ
READ(1,*)X1(I),X2(I),X3(I)
40 WRITE(2,101)I,X1(I),X2(I),X3(I)
101 FORMAT(I4,2X,3F10.4)

C READ MEMBER INCIDENCE MATRIX.

C

C

C

C

90 DO 50 I=1,NE
READ(1,*)MINC(I,1),MINC(I,2)
IF(MINC(I,2).GT.MINC(I,1))GOTO 50
TEMP=MINC(I,2)
MINC(I,2)=MINC(I,1)
MINC(I,1)=TEMP
50 CONTINUE

C READ AND WRITE MEMBER AND MATERIAL PROPERTIES.

C

C

C

C

WRITE(2,18)
18 FORMAT(/'' MEMBER PROPERTIES'')
GOTO(61,62),ICOORD
61 WRITE(2,30)
30 FORMAT(4X,'MEMBER',5X,'AREA',10X,'E',7X,'FROM',2X,'TO')
   DO 63 I=1,NE
      READ(1,*)A(I),E(I)
   63 WRITE(2,103)I,A(I),E(I),MINC(I,1),MINC(I,2)
103 FORMAT(I8,F12.4,F14.4,3X,13,2X,13)
GOTO 65
62 WRITE(2,19)
19 FORMAT(4X,'MEMBER',3X,'ID',7X,'AREA',10X,'E',7X,'FROM',2X,'TO')
   DO 60 I=1,NE
      READ(1,*)ID(I),A(I),E(I)
   60 WRITE(2,102)I,ID(I),A(I),E(I),MINC(I,1),MINC(I,2)
102 FORMAT(I8,I7,F12.4,F14.4,3X,13,2X,13)
CONSTRUCT JCODE

INITIALIZE JCODE TO UNITY

DO 5 I=1,NJ
   DO 5 J=1,3
      JCODE(I,J)=1
  5

TEST FOR LAST CARD AND PLACE ZEROS FOR CONSTRAINTS.

READ(1,*) JNUM,JDIR
IF(JDIR.EQ.4) GOTO 75
IF(JNUM.EQ.0) GOTO 80
JCODE(JNUM,JDIR)=0
GOTO 70

75 JCODE(JNUM,1)=0
   JCODE(JNUM,2)=0
   JCODE(JNUM,3)=0
   GOTO 70
80 RETURN
SUBROUTINE FORCE

SUBROUTINE FORCE(CG1, CG2, CG3, C1, C2, C1H, C2H, Q, GAMMA, MCODE, ID, NJ, NE,
1ICOORD, NEQS, P, MINC)
DIMENSION CG1(NE), CG2(NE), CG3(NE), C1(NE), C2(NE), C1H(NE), C2H(NE), Q(1
NEQS), GAMMA(NE), P(NJ, 3), D(6), MCODE(NE, 6), ID(NE), MINC(NE, 2)
DO 10 I = 1, NJ
   DO 10 J = 1, 3
10 P(I, J) = 0.
WRITE(2, 99)
99 FORMAT(/// ' ELEMENT FORCES')
WRITE(2, 101)
101 FORMAT(4X, 'ELEMENT', 6X, 'FORCE')
IF (ICOORD.EQ.1) IDI = 5
   DO 20 I = 1, NE

C COMPUTE D FROM Q USING MCODE.

C

DO 30 J = 1, 6
   K = MCODE(I, J)
IF (K.EQ.0) GOTO 40
   D(J) = Q(K)
   GOTO 30
40 D(J) = 0.
30 CONTINUE
C3I = CG3(I)
GAMMAI = GAMMA(I)
GOTO(200, 210), ICOORD
200 C1I = CG1(I)
   C2I = CG2(I)
GOTO 220
COMPUTE THE ELEMENT FORCES MAKING APPROPRIATE SIMPLIFICATIONS FOR THE VARIOUS ID NUMBERS.

```
210  C1I=C1(I)
     C2I=C2(I)
     IDI=IDI(I)
GOTO(60,70,80,90,100),IDI
220  DL4=D(1)-D(4)
     D36=D(3)-D(6)
     F1=GAMMA1*(D14*C1I+D36*C3I)
     GOTO 43
60   DL4=D(1)+D(4)
     D25=D(2)-D(5)
     F1=GAMMA1*(DL4*C1I+D25*C2I)
     GOTO 43
70   DL4=D(1)+D(4)
     D25=D(2)-D(5)
     D36=D(3)-D(6)
     F1=GAMMA1*(DL4*C1I+D25*C2I+D36*C3I)
     GOTO 43
80   C1HI=C1H(I)
     C2HI=C2H(I)
     D36=D(3)-D(6)
     F1=GAMMA1*(C1I*D(1)*C2I*D(2)+C3I*D36-C1HI*D(4)-C2HI*D(5))
     GOTO 43
90   DL4=D(1)-D(4)
     D25=D(2)-D(5)
     D36=D(3)-D(6)
     F1=GAMMA1*(DL4*C1I+D25*C2I+D36*C3I)
     GOTO 43
100  IF(F1)44,48,49
43   F1=-F1
```
WRITE(2,45)I,F1
45 FORMAT(I8,4X,F13.7,3X,'TENSION')
F1=-F1
GOTO 52
48 WRITE(2,53)I,F1
53 FORMAT(I8,4X,F13.7)
GOTO 52
49 WRITE(2,54)I,F1
54 FORMAT(I8,4X,F13.7,3X,'COMPRESSION')

C COMPUTE THE JOINT FORCES FROM THE ELEMENT FORCES AND MINC.

C 52 J=MINC(I,1)
K=MINC(I,2)
P(J,1)=P(J,1)+C11*F1
GOTO(55,56,57,58,59),IDI
55 P(J,3)=P(J,3)+C31*F1
P(K,1)=P(K,1)-C11*F1
P(K,3)=P(K,3)-C31*F1
GOTO 20
56 P(J,2)=P(J,2)+C21*F1
P(K,1)=P(K,1)+C11*F1
P(K,2)=P(K,2)-C21*F1
GOTO 20
57 P(J,2)=P(J,2)+C21*F1
P(J,3)=P(J,3)+C31*F1
P(K,1)=P(K,1)+C11*F1
P(K,2)=P(K,2)-C21*F1
P(K,3)=P(K,3)-C31*F1
GOTO 20
58 P(J,2)=P(J,2)+C21*F1
P(J,3)=P(J,3)+C31*F1
\( P(K,1) = P(K,1) - C_{11} F \)  
\( P(K,2) = P(K,2) - C_{21} F \)  
\( P(K,3) = P(K,3) - C_{31} F \)  
GOTO 20

\( P(J,2) = P(J,2) + C_{21} F \)  
\( P(J,3) = P(J,3) + C_{31} F \)  
\( P(K,1) = P(K,1) - C_{11} F \)  
\( P(K,2) = P(K,2) - C_{21} F \)  
\( P(K,3) = P(K,3) - C_{31} F \)  
GOTO 20

\( P(J,2) = P(J,2) + C_{21} F \)  
\( P(J,3) = P(J,3) + C_{31} F \)  
\( P(K,1) = P(K,1) - C_{11} F \)  
\( P(K,2) = P(K,2) - C_{21} F \)  
\( P(K,3) = P(K,3) - C_{31} F \)  

CONTINUE

WRITE(2,46)

46 FORMAT(///' JOINT FORCES')
WRITE(2,51)

51 FORMAT(4X,'JOINT',6X,'1-DIR.',11X,'2-DIR.',11X,'3-DIR.')
DO 50 J=1,NJ

50 WRITE(2,47)J,(P(J,K),K=1,3)

47 FORMAT(17,2X,3F17.7)
RETURN

END
SUBROUTINE LOKATE

SUBROUTINE LOKATE(JCODE,LOCATE,NJ,NEQS)
DIMENSION JCODE(NJ,3),LOCATE(NEQS,2)
DO 10 I=1,NJ
   DO 10 J=1,3
      JC=JCODE(I,J)
      IF(JC.EQ.0)GOTO 10
      LOCATE(JC,1)=I
      LOCATE(JC,2)=J
10   CONTINUE
RETURN
SUBROUTINE PROCES

SUBROUTINE PROCES(X1,X2,X3,CG1,CG2,CG3,C1,C2,C1H,C2H,GAMMA,XL,MCOD
1E,JCODE,MINC,ID,NEQS,NJ,NE,INHBW,A,E,ICORD)

DIMENSION X1(NJ),X2(NJ),X3(NJ),CG1(NE),CG2(NE),CG3(NE),C1(NE),C2(NE),C1H(NE),C2H(NE),
GAMMA(NE),MCODE(NE,6),JCODE(NJ,3),MINC(NE,2),ID
2(NE),XL(NE),A(NE),E(NE)

GENERATE JCODE AND DETERMINE THE NUMBER OF DEGREES OF FREEDOM.

NEQS=0
DO 10 I=1,NJ
   DO 10 J=1,3
      NEQS=NEQS+JCODE(I,J)
   10  CONTINUE
WRITE(2,1)
1 FORMAT(///' JCODE')
DO 20 I=1,NJ
20 WRITE(2,2)(JCODE(I,J),J=1,3)
2 FORMAT(315)
WRITE(2,501)NEQS
501 FORMAT(///' NEQS=',I3)
MAXID=0

COMPUTE MEMBER PROPERTIES AND GENERATE MCODE WITHIN THE SAME DO-LOOP.

GENERATE MCODE FROM JCODE WITH INFORMATION CONTAINED IN MINC.

DO 40 I=1,NE
   J=MINC(I,1)
40 CONTINUE
K=MINC(I,2)

DO 41 L=1,3
    MCODE(I,L)=JCODE(J,L)
    MCODE(I,L+3)=JCODE(K,L)

41

COMPUTE MEMBER PROPERTIES—LENGTH, CG1, CG2, CG3, C1, C2, CIH, C2H, AND GAMMA, TAKING ID NUMBER INTO ACCOUNT AND MAKING THE APPROPRIATE SIMPLIFICATIONS.

XL1=X1(K)-X1(J)
XL2=X2(K)-X2(J)
XL3=X3(K)-X3(J)
XL(I)=SQRT(XL1**2+XL2**2+XL3**2)
XLONG=XL(I)
CG1(I)=XL1/XLONG
CG2(I)=XL2/XLONG
GAMMA(I)=A(I)*E(I)/XLONG
GOTO(200,300),ICOORD

200
CG3(I)=XL3/XLONG
GOTO 110

300
XLJ=X1(J)
X2J=X2(J)
R=SQRT(X1J**2+X2J**2)
CG1I=CG1(I)
CG2I=CG2(I)
IF(R.NE.0.)GOTO 61
C1(I)=CG1I
C2(I)=CG2I
GOTO 65

61
C1(I)=X1J/R*CG1I+X2J/R*CG2I
C2(I)=-X2J/R*CG1I+X1J/R*CG2I

65
IDI=ID(I)
GOTO(70, 80, 70, 90, 101)
70  XL3=X3(K)-X3(J)
    CG3(I)=XL3/XLONG
    GOTO 110
80  CG3(I)=0.
    GOTO 110
90  XL3=X3(K)-X3(J)
    CG3(I)=XL3/XLONG
    X1K=X1(K)
    X2K=X2(K)
    R=SQRT(X1K**2+X2K**2)
    IF(R.NE.0.)GOTO 100
    C1H(I)=CG1I
    C2H(I)=CG2I
    GOTO 110
100 C1H(I)=X1K/R*CG1I+X2K/R*CG2I
    C2H(I)=-X2K/R*CG1I+X1K/R*CG2I
C
C COMPUTE AND WRITE THE HALF BAND WIDTH.
C
110  J=0
42   J=J+1
    IS=MCODE(I,J)
    IF(IS.EQ.C)GOTO 42
    J=7
44   J=J-1
    IL=MCODE(I,J)
    IF(IL.EQ.0)GOTO 44
    IE=IL-IS
40   IF(IE.GT.MAXID)MAXID=IE
    IHBW=MAXID+1
    WRITE(2,8) IHBW

8 FORMAT(// * IHBW='13)
CWRITE OUT MEMBER PROPERTIES.
CWRITE(2, 3)
3 FORMAT(// * MCODE*)
   DO 50 I=1,NE
50 WRITE(2, 4) (MCODE(I, J), J=1, 6)
4 FORMAT(6I5)
   GOTO(51, 52), ICOORD
51 WRITE(2, 7)
7 FORMAT(// * MEMBER*, 3X, 'LENGTH', 8X, 'CG1', 10X, 'CG2', 10X, 'CG3', 6X, 'GAMMA')
   DO 55 I=1, NE
55 WRITE(2, 161) I, XL(I), CG1(I), CG2(I), CG3(I), GAMMA(I)
161 FORMAT(I5, F11.2, 3F13.7, F11.4)
   RETURN
52 WRITE(2, 6)
   DO 60 I=1, NE
   IDI=ID(I)
   GOTO(120, 130, 140, 150), IDI
120 WRITE(2, 121) I, XL(I), CG1(I), CG2(I), CG3(I), C1(I), C2(I), GAMMA(I)
   GOTO 60
130 WRITE(2, 131) I, XL(I), CG1(I), CG2(I), CG3(I), C1(I), C2(I), GAMMA(I)
   GOTO 60
140 WRITE(2, 141) I, XL(I), CG1(I), CG2(I), CG3(I), C1(I), C2(I), GAMMA(I)
   GOTO 60
150 WRITE(2, 151) I, XL(I), CG1(I), CG2(I), CG3(I), C1(I), C2(I), C1H(I), C2H(I), GAMMA(I)
   GOTO 60
1 CONTINUE
121 FORMAT(I5,F11.2,5F13.7,6X,'RADIAL ELEMENT',6X,F11.4)
131 FORMAT(I5,F11.2,5F13.7,7X,'RING ELEMENT',7X,F11.4)
141 FORMAT(I5,F11.2,5F13.7,5X,'SKEW RING ELEMENT',4X,F11.4)
151 FORMAT(I5,F11.2,7F13.7,F11.4)
   RETURN
SUBROUTINE SOLVE

DIMENSION SS(NEQS,NEQ),Q(NEQS),LOCATE(NEQS,2)

IF(NACT.GT.NCOUNT)GOTO 800

C REDUCE THE STIFFNESS MATRIX.

DO 790 N=1,NEQS
   DO 780 L=2,NEQBW
      IF(SS(N,L).EQ.0.)GOTO 780
      I=N+L-1
      C=SS(N,L)/SS(N,1)
      J=0
      DO 750 K=L,NEQBW
         J=J+1
         SS(I,J)=SS(I,J)-C*SS(N,K)
      END
      SS(N,L)=C
   CONTINUE

C REDUCE FORCE VECTOR.

CONTINUE

DO 830 N=1,NEQS
   DO 820 L=2,NEQBW
      IF(SS(N,L).EQ.0.)GOTO 820
      I=N+L-1
      Q(I)=Q(I)-SS(N,L)*Q(N)
   CONTINUE

C BACK-SOLVE FOR THE GENERALIZED DISPLACEMENTS.
C

830  Q(N)=Q(N)/SS(N,1)

DO 860 M=2,NEQS
     N=NEQS+1-M
     DO 850 L=2,IHBW
        IF(SS(N,L).EQ.0.)GOTO 850
        K=N+L-1
        Q(N)=Q(N)-SS(N,L)*Q(K)
     850 CONTINUE

860  CONTINUE

WRITE(2,61)

61 FORMAT(//' GENERALIZED DISPLACEMENTS')

DO 900 I=1,NEQS

900  WRITE(2,60) I,Q(I),LOCATE(I,1),LOCATE(I,2)

60 FORMAT(4X,'Q(',I3,') =',F11.7,3X,'(JOINT',I3,', ',I1,'-DIRECTION)',1)

RETURN
SUBROUTINE STIFF(SS,CG1,CG2,CG3,C1,C2,C1H,C2H,GAMMA,MCODE,ID,NEQS, 
IHBW,ICOORD,NE)
DIMENSION SS(NEQS,IHBW),CG1(NE),CG2(NE),CG3(NE),C1(NE),C2(NE),C1H( 
NE),C2H(NE),GAMMA(NE),G(15),INDEX(6,6),MCODE(NE,6),ID(NE)
INTEGER INDEX/1,0,3,-1,0,-3,6*0,3,0,10,-3,0,-10,-1,0,-3,1,0,3,6*0, 
1-3,0,-10,3,0,10,1,2,0,1,-2,0,2,6,0,2,-6,7*0,1,2,0,1,-2,0,-2,-6,0,- 
22,6,7*0,1,2,3,1,-2,-3,2,6,7,2,-6,-7,3,7,10,3,-7,-7,-10,1,2,3,1,-2,-3, 
3-2,-6,-7-2,6,7-3,-7-10,1,2,3,-4,-5,3-2,6,7,8,9,-9,-7,3, 
47,10,-11,-12,-10,-4,-8,-11,13,14,11,-5,-9,-12,-14,-15,-12,-3,-7,-10,1 
51,12,10,1,2,-3,1,-2,-3,2,6,7,-2,-6,-7,3,7,10,-3,-7,-10,1,-2,3,1, 
62,-3,-2,-6,-7,2,6,7,3,-7-10,3,7,10/
DO 1 I=1,NEQS
  DO 1 J=1,IHBW
    SS(I,J)=0.
IF(ICOORD.EQ.1)IDI=5
  DO 10 I=1,NE
    GAMMAI=GAMMA(I)
    C3I=CG3(I)
    GOTO(200,300),ICOORD
  200 C1I=CG1(I)
  300 C2I=C2(I)
  GOTO 100
  300 C1I=C1(I)
  200 C2I=C2(I)
  IDI=ID(I)

COMPUTE THE APPROPRIATE G FACTORS FOR THE ELEMENT BASED ON ITS ID NUMBER.
G(1)=GAMMAI*C1I**2
GOTO(110,120,130,140,130),ID1

110  G(3)=GAMMAI*C1I*C31
     G(10)=GAMMAI*C31**2
     GOTO 150

120  G(2)=GAMMAI*C1I*C2I
     G(6)=GAMMAI*C2I**2
     GOTO 150

130  G(2)=GAMMAI*C1I*C2I
     G(3)=GAMMAI*C1I*C31
     G(6)=GAMMAI*C2I**2
     G(7)=GAMMAI*C2I*C31
     G(10)=GAMMAI*C31**2
     GOTO 150

140  C1HI=C1H(1)
     C2HI=C2H(1)
     G(2)=GAMMAI*C1I*C2I
     G(3)=GAMMAI*C1I*C31
     G(4)=GAMMAI*C1I*C1HI
     G(5)=GAMMAI*C1I*C2HI
     G(6)=GAMMAI*C2I**2
     G(7)=GAMMAI*C2I*C31
     G(8)=GAMMAI*C2I*C1HI
     G(9)=GAMMAI*C2I*C2HI
     G(10)=GAMMAI*C31**2
     G(11)=GAMMAI*C31*C1HI
     G(12)=GAMMAI*C31*C2HI
     G(13)=GAMMAI*C1HI**2
     G(14)=GAMMAI*C1HI*C2HI
     G(15)=GAMMAI*C2HI**2

C  PLACE G FACTORS IN BANDED SYSTEM STIFFNESS MATRIX USING INFORMATION CONTAINED IN MCODE AND INDEX.
C

150 DO 20 JM=1,6
   J=MCODE(I,JM)
   IF(J.EQ.0)GOTO 20
   DO 30 KM=JM,6
      K=MCODE(I,KM)
      IF(K.EQ.0)GOTO 30
      KB=K-J+1
      L=INDEX(JM,KM,IDI)
      IF(L)60,30,40
      L=-L
      SS(J,KB)=SS(J,KB)-G(L)
   GOTO 30
40   SS(J,KB)=SS(J,KB)*G(L)
30   CONTINUE
20   CONTINUE
10   CONTINUE
RETURN
APPENDIX C

NONLINEAR PROGRAM
THIS PROGRAM USES THE MODIFIED RIKS/WEMPNER METHOD TO TRACE THE EQUILIBRIUM PATHS OF GEOMETRICALLY NONLINEAR SPACE TRUSSES. IT IS ASSUMED THAT THE USER IS FAMILIAR WITH THE METHOD.

This program makes use of five different files:

- FILE 1 contains all input data.
- FILE 2 contains output of the data and the normal vector used in branching onto a secondary equilibrium path.
- FILE 3 contains output of the load level and displacement vector of the equilibrium points.
- FILE 4 contains output of the load level and displacement vector of the equilibrium points obtained in the determination of the critical point.
- FILE 7 contains output of the load level and displacement vector of all equilibrium points in a suitable format for input into a plotting program.
THE EXECUTION OF THIS PROGRAM CAN BE BROKEN INTO SEVEN DIFFERENT CATEGORIES.

TYPE 1: N number of points are obtained on the equilibrium path.

TYPE 2: One requires the program to execute until a stable point is encountered.

TYPE 3: N number of points are obtained after the first critical point. It is recommended that N be less than 5.

TYPE 4: Branching onto secondary equilibrium paths. The search for the critical point will begin at point N and M points are obtained on the branching path.

TYPE 5: Determination of a critical point. The search begins at point N.

TYPE 6: Running the program to obtain the first critical point. The number of points, N, should be large enough for the program to reach a critical point.

TYPE 7: Starting the program at a known equilibrium point on the equilibrium path. This can be done in combination with types 1-6.
THE FOLLOWING VARIABLES IN THE INPUT DATA ARE TABULATED TO AID THE USER IN DETERMINING INPUT VALUES.

N  E  E  B  M  B  E  E  E
P  N  N  R  P  P  P  P  P  L
C  T  O  C  D  D  A  Q  Q  S  S  S  S  A  N  D  E  D
C  Y  I  O  L  S  N  I  I  L  L  M  N  E  X  S
C  P  N  I  T  C  N  N  N  N  N  D  E  T  P  B
C  E  T  T  M  B  H  T  T  1  2  3  A  G  U  O  R  Q  O
C  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  --
C  1  N  1,2  2  2  -#  a  2  S  S  a  +  +  C  C  a  +
C  2  a  2  2  1  -#  a  2  S  S  a  +  +  C  C  a  +
C  3  N  1,2  1  2  -#  a  2  S  S  a  +  +  C  C  a  +
C  4  a  1,2  2  2  N  M  2  S  S  S  +  +  C  C  S  +
C  5  a  2  2  2  N  a  1  S  S  S  +  +  C  C  a  +
C  6  N  2  2  2  -#  a  3  S  S  S  +  +  C  C  a  +
C  7  (CAN BE USED IN 1-6)  S  S  S  S  S  S

WHERE N AND M ARE DEFINED ABOVE
S = SPECIFY
# = ARBITRARY NUMBER
a = NOT APPROPRIATE, BUT SPECIFY
QO = INITIAL DISPLACEMENT VECTOR (NOT ENTERED IF ZERO)
+ = ZERO IF NOT USING TYPE 7
C = CAN BE SPECIFIED
1,2 = 1 OR 2
FOR VARIABLE INPUT, (I) REPRESENTS AN INTEGER VARIABLE
(R) REPRESENTS A REAL VARIABLE.

***** ALL INPUT IS FORMAT FREE *****

DATA SET

1 INPUT OF THE FOLLOWING VARIABLES

NM (I)
NJ (I)
SYSTEM (I)
UPDATE (I)
IHBW (I)
SCALE (I)
LHAT (I)
LMAX (I)
NPOINT (I)
METHOD (I)
CONT (I)
ENDLIM (I)
ENDST8 (I)
INTEST (I)
BRANCH (I)
IMAX (I)
JMAX (I)
INPUT OF THE FOLLOWING VARIABLES

EPSLN1 (R)
EPSLN2 (R)
EPSLN3 (R)
NDIGIT (I)
DS (R) (SPECIFY EITHER DS OR DLAMD)
DLAMD (R) AS NONZERO
LAMDA (R)
NNEG (I)
DETO (R)
EXPO (I)
OSBR (R)

JOINT COORDINATE DATA IN INCREASING JOINT NUMBER

(X, Y, Z) OR (R, THETA, PHI)
(R) (R) (R) (R) (R) (R)

IN SPHERICAL COORDINATES:

R = RADIUS FROM THE ORIGIN
THETA = ANGLE FROM THE GLOBAL 2-AXIS IN THE 1-2 PLANE
PHI = ANGLE FROM THE GLOBAL 3-AXIS

CONSTRAINED JOINT DATA. A JOINT DIRECTION OF 4
INDICATES THAT ALL 3 DEGREES OF FREEDOM FOR THAT
JOINT ARE CONSTRAINED. FOLLOW THIS SET WITH A
ZERO JOINT NUMBER AND JOINT DIRECTION.

JNUM, JDIR
(I)  (I)

5 MEMBER DATA IN INCREASING MEMBER NUMBER. A NEGATIVE
AREA FOR MEMBER 1 INDICATES THAT A AND E ARE 1.0 FOR
ALL MEMBERS AND THE REMAINING MEMBERS NEED ONLY SUPPLY
MINC.

A, E, MINC
(R) (R) (I) (I)

6 LOAD DATA. FOLLOW WITH A ZERO FOR JNUM, JDIR, AND P1

JNUM, JDIR, P1
(I)  (I)  (R)

7 DISPLACEMENT VECTOR. FOLLOW SET 6 WITH A NEGATIVE
JOINT NUMBER IF THERE IS NO DISPLACEMENT VECTOR.
COMMON /BLOCK1/ SS(NEQS),IHBW),DQ(NEQS),DQ1(NEQS),DQII(NEQS),
                   DISTR(NEQS),LOAD(NEQS),Q(NEQS),R(NEQS),QI(NEQS),
                   R1(NEQS),KPVT(NEQS)
COMMON /BLOCK2/ A(NM),CG1(NM),CG2(NM),CG3(NM),E(NM),XL(NM),
                   XL0(NM),M CODE(NM,6),MINC(NM,2)
COMMON /BLOCK3/ X(NJ),Y(NJ),Z(NJ),C1(NJ),C2(NJ),JCODE(NJ,3)
COMMON /BLOCK4/ DIMENSIONS ARE THE SAME FOR ALL PROBLEMS.
FOR SUBROUTINE EIGEN: XTEMP(NEQS)
AT THE BEGINNING OF MAINLINE, LDA = DIMENSIONED VALUE OF NEQS.

NM = NUMBER OF MEMBERS
NJ = NUMBER OF JOINTS
NEQS = NUMBER OF EQUATIONS (DEGREES OF FREEDOM)
IHBW = HALF-BAND WIDTH

NOTES: C1 AND C2 ARE ONLY USED WHEN THE PROGRAM IS OPERATING IN
CYLINDRICAL COORDINATES. TO AVOID WASTING STORAGE SPACE
DIMENSION (1) WHEN OPERATING IN GLOBAL COORDINATES.
KPVT IS USED ONLY WHEN USING THE LINPACK EQUATION SOLVER
TO AVOID WASTING STORAGE SPACE, DIMENSION (1) WHEN USING
THE BANDED EQUATION SOLVER.
WHEN USING THE LINPACK EQUATION SOLVER, DIMENSION
SS(NEQS,NEQS).
********** FOR ALL VARIABLES, 1=YES, 2=NO **********

A(I) AREA OF MEMBER I.
A1, B1, D1 COEFFICIENTS OF THE QUADRATIC EQUATION IN
SUBROUTINE SPHERE.
BPOINT ALLOWS CONVERGENCE ON THE BIFURCATION
POINT WITHOUT BRANCHING ONTO ANOTHER PATH.
BRANCH DETERMINES IF THE PROGRAM IS TO BRANCH OFF
OF THE FUNDAMENTAL PATH ONTO THE BRANCHING
PATH.
   -# = DO NOT BRANCH
   0 = BRANCH IMMEDIATELY
   # = BEGIN BRANCHING ROUTINE AT POINT #
CG1(I), CG2(I), CG3(I) DEFORMED GLOBAL DIRECTION COSINES FOR MEM-
BER I.
C1(I), C2(I) THE GLOBAL DIRECTION COSINES FOR THE 1-AXIS
FOR JOINT I WHEN WORKING IN CYLINDRICAL CO-
ORDINATES.
CONT DETERMINES IF THE PROGRAM SHOULD CONTINUE
TO EXECUTE WHEN THE PROGRAM HAS BECOME
MORE STABLE (ALL NEGATIVE EIGENVALUES HAVE
DISAPPEARED).
D(I) DISPLACEMENT FOR AN ELEMENT IN DIRECTION I,
OR MEMBER END FORCES USED TO COMPUTE R(I).
DET THE DETERMINANT OF THE TANGENT STIFFNESS
MATRIX LESS THE SCALE FACTOR.
DETO  DET FOR THE UNDEFORMED TRUSS. IF DETO IS
INPUT AS 0, IT WILL BE CALCULATED.
(INPUT THE DETERMINANT WHEN STARTING AT A
KNOWN POINT ON THE EQUILIBRIUM PATH)

DETNEW  THE NEW DETERMINANT WHEN CONVERGING ON THE
BIFURCATION POINT.

DETOLD  THE OLD DETERMINANT WHEN CONVERGING ON THE
BIFURCATION POINT.

DISTR(I) LOAD DISTRIBUTION VECTOR.

DLAMD CHANGE IN LOAD LEVEL.

DLAMD1 CHANGE IN LOAD LEVEL AT THE FIRST STEP.

DQ(I) CHANGE IN DISPLACEMENT.

DQI(I) SOLUTION TO K DQKI = DISTR FOR ITERATION I.

DQII(I) SOLUTION TO K DQKI = DISTR.

DS THE LENGTH OF THE TANGENT VECTOR.

DSBR THE INITIAL CHANGE IN LOAD LEVEL WHEN
BRANCHING.

E(I) YOUNG'S MODULUS FOR MEMBER I.

ELONG ELONGATION OF A MEMBER.

ENDLIM STOPS EXECUTION AT NPOINT POINTS BEYOND THE
FIRST CRITICAL POINT.

ENDSTB STOPS EXECUTION WHEN THE TRUSS HAS BECOME
STABLE.

EPSLN1 CONVERGENCE CRITERIA FOR THE UNBALANCED FORCE
TO OBTAIN THE POINT ON THE EQUILIBRIUM PATH.

EPSLN2 CONVERGENCE CRITERIA FOR THE CHANGE IN
DISPLACEMENT TO OBTAIN THE POINT ON THE
EQUILIBRIUM PATH.

EPSLN3 FOR CONVERGENCE TO THE BIFURCATION POINT,
THIS IS THE TOLERANCE OF THE RATIO OF THE
DETERMINANT OF THE STIFFNESS MATRIX AT SOME
POINT TO THE DETERMINANT OF THE STIFFNESS
C MATRIX FOR THE UNDEFORMED TRUSS.
C EXP THE VALUE OF THE EXPONENT THAT THE DETER-
C MINANT OF THE TANGENT STIFFNESS MATRIX IS
C SCALED BY TO AVOID OVER/UNDERFLOW ERRORS.
C EXPO THE VALUE OF EXP FOR THE UNDEFORMED TRUSS.
C EXPNEW THE NEW VALUE OF EXP WHEN CONVERGING ON THE
C BIFURCATION POINT.
C EXPOLD THE OLD VALUE OF EXP WHEN CONVERGING ON THE
C BIFURCATION POINT.
C G1,G2 ROOTS OF THE QUADRATIC EQUATION IN SPHERE.
C ICOORD INDICATES WHETHER THE JOINT CONSTRAINTS,
C FORCES, AND DISPLACEMENTS ARE IN GLOBAL OR
C CYLINDRICAL COORDINATES.
C ICOORD = 1, GLOBAL COORDINATES
C ICOORD = 2, CYLINDRICAL COORDINATES
C IFLAG INDICATES IF THE ITERATION PROCESS HAS
C CONVERGED.
C IHBW THE HALF-BAND WIDTH OF THE SYSTEM STIFFNESS
C MATRIX. THIS IS INPUT TO VERIFY THAT THE
C MATRIX IS DIMENSIONED LARGE ENOUGH.
C IHELP 'TURNS OFF THE AUTOMATION WHEN BRANCHING'.
C NOTE: IF IHELP = 3 (YES) THE AUTOMATION IS
C 'TURNED OFF' FOR ALL ITERA-
C TIONS.
C IHELP = 1 (YES) THE AUTOMATION IS
C 'TURNED OFF' FOR THE FIRST
C ITERATION ONLY.
C IMAX THE MAXIMUM NUMBER OF ITERATIONS TO CONVERGE
C ON THE BIFURCATION POINT.
C INTEST STOPS THE PROGRAM AFTER THE DATA IS INPUT.
C ISCALE RECORDS THE SCALE VALUE SO THAT THE ASSUMED
C DIRECTION IN SUBROUTINE EIGEN CAN BE DETER-
ISOLVE determines whether the banded equation solver or the LINPACK solver is used.

If ISOLVE = 1, banded solver

If ISOLVE = 2, LINPACK solver

ITRACE(I) records critical locations in the program for future reference in tracing the 'path' that the program followed.

JCODE(I,J) degree of freedom (DOF) in direction J for joint I.

JFLAG determines if subroutine SOLVE is used to perform forward reduction of the matrix or to solve a set of equations.

If JFLAG=0, forward reduction

JFLAG=1, SOLVE equations

JHBW verifies that JHBW is large enough.

JMAX the maximum number of iterations for the determination of the eigenvector used to branch onto the branching path.

K15 the counter for storing values in ITRACE.

L number of iterations in a step.

LAMDA load level at point I.

LAMDAI load level for the first step.

LDA control parameter used in the LINPACK equation solver which must be set equal to the dimensioned value of NEQS.

LHAT number of desired iterations in a step.

LMAX maximum number of iterations in a step.

LOAD(I) the force vector used in SOLVE to obtain the displacement vector. This vector is also used as temporary storage throughout the program.
MCOQ(1,J) DOF IN DIRECTION J FOR MEMBER I.

METHOlJ) METHOD PROCEDURE USED IN THE PROGRAM.

1=NORMAL PLANE ITERATION

2=SPHERE ITERATION

3=NEWTON-RAPHSON

MINCl,l,J MEMBER INCIDENCE FOR MEMBER I AT END J.

MPOlJN NUMBER OF POINTS WANTED AFTER BRANCHING.

NCOUNT THE NUMBER OF THE POINT ON THE EQUILIBRIUM PATH.

NDIGIT THE NUMBER OF SIGNIFICANT DIGITS WANTED FOR THE EIGENVALUE IN SUBROUTINE POWINV.

NEQS NUMBER OF EQUATIONS.

NJ NUMBER OF JOINTS.

NM NUMBER OF MEMBERS.

NNEG NUMBER OF NEGATIVE EIGENVALUES FOR SS.

NPOINT NUMBER OF POINTS WANTED ON THE EQUILIBRIUM PATH.

PHI ANGLE OF ROTATION ABOUT THE Z-AXIS IN POLAR COORDINATES.

Q(I) DISPLACEMENT FOR DOF I.

Q1(I) DISPLACEMENT FOR DOF I BEFORE THE FIRST STEP.

R(I) UNBALANCED FORCE VECTOR.

R1(I) THE EIGENVECTOR USED TO OBTAIN THE NORMAL VECTOR.

RADIUS THE VECTOR LENGTH IN POLAR COORDINATE DATA.

SCALE DETERMINES IF THE TANGENT VECTOR IS TO BE SCALLED.

SS(1,J) THE SYSTEM TANGENT STIFFNESS MATRIX. (STORED IN BANDED FORM)

SYSTEM COORDINATE SYSTEM USED FOR INPUT OF JOINT LOCATIONS.

1=Spherical

2=RECTANGULAR
THETA ANGLE FROM THE Z-AXIS IN POLAR COORDINATES.
UPDATE DETERMINES IF THE STIFFNESS MATRIX IS TO BE
UPDATED EVERY ITERATION.
X(I),Y(I),Z(I) GLOBAL JOINT COORDINATES.
RECTANGULAR: X=X, Y=Y, Z=Z
SPHERICAL: X=R, Y=THETA, Z=PHI
X(I),Y(I),Z(I) VARIABLES USED FOR CALCULATIONS IN
SUBPROGRAMS.
XL(I) UNDEFORMED LENGTH OF MEMBER I.
XLAMDA(I) TEMPORARY STORAGE FOR Q(I) FOR THE CASE
WHERE THE ASSUMPTION IN SUBROUTINE EIGEN IS
WRONG.
XLD(I) DEFORMED LENGTH OF MEMBER I.
XTEMP TEMPORARY STORAGE FOR LAMDA FOR THE CASE
WHERE THE ASSUMPTION IN SUBROUTINE EIGEN IS
WRONG.

C*********************************************************************
C* SUBROUTINE DESCRIPTIONS *
C*********************************************************************

BPATH THIS SUBROUTINE FINDS THE CRITICAL POINT GIVEN THE TOLERANCE
EPSLN3. IT IS ASSUMED THAT THE POINT FROM WHICH THE SEARCH
IS MADE IS RELATIVELY 'CLOSE' TO THE CRITICAL POINT. THIS
ALLOWS ONE TO ASSUME A LINEAR RELATIONSHIP BETWEEN DS AND THE
DETERMINANT OF THE TANGENT STIFFNESS MATRIX. (AT A CRITICAL
POINT, THE DETERMINANT IS ZERO.) EPSLN3 SPECIFIES THE LOSS
IN STIFFNESS OF THE TRUSS.

CONVG THIS SUBROUTINE DETERMINES IF THE RIKS/WEMPNER ITERATION
HAS CONVERGED ON THE EQUILIBRIUM PATH.
DATA THIS SUBROUTINE INPUTS THE DATA FOR THE PROGRAM FROM FILE
NUMBER 1. SOME INITIALIZATION OF VARIABLES OCCURS HERE ALSO.

DEFORM THIS SUBROUTINE DETERMINES THE MEMBERS DEFORMED LENGTHS AND
DEFORMED COSINES FOR THE PRESENT CONFIGURATION.

DTNEG1 THIS SUBROUTINE CALCULATES THE NUMBER OF NEGATIVE EIGEN-
VALUES AND THE DETERMINANT OF THE TANGENT STIFFNESS MATRIX
USING THE BANDED SOLVER. THIS IS PERFORMED AFTER THE FORWARD
REDUCTION OF THE STIFFNESS MATRIX.

DTNEG2 THIS SUBROUTINE PERFORMS THE SAME FUNCTION AS DTNEG1 USING THE
LINPACK EQUATION SOLVER.

EIGEN THIS SUBROUTINE AUTOMATES THE PROGRAM SO THAT IT WILL
DETERMINE THE DIRECTION OF TRAVEL WHEN CRITICAL POINTS ARE
ENCOUNTERED. SEVERAL CONTROL VARIABLES ARE USED HERE AND
THEY CAUSE THE SUBROUTINE TO APPEAR COMPLEX.

F THIS FUNCTION DETERMINES THE DOT PRODUCT OF SEVERAL VECTORS.
FOR THE ARGUMENTS IN THE ARGUMENT LIST
F = X1 * ( Y1 + K * Z1 )

INIT THIS SUBROUTINE DETERMINES THE INITIAL TANGENT VECTOR FOR
THE ITERATION TO BEGIN. IT ALSO CALCULATES DS OR DLAMD
FOR THE INITIAL CONFIGURATION. (EITHER DS OR DLAMD IS
SPECIFIED IN INPUT)

NEWTON THIS SUBROUTINE USES THE NEWTON-RAPHSON METHOD TO DETERMINE
THE EQUILIBRIUM PATH.
NORMAL: This subroutine determines $\lambda$ and $q$ for the 'Normal Plane' iteration.

POWINV: This subroutine uses the inverse power method to determine an eigenvalue and eigenvector of the tangent stiffness matrix. This method converges on the eigenvalue nearest to $\alpha$.

RESID: This subroutine determines the residual force vector from the applied load and the joint forces obtained from the displacement vector.

RWC: This subroutine uses the modified Riks/Wempner method to obtain the equilibrium path.

SOLVE1: This subroutine solves a set of simultaneous equations using the banded stiffness matrix. The first call to SOLVE (after the stiffness matrix has been calculated) does forward reduction of the matrix. All subsequent calls to SOLVE do the forward reduction of constants and back-substitution (until STIFF is called again).

SOLVE2: This subroutine performs the same functions as SOLVE1 except using the LINPACK equation solver. Subroutines SSIFA, SSISL, SSWAP, and SAXPY are called from here, as are functions SDOT and ISAMAX.

SPHERE: This subroutine determines $\lambda$ and $q$ for the 'Sphere' iteration.

STIFF: This subroutine calculates the banded tangent stiffness matrix.
STORE  THIS SUBROUTINE STORES A COMBINATION OF SEVERAL VECTORS INTO
ANOTHER VECTOR. FOR THE ARGUMENTS IN THE ARGUMENT LIST
\[ x_1 = y_1 + v * z_1 \]

VECTOR  THIS SUBROUTINE DETERMINES THE NORMAL VECTOR FROM THE
EIGENVECTOR IN POWINV. THIS ALLOWS ONE TO BRANCH ONTO
SECONDARY EQUILIBRIUM PATHS.
C*********************************************************************
C* MAINLINE *
C*********************************************************************
IMPLICIT REAL*8 (A-H,O-Z)
INTEGER SCALE, SYSTEM, UPDATE, CONT, ENOLIM, ENOSTB, BRANCH,
       EXP, EXPLO, EXPNEW, BPOINT
REAL*8 LOAD, LAMDA, LAMDA1
CHARACTER*2 ITRACE
COMMON /BLOCK1/ SS(70,70), DQ(70), DQI(70), DQI(70), DISTR(70),
                   LOA(70), Q(70), R(70), Q(70), R(70), KPV(70)
COMMON /BLOCK2/ A(70), CG1(70), CG2(70), CG3(70), E(70), X(70), XL(70), XLD(70)
                   MCODE(70,6), MINC(70,2)
COMMON /BLOCK3/ X(31), Y(31), Z(31), C1(31), C2(31), JCODE(31, 3)
COMMON /BLOCK4/ D(6), DLAMD, DLAMDI, DS, ELONG, EPSLN1, EPSLN2, EPSLN3,
                   DETO, DET, DETLD, DETNEW, LAMDA, LAMDA1, DSBR,
                   IHBW, JFLAG, JHBW, L, IFLAG, LHA1, LMAX, ITRACE(2000),
                   NCOUNT, NEQS, NJ, NM, NNEG, NPONT, SCALE, SYSTEM, K15,
                   UPDATE, CONT, ENOLIM, ENOSTB, INTESI, METHOD, BRANCH,
                   EXP, EXP1, EXPLOD, EXPNEW, IMAX, JMAX, MPOINT, NDIGIT,
                   HELP, BPOINT, LFLAG, ICOORD, NUMBER, LDA, ISOLVE
       LDA=70
CALL DATA
WRITE OUT THE INITIAL (STARTING) POINT DATA.
L=LHAT
K15=0
LFLAG=2
I=0
NUMBER=1
WRITE(3,1) I, LAMDA
1 FORMAT(13, ' LAMDA =', G24.16, ', ', 83('**'))
   WRITE(3,2)
2 FORMAT(' DISPLACEMENT')
WRITE(3,3) (Q(I),I=1,NEQS)
3 FORMAT(5G24.16)
WRITE(7,42) LAMDA
42 FORMAT(F15.9)
WRITE(7,41) (Q(I),I=1,NEQS)
NCOUNT=0
C CHECK TO SEE IF THE PROGRAM IS TO START BRANCHING FROM THE
C POINT ON THE PATH THAT WAS INPUT.
 IF (BRANCH.NE.0) GO TO 13
C $$$ A $$$
K15 = K15 + 1
ITRACE(K15) = 'A'
CALL DEFORM
CALL STIFF
CALL VECTOR
IMETH = METHOD
METHOD = 2
CALL RWC
METHOD = IMETH
GO TO 80
C CHECK TO SEE IF THE DETERMINANT OF THE UNDEFORMED TRUSS WAS
C INPUT.
13 IF (DETO.NE.0.0) GO TO 10
C $$$ B $$$
K15 = K15 + 1
ITRACE(K15) = 'B'
C DETERMINE THE INITIAL DETERMINANT.
CALL DEFORM
CALL STIFF
IF (ISOLVE.EQ.1) CALL SOLVE1
IF (ISOLVE.EQ.2) CALL SOLVE2
CALL EIGEN
DETO=DET
EXPO=EXP
NCOUNT=1
GO TO 30
10 NCOUNT=NCOUNT+1
C
$$\$C$$$
K15=K15+1
ITRACE(K15)='C'
C CHECK TO SEE IF THE BRANCHING ROUTINE IS TO BEGIN.
IF(BRANCH.NE.NCOUNT-1) GO TO 15
C
$$\$D$$$
20 K15=K15+1
ITRACE(K15)='D'
CALL BPATH
CALL DEFORM
CALL STIFF
CALL VECTOR
IMETH=METHOD
METHOD=1
CALL RWC
METHOD=IMETH
C CHECK TO SEE IF DS HAS THE CORRECT SIGN. (DS HAS THE SIGN
C OF DLAMDA.)
IF(DS.GT.0.00.AND.LAMDA.LT.LAMDA1) DS=-DS
IF(DS.LT.0.00.AND.LAMDA.GT.LAMDA1) DS=-DS
GO TO 80
C FIND IQKI FOR THE FIRST ITERATION.
15 CALL DEFORM
CALL STIFF
C NOTE: CALLING SOLVE AFTER STIFF ONLY CAUSES THE FORWARD
C REDUCTION OF THE CONSTANTS.
IF(ISOLVE.EQ.1)CALL SOLVE1
IF(ISOLVE.EQ.2)CALL SOLVE2
C CHECK TO SEE IF THERE IS A BIFURCATION OR LIMIT POINT.
CALL EIGEN
C CHECK TO SEE IF THE CRITICAL POINT IS TO BE LOCATED.
IF(LF-LAG.EQ.2)GOTO 12
DS=DS/2.DO
BPOINT=1
GOTO 20
C $$$ E $$$
12 K15=K15+1
ITRACE(K15)='E'
30 CALL STORE(LOAD,DISTR,LOAD,0.DO)
C SOLVE K DQKI = DISTR.
IF(ISOLVE.EQ.1)CALL SOLVE1
IF(ISOLVE.EQ.2)CALL SOLVE2
C GET THE TANGENT VECTOR.
CALL INIT
C CALL THE SUBROUTINE THAT CORRESPONDS TO THE METHOD DESIRED.
IF(METHOD.EQ.3) GO TO 60
CALL WRC
GO TO 70
60 CALL NEWTON
C WRITE OUT THE POINT DATA.
70 WRITE(7,42) LAMDA
WRITE(7,41) (Q(I),I=1,NEQS)
41 FORMAT(13F10.6)
80 WRITE(3,1) NCOUNT,LAMDA
WRITE(3,2)
WRITE(3,3) (Q(I),I=1,NEQS)
WRITE(3,32) L
32 FORMAT(' NUMBER OF ITERATIONS =' ,I3)
WRITE(3,31) DS
CONTINUE IF 1) A STABLE POINT IS WANTED (ENDSTB=1)
C OR 2) A LIMIT OR BIFURCATION POINT IS WANTED
C (ENOLIM=1)
C OR 3) NCOUNT NOT=NPOINT, ENOLIM=2, AND ENDSTB=2.
IF((NCOUNT.NE.NPOINT.AND.ENDSTB.NE.1.AND.ENOLIM.NE.1).OR.
1 ENDSTB.EQ.1.OR.ENOLIM.EQ.1) GO TO 10
IF(ISOLVE.EQ.1)CALL DTNEG1
IF(ISOLVE.EQ.2)CALL DTNEG2(SSID,NEQS,KPVT,DET,EXP,NNEG)
WRITE(3,11) NNEG,EXP,DET
11 FORMAT(' NUMBER OF NEGATIVE EIGENVALUES =',I3,' DETERMIN*
1 'ANT * 1.D',I3, ' =',G24.16,')
WRITE(2,1001)
1001 FORMAT(' TRACE OF PROGRAM PATH',/) WRITE(2,1002) (ITRACE(I),I=1,K15)
1002 FORMAT(1X,10A2,5X,10A2,5X,10A2,5X,10A2,5X,10A2,5X)
WRITE(2,4) NCOUNT
4 FORMAT(' NUMBER OF POINTS =',14,//, ' ***** NORMAL ',
1 'COMPLETION *****')
STOP
SUBROUTINE BPATH
IMPLICIT REAL*8 (A-H,O-Z)
INTEGER SCALE, SYSTEM, UPDATE, CONT, ENDLIM, ENSTB, BRANCH,
EXP, EXP2, EXP2D, EXP2N, BPOINT
REAL*8 LOAD, LAMDA, LAMDA1
CHARACTER*2 ITRACE
COMMON /BLOCK1/ SS(70, 70), DQ(70), DQ1(70), DQ2(70), DISTR(70),
LOAD(70), Q(70), Q1(70), Q2(70), Q3(70), Q4(70),
COMMON /BLOCK2/ A(70), C1(70), C2(70), C3(70), C4(70), C5(70),
COMMON /BLOCK3/ X(31), Y(31), Z(31), C1(31), C2(31), JCODE(31, 3),
COMMON /BLOCK4/ D(6), DLAMDA, DLAMDA1, DS, DLAMDA2, DLAMDA3,
DETO, DET2, DET2D, DET2N, LAMDA, LAMDA1, DSBR,
IHBRW, IHBRW2, IFLAG, IHBRW1, IHBRW3, LMAX, ITRACE(2000),
NCOUNT, NEQS, NJ, NM, NNEG, NPOINT, SCALE, SYSTEM, K15,
UPDATE, CONT, ENDLIM, ENSTB, INTEST, METHOD, BRANCH,
EXP, EXP2, EXP2D, EXP2N, IMAX, JMAX, MPOINT, NDIGIT,
HELP, BPOINT, LFLAG, ICOKD, NUMBER, LDA, ISOLVE

K15 = K15 + 1
ITRACE(K15) = 'S'
KOUNT = 0
ISCALE = SCALE
SCALE = 2
CALL DEFORM
CALL STIFF
IF(ISOLVE.EQ.1) CALL SOLVE1
IF(ISOLVE.EQ.2) CALL SOLVE2
IF(ISOLVE.EQ.1) CALL DTNEG1
IF(ISOLVE.EQ.2) CALL DTNEG2(SS,LOA,NEQS,KPVT,DET,EXP,NNEG)

10 K=NNEG
DETOLD=DET
EXPOLD=EXP
CALL STORE(LOAD,DISTR,LOAD,DO)
IF(ISOLVE.EQ.1) CALL SOLVE1
IF(ISOLVE.EQ.2) CALL SOLVE2
CALL INIT
CALL RWC
CALL DEFORM
CALL STIFF
IF(ISOLVE.EQ.1) CALL SOLVE1
IF(ISOLVE.EQ.2) CALL SOLVE2
IF(ISOLVE.EQ.1) CALL DTNEG1
IF(ISOLVE.EQ.2) CALL DTNEG2(SS,LOA,NEQS,KPVT,DET,EXP,NNEG)
C CHECK TO SEE IF THE CRITICAL POINT WAS PASSED.
IF(K.EQ.NNEG) GO TO 10
C CHECK TO SEE IF THE CRITICAL POINT IS FOUND.
15 ALPH=DABS(DET/(DETO*10**(EXPO-EXP))
IF(ALPH.LT.EPSLN3) GO TO 30
C CHECK TO SEE IF THE MAXIMUM NUMBER OF ITERATIONS
C HAS BEEN REACHED.
IF(KOUNT.LT.IMAX) GO TO 20
WRITE(2,1001)
1001 FORMAT(//' Trace of program path',/)
WRITE(2,1002) (ITRACE(I),I=1,K15)
1002 FORMAT(1X,10A2,5X,10A2,5X,10A2,5X,10A2,5X,10A2)
WRITE(2,11)
11 FORMAT(//' **** iteration limit exceeded to locate',
1 ' the critical point ****')
STOP
20 KOUNT=KOUNT+1
PREDICT THE NEXT DS THAT WILL GIVE THE CRITICAL POINT.

IT IS ASSUMED THAT EXPOLD=EXP.

DS=-(DET*DS)/(DET-DETOLD)

DETOLD=DET

EXPOLD=EXP

CALL STORE(LOAD, DISTR, LOAD, 0.0)

IF(ISOLVE.EQ.1) CALL SOLVE1

IF(ISOLVE.EQ.2) CALL SOLVE2

CALL INIT

CALL RWC

CALL DEFORM

CALL STIFF

IF(ISOLVE.EQ.1) CALL SOLVE1

IF(ISOLVE.EQ.2) CALL SOLVE2

IF(ISOLVE.EQ.1) CALL DTNEG1

IF(ISOLVE.EQ.2) CALL DTNEG2(SSID, LDA, NEQS, KPVT, DET, EXP, NNEG)

WRITE(4,1) KGUNT, LAMDA

1 FORMAT(I3, ' *** LAMDA = ', G24.16, ', 79(* *)')

WRITE(4,2)

2 FORMAT(' DISPLACEMENT')

WRITE(4,3) (Q(I), I=1, NEQS)

3 FORMAT(5G24.16)

WRITE(4,32) NNEG, EXP, DET

32 FORMAT(' NUMBER OF NEGATIVE EIGENVALUES = ', I3, ')

WRITE(4,33) DS, DLAMD

33 FORMAT(' DS = ', G24.16, ', DLAMD = ', G24.16)

WRITE(4,34) L

34 FORMAT(' NUMBER OF ITERATIONS = ', I3)

GO TO 15

$$$ T $$$

30 K15=K15+1
ITRACE(K15)='I'
SCALE=ISCALE
WRITE(2,31) KCOUNT
31 FORMAT(///,' NUMBER OF ITERATIONS FOR BIF. POINT =',I3)
C STOP EXECUTION IF ONLY THE CRITICAL POINT WAS WANTED.
IF(BPOINT.EQ.2) RETURN
   WRITE(2,1001)
   WRITE(2,1002) (ITRACE(I),I=1,K15)
   WRITE(2,1003)
1003 FORMAT(///,' BIFURCATION POINT FOUND')
STOP
SUBROUTINE CONVG
IMPLICIT REAL*8 (A-H,O-Z)
INTEGER SCALE, SYSTEM, UPDATE, CONT, ENDLIM, ENDSTB, BRANCH,
1 EXP, EXP0, EXPOLD, EXPNEW, BPOINT
REAL*8 LOAD, LAMDA, LAMDA1
CHARACTER*2 ITRACE
COMMON /BLOCK1/ SS(70,70), DQ(70), DQ1(70), DQI(70), DISTR(70),
1 LOAD(70), Q(70), R(70), QI(70), R1(70), KPVT(70)
COMMON /BLOCK2/ A(70), CG1(70), CG2(70), CG3(70), E(70), XL(70), XLD(70)
1 MCODE(70,6), MINC(70,2)
COMMON /BLOCK3/ X(31), Y(31), Z(31), C1(31), C2(31), JCODE(31,3)
COMMON /BLOCK4/ D(6), DLAMD, DLAMD1, DS, ELONG, EPSLN1, EPSLN2, EPSLN3,
1 DETO, DET, DETOLD, DETNEW, LAMDA, LAMDA1, DSBR,
2 IHBW, JFLAG, JHBW, L, IFLAG, LMAX, ITRACE(2000),
3 NCOUNT, NEQS, NJ, NM, NNEG, NPOINT, SCALE, SYSTEM, K15,
4 UPDATE, CONT, ENDLIM, ENDSTB, INTEST, METHOD, BRANCH,
5 EXP, EXP0, EXPOLD, EXPNEW, IMAX, JMAX, NPOINT, NDIGIT,
6 IHELP, BPOINT, LFLAG, ICOORD, NUMBER, LDA, ISOLVE
IFLAG=2
C CONVERGENCE OCCURS IF THE NORM OF THE REIDUAL FORCES IS LESS
C THAN EPSLN1 AND IF THE NORM OF THE DISPLACEMENT VECTOR IS LESS
C THAN EPSLN2.
IF(DSQR(FIR,R,R,0).LT.EPSLN1 .AND.  
1 DSQR(F(DQ,DQ,R,0)).LT.EPSLN2) IFLAG=1
RETURN
SUBROUTINE DATA
IMPLICIT REAL*8 (A-H,O-Z)
INTEGER SCALE,SYSTEM,UPDATE,CONT,ENULIM,ENDSTB,BRANCH,
EXP,EXPOLD,EXPNEW,BPOINT
REAL*8 LOAD,LAMDA,LAMDA1
CHARACTER*2 ITRACE
COMMON /BLOCK1/ SS(70,70),DQ(70),DQI(70),DQ1(70),DISTR(70),
1 LOAD(70),Q(70),R(70),Q1(70),R1(70),KPVT(70)
COMMON /BLOCK2/ A(70),CG1(70),CG2(70),CG3(70),E(70),XL(70),XLD(70)
1 MCODE(70,61),MINC(70,2)
COMMON /BLOCK3/ X(31),Y(31),Z(31),C(31),C2(31),JCODE(31,3)
COMMON /BLOCK4/ O(6),DLAMD,DLAMD1,DS,ELONG,EPSLN1,EPSLN2,EPSLN3,
1 DET,DET,DET,LAMDA,LAMDA1,DSBR,
2 IHBW,IFLAG,JHBW,L,IFLAG,LHAT,LMAX,ITRACE(2000),
3 NCOUNT,NEQS,NM,NNEG,NPOINT,SCALE,SYSTEM,K,15,
4 UPDATE,CONT,ENDLIM,ENDSTB,INTEST,METHOO,BRANCH,
5 EXP,EXPOLD,EXPNEW,IMAX,JMAX,MPOINT,NDIGIT,
6 IHELP,BPOINT,LFLAG,ICOORD,NUMBER,LDA,ISOLVE

INPUT PROGRAM CONTROL VARIABLES
READ(1,*)NM,NJ,SYSTEM,UPDATE,IHBW,SCALE,LHAT,LMAX,NPOINT,METHOO,
1 CONT,ENDLIM,ENDSTB,INTEST,BRANCH,IMAX,JMAX,MPOINT,IHELP,
2 BPOINT,ICOORD,ISOLVE
WRITE(2,1)NM,NJ,SYSTEM,UPDATE,IHBW,SCALE,LHAT,LMAX,NPOINT,METHOO,
1 CONT,ENDLIM,ENDSTB,INTEST,BRANCH,IMAX,JMAX,MPOINT,
2 IHELP,BPOINT,ICOORD,ISOLVE
1 FORMAT(* NM =',15,' NJ =',15,' SYSTEM =',15,' UPDATE =',15,' IHBW =',15,
1 SCALE =',15,' LHAT =',15,' LMAX =',15,' NPOINT =',15,
2 METHOD =',15,' CONT =',15,' ENDLIM =',15,'
INTEST = $1,15,/$
IMAX = $1,15,/$
IHELP = $1,15,/$
ISOLVE = $1,15,/$

INPUT THE SOLUTION TOLERANCES, THE INITIAL DELTA LAMDA VALUE,
AND THE NUMBER OF NEGATIVE EIGENVALUES.
READ(1,*) EPSLN1, EPSLN2, EPSLN3, NDIGIT, DS, DLAMD, LAMDA, NNEG, DET0,
EXP0, DSBR
WRITE(2,2) EPSLN1, EPSLN2, EPSLN3, NDIGIT, DS, DLAMD, LAMDA, NNEG, DET0,
EXP0, DSBR
2 FORMAT(' EPSLN1 = $1,24.16,/$ EPSLN2 = $1,24.16,/$
EPSLN3 = $1,24.16,/$
NDIGIT = $1,15,/$
DS = $1,24.16,/$
DLAMD = $1,24.16,/$
LAMDA = $1,24.16,/$
NNEG = $1,15,/$
DETO = $1,24.16,/$
EXP0 = $1,15,/$
DSBR = $1,24.16,/$

READ IN THE JOINT COORDINATES.
READ(1,*) (X(I), Y(I), Z(I), I=1, NJ)
IF(SYSTEM.EQ.1) WRITE(2,12)
12 FORMAT(/,' JOINT', 6X, 'R', 7X, 'THETA', 6X, 'PHI')
IF(SYSTEM.EQ.2) WRITE(2,13)
13 FORMAT(/,' JOINT', 6X, 'X', 9X, 'Y', 9X, 'Z')
WRITE(2,4) (1, X(I), Y(I), Z(I), I=1, NJ)
4 FORMAT(14,2X, 3F10.4)

SET JC0DE EQUAL TO ZERO.
DO 95 I=1, NJ
   DO 85 J=1,3
      JCODE(I,J) = 0
85   CONTINUE
95 CONTINUE

READ IN THE CONSTRAINT DATA.
WRITE(2,86)
READ(1,*) JNUM, JDIR
   IF(JNUM.EQ.0) GO TO 25
   WRITE(2,87) JNUM, JDIR
87   FORMAT(I6,I11)
   IF(JDIR.EQ.4) GO TO 27
   JCODE(JNUM, JDIR) = 1
   GO TO 15
27   JCODE(JNUM, 1) = 1
   JCODE(JNUM, 2) = 1
   JCODE(JNUM, 3) = 1
   GO TO 15
C (IF SPHERICAL COORDINATES CONVERT TO CARTESIAN COORDINATES.
CONVERT DEGREE ANGLES TO RADIAN ANGLES)
25 IF(SYSTEM.NE.1) GO TO 35
   DO 70 I=1,NJ
      RADIUS=X(I)
      THETA=Y(I)*.1745329251994330D-01
      PHI=Z(I)*.1745329251994330D-01
      X(I) = RADIUS*DSIN(PHI)*DCOS(THETA)
      Y(I) = RADIUS*DSIN(PHI)*DSIN(THETA)
      Z(I) = -1.DO*RADIUS*DCOS(PHI)
70   CONTINUE
C READ IN THE MEMBER PROPERTIES.
35 WRITE(2,5)
5   FORMAT(//,2X,'ELEMENT',9X,'A',9X,'E',5X,'A-END',5X,'B-END')
   READ(1,*) A(1), E(1), MINC(1,1), MINC(1,2)
   IF(A(1).GT.0) GO TO 120
   DO 110 I=1,NM
      A(I) = 1.DO
      E(I) = 1.DO
110   CONTINUE
READ(1,*) (MINC(I,1),MINC(I,2),I=2,NM)
GO TO 130
120 READ(1,*) (A(I),E(I),MINC(I,1),MINC(I,2),I=2,NM)
C
COMPUTE JCODE AND NEQS.
130 K=0
DO 30 I=1,NJ
  DO 20 J=1,3
    IF(JCODE(I,J).EQ.0) GO TO 10
    JCODE(I,J)=0
    GO TO 20
 10    K=K+1
    JCODE(I,J)=K
20   CONTINUE
30 CONTINUE
NEQS=K
WRITE(7,101) NEQS
101 FORMAT(I4)
JHBW=0
C
CALCULATE MCODE, XL, AND THE REQUIRED HALF-BAND WIDTH, JHBW.
DO 80 I=1,NM
  JA=MINC(I,1)
  JB=MINC(I,2)
  XDIFF=X(JB)-X(JA)
  YDIFF=Y(JB)-Y(JA)
  ZDIFF=Z(JB)-Z(JA)
  XL(I)=DSQRT(XDIFF*XDIF**F+YDIFF*YDIFF+ZDIFF*ZDIFF)
  MIN=1000
  MAX=0
  DO 90 J=1,3
    MCODE(I,J)=JCODE(JA,J)
    MCODE(I,J+3)=JCODE(JB,J)
    IF(JCODE(JA,J).LT.MIN.AND.JCODE(JA,J).NE.0) MIN=JCODE(JA,J)
IF(JCODE(JB,J) .LT. MIN .AND. JCODE(JB,J) .NE. 0) MIN = JCODE(JB,J)
IF(JCODE(JA,J) .GT. MAX) MAX = JCODE(JA,J)
IF(JCODE(JB,J) .GT. MAX) MAX = JCODE(JB,J)

90 CONTINUE
IF(JHBW .LT. MAX - MIN + 1) JHBW = MAX - MIN + 1
WRITE(2, 6) I, A(I), E(I), MINC(1, 1), MINC(1, 2)
6 FORMAT(I6, 5X, 2F10.2, 16, 110)
80 CONTINUE
WRITE(2, 11) NEQS
11 FORMAT(/, " NUMBER OF EQUATIONS = " , I4)
C CHECK TO SEE IF THE SPECIFIED HALF-BAND WIDTH IS TOO SMALL.
IF(IHBW .GE. JHBW) GO TO 100
WRITE(2, 8)
8 FORMAT(/, " **** HALF-BAND WIDTH IS TOO SMALL *****")
STOP
100 IHBW = JHBW
WRITE(2, 9) IHBW
9 FORMAT(/, " HALF-BAND WIDTH = " , I4)
WRITE(2, 7)
7 FORMAT(/, " JOINT NUMBER JOINT DIRECTION FORCE")
C INITIALIZE THE DISTRIBUTION VECTOR, DISPLACEMENT VECTOR, AND
C JOINT DIRECTION COSINES TO ZERO.
DO 40 I = 1, NEQS
   DISTR(I) = 0.0000
   Q(I) = 0.0000
40 CONTINUE
DO 45 I = 1, NJ
   C1(I) = 0.00
   C2(I) = 0.00
45 CONTINUE
C INPUT THE CONSTANT LOAD DISTRIBUTION VECTOR AND THE INITIAL
C DISPLACEMENT VECTOR.
50 READ(1,*) JNUM, JDIR, P1
C CHECK TO SEE IF THIS IS THE END OF THE LOAD DATA.
   IF(JNUM.LT.0) GO TO 60
C CHECK TO SEE IF THE DISPLACEMENT VECTOR IS TO BE INPUT.
   IF(JNUM.EQ.0) GO TO 200
   WRITE(2,18) JNUM, JDIR, P1
18   FORMAT(I8, I15, G23.10)
   K=JCODE(JNUM, JDIR)
C CHECK TO SEE IF THERE IS A DATA ERROR.
   IF(K.NE.0) GO TO 55
   WRITE(2,51)
51   FORMAT(/, ' DATA ERROR IN FORCE INPUT',/)
   STOP
55   DISTRI(K)=P1
   GO TO 50
200 WRITE(2,21)
21   FORMAT(/, ' DISPLACEMENT VECTOR',/) 
   READ(1,*) (Q(I), I=1, NEQS)
   WRITE(2,22) (Q(I), I=1, NEQS)
22   FORMAT(5G24.16)
C STOP IF THE PROGRAM IS RUN FOR A DATA CHECK.
60 IF(INTEST.NE.1) RETURN
   WRITE(2,19)
19   FORMAT(/, ' ***** PROGRAM IS STOPPED FOR DATA TEST *****')
   STOP
C*************************************************************************
C* DEFORM                                           *
C*************************************************************************
SUBROUTINE DEFORM
IMPLICIT REAL*8 IA-H,G-ZJ
INTEGER SCALE,SYSTEM,UPDATE,CONT,ENDLAB,ENDSTB,BRANCH,
1 EXP,EXPO,EXPOLD,EXPNEW,BPOINT
REAL*8 LOAD,LAMDA,LAMDA1
CHARACTER*2 ITRACE
COMMON /BLOCK1/ SS(70,70),DQ(70),DQ1(70),DQ1(70),DSTR(70),
1 LOAD(70),Q(70),R(70),Q1(70),R1(70),KPVT(70)
COMMON /BLOCK2/ A(70),CG1(70),CG2(70),CG3(70),E(70),XL(70),XLD(70)
1 ,MCODE(70,6),MINC(70,2)
COMMON /BLOCK3/ X(31),Y(31),Z(31),C1(31),C2(31),JCODE(31,3)
COMMON /BLOCK4/ D(6),DLAMD,DLAMD1,DS,ELONG,EPSLN1,EPSLN2,EPSLN3,
1 DETO,DET,GETOLD,GETNEW,LAMDA,LAMDA1,USBR,
2 IHBW,JFLAG,JHBW,L,IFLAG,LHAT,LMAX,ITRACE(2000),
3 NCOUNT,NEQS,NJ,NM,NNEG,NPOINT,SCALE,SYSTEM,K15,
4 UPDATE,CONT,ENDLAB,ENDSTB,INTEST,METHOD,BRANCH,
5 EXP,EXPO,EXPOLD,EXPNEW,JMAX,MPPOINT,NDIGIT,
6 IHELP,BPOINT,LFLAG,ICOORO,NUMBER,LOA,ISOLVE
C CALCULATE THE DEFORMED LENGTHS AND COSINES.
DO 30 I=1,NM
  DO 20 J=1,6
    IF(MCODE(I,J).EQ.0) GO TO 10
    N=MCODE(I,J)
    D(J)=Q(N)
    GO TO 20
  10   D(J)=0.0000
  20 CONTINUE
  JA=MINC(I,1)
  JB=MINC(I,2)
DO 30 J=1,6
  IF(MCODE(I,J).EQ.0) GO TO 10
  N=MCODE(I,J)
  D(J)=Q(N)
  GO TO 20
  10   D(J)=0.0000
  20 CONTINUE
  JA=MINC(I,1)
  JB=MINC(I,2)
DO 30 J=1,6
  IF(MCODE(I,J).EQ.0) GO TO 10
  N=MCODE(I,J)
  D(J)=Q(N)
  GO TO 20
  10   D(J)=0.0000
  20 CONTINUE
  JA=MINC(I,1)
  JB=MINC(I,2)
D63 = D(6) - D(3)
GOTO (21, 22), ICOORD

21
D41 = D(4) - D(1)
D52 = D(5) - D(2)
GOTO 23

22
C1I = C1(JA)
C2I = C2(JA)
C1H1 = C1(JB)
C2H1 = C2(JB)
D41 = C1H1 * D(4) - C2H1 * D(5) - C1I * D(1) + C2I * D(2)
D52 = C2H1 * D(4) + C1H1 * D(5) - C2I * D(1) + C1I * D(2)

23
XLD1 = X(JB) - X(JA) + D41
XLD2 = Y(JB) - Y(JA) + D52
XLD3 = Z(JB) - Z(JA) + D63
XLD(I) = DSQRT(XLD1 * XLD1 + XLD2 * XLD2 + XLD3 * XLD3)
CG1(I) = XLD1 / XLD(I)
CG2(I) = XLD2 / XLD(I)
CG3(I) = XLD3 / XLD(I)

30 CONTINUE
IF (ICOORD.EQ.1 OR NUMBER.EQ.2) RETURN
DO 40 I = 1, NJ
RAD = DSQRT(X(I)**2 + Y(I)**2)
IF (RAD .NE. 0. DO) GOTO 50
C1(I) = 1.00
C2(I) = 0.00
GOTO 40

50
C1(I) = X(I) / RAD
C2(I) = Y(I) / RAD

40 CONTINUE
NUMBER = 2
RETURN
SUBROUTINE DTNEGl
IMPLICIT REAL*8 (A-H,O-Z)
INTEGER SCALE, SYSTEM, UPDATE, CONT, ENDLIM, ENDSTB, BRANCH,
1 EXP, EXP0, EXPOLD, EXPNEW, BPOINT
REAL*8 LOAD, LAMDA, LAMDA1
CHARACTER*2 ITRACE
COMMON /BLOCK1/ SS(70,70), DQ(70), DQ1(70), DQ1(70), DISTR(70),
1 LOAD(70), Q(70), R(70), Q1(70), R1(70), KPVI(70)
COMMON /BLOCK2/ A(70), CG1(70), CG2(70), CG3(70), E(70), XL(70), XLD(70)
1 COMMON /BLOCK3/ X(31), Y(31), Z(31), C1(31), C2(31), JCODE(31, 3)
COMMON /BLOCK4/ D(6), DLAMD, DLAML, DS, ELOHI, EPSN1, EPSLN2, EPSLN3,
1 CETO, DET, DETOLO, DETNEW, LAMDA, LAMDA1, OSBR,
2 IHBW, JFLAG, JHBW, L, IFLAG, LMAX, ITRACE(2000),
3 NCOUNT, NEQS, NJ, NN, NNEG, NPOINT, SCALE, SYSTEM, K15,
4 UPDATE, CONT, ENDLIM, ENDSTB, IMEST, METHOD, BRANCH,
5 EXP, EXP0, EXPOLD, EXPNEW, IMAX, JMAX, MPOINT, NDIGI,
6 IHELP, BPOINT, LFLAG, ICOORD, NUMBER, LDA, LSAVE
NNEG=0
EXP=0
DET=1.00
DO 20 I=1, NEQS
1 IF(SS(I,1), LT=0.0) NNEG=NNEG+1
DET=DET*SS(I,1)
IF(DABS(DET), LT=1.030) GO TO 10
EXP=EXP-30
DET=DET*1.0-30
GO TO 20
10 IF(DABS(DET), GT=1.0-30) GO TO 20
EXP=EXP+30
DET=DET*1.D30
20 CONTINUE
RETURN
SUBROUTINE DTNEG2(SS, LDA, NEQS, KPVT, DET, EXP, NNEG)
IMPLICIT REAL*8 (A-H, C-Z)
REAL*8 SS(LDA, NEQS)
INTEGER KPVT(1), EXP
NNEG = 0
DET = 1.00
EXP = 0
TEN = 10.00
T = 0.00
DO 130 K = 1, NEQS
    D = SS(K, K)
    IF (KPVT(K) .GT. 0) GO TO 50
    IF (T .NE. 0.00) GO TO 30
    T = DABS(SS(K, K + 1))
    D = (D/T) * SS(K + 1, K + 1) - T
    GO TO 40
30 CONTINUE
    D = T
    T = 0.00
40 CONTINUE
50 CONTINUE
    IF (D .LT. 0.00) NNEG = NNEG + 1
    DET = D * DET
    IF (DET .EQ. 0.00) GO TO 110
70 IF (DABS(DET) .GE. 1.00) GO TO 80
    DET = TEN * DET
    EXP = EXP - 1
    GO TO 70
80 CONTINUE
90 IF(DABS(DET).LT.TEN)GOTO 100
   DET=DET/TEN
   EXP=EXP+1
   GOTO 90
100 CONTINUE
110 CONTINUE
130 CONTINUE
   EXP=-EXP
   RETURN
SUBROUTINE EIGEN
IMPLICIT REAL*8 (A-H,O-Z)
INTEGER SCALE, SYSTEM, UPDATE, CONT, ENDLIM, ENDSTB, BRANCH,
EXP, EXPD, EXPOLD, EXPNEW, BPOINT
REAL*8 LOAD, LAMDA, LAMDA1
CHARACTER*2 ITRACE
COMMON /BLOCK1/ SS(70,70), DQ(70), DQ1(70), DQ1(70), DISTR(70),
LOAD(70), Q(70), R(70), Q1(70), R1(70), KPVT(70)
COMMON /BLOCK2/ A(70), CG1(70), CG2(70), CG3(70), E(70), XL(70), XLD(70)
COMMON /BLOCK3/ X(31), Y(31), Z(31), C1(31), C2(31), JCODE(31, 3)
COMMON /BLOCK4/ D(6), DLAMD, DLAMD1, OS, ELONG, EPSLN1, EPSLN2, EPSLN3,
DET, DET, DETOLD, DETNEW, LAMDA, LAMDA1, DSR,
IHBL, JFLAG, JHBWL, IFLAG, LMAX, ITRACE(2000),
NCOUNT, NEQS, NJ, NM, NNEG, NPOINT, SCALE, SYSTEM, K15,
UPDATE, CONT, ENDLIM, ENDSTB, INTEST, METHOD, BRANCH,
EXP, EXPD, EXPOLD, EXPNEW, JMAX, MPOINJ, NDIGIT,
IHELP, BPOINT, LFLAG, ICOORD, NUMBER, LOA, ISOLVE
DIMENSION XTEMP(70)
C
K15=K15+1
ITRACE(K15)="J"
K=NNEG
IF(ISOLVE.EQ.1) CALL DTNEG1
IF(ISOLVE.EQ.2) CALL DTNEG2(SS, LOA, NEQS, KPVT, DET, EXP, NNEG)
WRITE(3,11) NNEG, EXP, DET
11 FORMAT('NUMBER OF NEGATIVE EIGENVALUES =', I3, '/', 'DETERMINANT ',
1 ' 1.0', I3, ' =', G24.16, '/)
C
RETURN IF BRANCHING IS TO OCCUR.
IF(BRANCH.EQ.NCOUNT-1) RETURN
C RETURN IF THERE ARE (AND WERE) NO NEGATIVE EIGENVALUES.
   IF(K.EQ.0.AND.NNEG.EQ.0) RETURN
C $$$ K $$$
    K15=K15+1
    ITRACE(K15)='K'
C CHECK TO SEE IF THE PROGRAM WAS RUN WITH ENDLIM=1.
C IF SO, BRANCH IF THERE IS (ARE) NEGATIVE EIGENVALUE(S).
   90 IF(NNEG.LE.0.OR.ENDLIM.EQ.2) GO TO 30
C $$$ L $$$
    K15=K15+1
    ITRACE(K15)='L'
C STOP IF NO MURE POINTS ARE WANTED.
   IF(NPOINT.GT.0) GO TO 20
    WRITE(2,1001)
1001   FORMAT(//'TRACE OF PROGRAM PATH',//)
    WRITE(2,1002) (ITRACE(I),I=1,K15)
1002   FORMAT(1X,10A2,5X,10A2,5X,10A2,5X,10A2,5X,10A2)
    WRITE(2,4) NCOUNT-1
4   FORMAT(//'NUMBER OF POINTS =',I3,//,'***** PROGRAM IS',//
       'STOPPED AT OR PAST A LIMIT OR BIFURCATION',//
       'POINT AS WAS REQUESTED *****')
   STOP
20   NPOINT=NPOINT-1
C RETURN IF THE NUMBER OF NEGATIVE EIGENVALUES IS THE SAME.
   30 IF(K.EQ.NNEG) RETURN
C CHECK TO SEE IF THE TRUSS IS BECOMING STABLE.
   IF(K.LT.NNEG) GO TO 50
C $$$ M $$$
    K15=K15+1
    ITRACE(K15)='M'
    WRITE(2,1) NCOUNT-1
1 FORMAT(' //', ' AT POINT', ' , I3, ' , THE TRUSS IS BECOMING MORE ',
1 ' STABLE' )  
C  STOP IF THE TRUSS IS STABLE AND CONT=2.  
   IF ( INEG . NE. 0 . OR . CONT . EQ . 1 ) GO TO 60  
     WRITE(2,1001)  
     WRITE(2,1002) ( ITRACE(I) , I=1,K15 )  
     WRITE(2,3) NCOUNT-1  
3   FORMAT(' //', ' ***** PROGRAM IS STOPPED AT POINT', ' , I3, ' , A ',
1 ' STABLE POINT *****')  
   STOP  
C  AT THIS POINT, A LIMIT OR BIFURCATION POINT WAS ENCOUNTERED.  
50 WRITE(2,2) NCOUNT-2,NCOUNT-1  
   2   FORMAT(' //', ' A LIMIT POINT OR BIFURCATION POINT WAS ENCOUNTERED',
1 ' BETWEEN POINT', ' , I3, ' AND POINT', ' , I3, '.', ')  
C  CHECK TO SEE IF THE CRITICAL POINT IS TO BE LOCATED.  
60 IF ( BPOINT . NE. 3 ) GO TO 55  
   LFLAG=1  
   RETURN  
C  $$$$ N $$$$  
55 K15=K15+1  
   ITRACE(K15)='N'  
   LFLAG=2  
C  RETURN IF THE NEWTON-RAPHSON METHOD IS USED.  
   IF ( METHOD . EQ . 3 ) RETURN  
   IF ( NCOUNT . NE . 1 ) GO TO 61  
      CALL STORE(LOAD, DISTR, LOAD, 0.00)  
      IF ( ISOLVE . EQ . 1 ) CALL SOLVE1  
      IF ( ISOLVE . EQ . 2 ) CALL SOLVE2  
      DS=DLAM0*DSQRT(F(LOAD,LOAD,LOAD,0)+1.00)  
C  $$$$ O $$$$  
61 K15=K15+1  
   ITRACE(K15)='O'
IF(IHELP.EQ.2) GO TO 505
IF(IHELP.EQ.3) RETURN
IHELP=2
RETURN
C RECORD THE DISPLACEMENT VECTOR AND THE LOAD LEVEL.
C 505 CALL STORE(XTEMP,Q,Q,0.DO)
XLAMDA=LAMDA
CALL STORE(LOAD,DISTR,LOAD,0.DO)
C SOLVE K DQKI = DISTR.
IF(ISOLVE.EQ.1)CALL SOLVE1
IF(ISOLVE.EQ.2)CALL SOLVE2
C ASSUME DS = -DS.
DS=-1.DO*DS
C PREVENT SCALING AND ENLARGE DS TO ENSURE THAT THE ASSUMPTION
C THAT DS=-DS WILL PROVE WRONG IF IT IS WRONG (THE PROCEDURE MAY
C 'BACK' ON ITSELF).
ISCALE=SCALE
C PREVENT SCALING SO THAT DS REMAINS UNCHANGED.
SCALE=2
DS=1.100*DS
C GET THE TANGENT VECTOR.
CALL INIT
C RETAIN THE VARIABLES AS THEY WERE.
SCALE=ISCALE
CALL RWC
CALL DEFORM
CALL STIFF
IF(ISOLVE.EQ.1)CALL SOLVE1
IF(ISOLVE.EQ.2)CALL SOLVE2
C CALCULATE THE NUMBER OF NEGATIVE EIGENVALUES.
K=NNEG
IF(ISOLVE.EQ.1)CALL DTNEGI
IF(ISOLVE.EQ.2)CALL DTNEG2(SS,LDA,NEQS,KPVT,DET,EXP,NNEG)
C CHECK TO SEE IF THE NUMBER OF EIGENVALUES HAS DECREASED.
IF(K.LE.NNEG) GO TO 80
C $$$ P $$$
   K15=K15+1
   ITRACE(K15)=P
C THE ASSUMPTION THAT DS = -DS WAS WRONG. RETURN TO THE
C MAINLINE.
   CALL STORE(Q,XTEMP,Q,O.OO)
   LAMDA=XLAMDA
   DS=-1.DO*DS
   NNEG=K
   RETURN
C WRITE OUT THE POINT DATA.
80 WRITE(3,9) NCOUNT,LAMDA
   9 FORMAT(13,' LAMDA = ',G24.16,' * ',83('**'))
   WRITE(3,5)
   5 FORMAT(' DISPLACEMENT')
   WRITE(3,6) (Q(I),I=1,NEQS)
   WRITE(3,17) L
   17 FORMAT(' NUMBER OF ITERATIONS = ',13)
   6 FORMAT(5G24.16)
   WRITE(3,12) DS
   12 FORMAT(' DS = ',G24.16)
   DS=DS/1.100
   WRITE(3,11) NNEG,EXP,DET
   WRITE(7,42) LAMDA
   42 FORMAT(15.9)
   WRITE(7,41) (Q(I),I=1,NEQS)
   41 FORMAT(13F10.6)
   NCOUNT=NCOUNT+1
C $$$ Q $$$
K15=K15+1
ITRACE(K15)=Q'
RETURN IF BRANCHING IS TO OCCUR.
IF(BRANCH.EQ.NCOUNT-1) RETURN

C CONTINUE IF 1) A STABLE POINT IS WANTED (ENDSTD=1)
C OR 2) A LIMIT OR BIFURCATION POINT IS WANTED
C (ENDLIM=1)
C OR 3) NCOUNT=NPOINT, ENDLIM=2, AND ENDSB=2.
C IF(ENDLIM.EQ.1.OF.ENDSB.EQ.1.0R.(NCOUNT-1.NE.NPOINT.AND.
1 1 ENDSTB.NE.1.AND.ENDLIM.NE.1)) GO TO 90
WRITE(2,1001)
WRITE(2,1002) (ITRACE(I),I=1,K15)
WRITE(2,7) NCOUNT-1

7 FORMAT(//'NUMBER OF POINTS =',13,//,' **** normal ','
1 'COMPLETION ****')
STOP
C*************************************************************~******
C*
C*************************************************************~******
FUNCTION f(Xl,Yl,Zl,K)
IMPLICIT REAL*8 (A-H,O-Z)
INTEGER SCALE,SYSTEM,UPDATE,CONT,ENDLIM,ENDSTB,BRANCH,
1 EXP,EXPO,EXPOLD,EXPNEW,BPOINT
REAL*8 LOAD,LAMDA,LAMDA1
CHARACTER*2 !TRACE
COMMON /BLOCK1/ SS(70,70),DQ(70),DQI(70),DQ1(70),DISTR(70),
1 LOAD(70),Q(70),R(70),Q1(70),R1(70),KPV1(70)
COMMON /BLOCK2/ A(70),CG1(70),CG2(70),CG3(70),E(70),XL(70),XLD(70)
1 MCODE(70,6),MINC(70,2)
COMMON /BLOCK3/ X(31),Y(31),Z(31),C1(31),C2(31),JCODE(31,3)
COMMON /BLOCK4/ D(6),DLAMD,DLAMD1,DS,ELONG,EPslN1,EPslN2,EPslN3,
1 DET0,DET,DETOLD,DETNEW,LAMDA,LAMD1,DSBK,
2 IHBW,JFLAG,JHBW,L,IFLAG,LHAT,LMAX,ITRACE(2000),
3 NCOUNT,NEWS,NJ,NM,NNEG,NPOINT,SCALE,SYSTEM,K15,
4 UPDATE,CONT,ENDLIM,ENDSTB,INTEST,METHOD,BRANCH,
5 EXP,EXPO,EXPOLD,EXPNEW,MAX,JMAX,MPOINT,NDIGIT,
6 IHELP,BPOINT,LFLAG,ICOORD,NUMBER,LDA,ISOLVE
DIMENSION Xl(NEQS),Yl(NEQS),Zl(NEQS)
C F IS THE DOT PRODUCT OF Xl ( Y1 + K * Zl)
F=0.DO
IF(K.EQ.0) GO TO 20
DO 10 I=1,NEQS
  F=F+(Xl(I)*(Yl(I)+DFLOAT(K)*Zl(I)))
10  CONTINUE
RETURN
20 DO 30 I=1,NEQS
  F=F+Xl(I)*Yl(I)
30 CONTINUE
C***************************************************************~******
C*
C*****************************************************************************
SUBROUTINE INIT
IMPLICIT REAL*8 (A-H,O-Z)
INTEGER SCALE,SYSTEM,UPDATE,CONT,ENOLIM,ENDSTB,BRANCH,
1 EXP,EXPO,EXPNEW,BPOINT
REAL*8 LOAD,LAMDA,LAMDA1
CHARACTER*2 !TRACE
COMMON /BLOCK1/ SS(70,70),DQ(70),DQ1(70),DQ2(70),DQI(70),DISTR(70),
1 LOAD(70),Q(70),R(70),Q1(70),R1(70),KPVT(70)
COMMON /BLOCK2/ AI70),CG1(70),CG2(70),CG3(70),E(70),XL(70),XLD(70)
1 MCODE(70,6),MINS(70,2)
COMMON /BLOCK3/ X(31),Y(31),Z(31),Cl(31),C2(31),JCODE(31,3)
COMMON /BLOCK4/ D(6),DLAMD,DLAMD1,DS,ELONG,EPSTN1,EPSTN2,EPSTN3,
1 DETO,DET,DElicated,DEnew,LAMDA,LAMDA1,DSBR,
2 IHBW,IFLAG,JHBW,L,ITFAL,LHAT,LMAX,ITRACE(2000),
3 NCOUNT,NEQS,NJ,NM,NNEG,NPOINT,SCALE,SYSTEM,K15,
4 UPDATE,CONT,ENOLIM,ENDSTB,INTEST,METHOD,BRANCH,
5 EXP,EXPO,EXPNEW,IMP,LMAX,NPOINT,NDIGIT,
6 IHELP,BPOINT,LFLAG,ICOORD,NUMBER,LDA,ISOLVE
C CHECK TO SEE IF THIS IS THE FIRST ITERATION FOR A POINT.
IF(DS.EQ.0.0D0) GO TO 30
C FIND THE INCREMENT IN LOAD LEVEL GIVEN DS.
 IF(Scale.NE.1) GO TO 20
C SCALE THE TANGENT VECTOR.
   DS=DS*SQRT(DFLOAT(LHAT)/DFLOAT(L))
20   DLAMD=DS/DSQRT(F(LOAD,LOAD,LOAD,0)+1.0D0)
   GO TO 40
C FIND THE LENGTH IN THE TANGENT VECTOR, DS, GIVEN THE INITIAL
C INCREMENT IN LOAD LEVEL.
30   DS=DLAMD*DSQRT(F(LOAD,LOAD,LOAD,0)+1.0D0)
40 DO 50 I=1,NEQS
   C          FIND THE INITIAL DQ.
     DQ(I)=DLAMD*LOAD(I)
 50 CONTINUE
   C          SAVE THE INITIAL Q VECTOR AND LAMDA FOR THE SPHERE ITERATION.
     LAMDA1=LAMDA
     CALL STORE(Q1,Q,Q,0.DO)
   C          UPDATE THE VARIABLES FOR THE FIRST ITERATION.
     LAMDA=LAMDA+DLAMD
     CALL STORE(Q,Q,DQ,1.DO)
   RETURN
SUBROUTINE NEWTON
IMPLICIT REAL*8 (A-H,O-Z)
INTEGER SCALE,SYSTEM,UPDATE,CONT,ENDLIM,ENDSTB,BRANCH
REAL*8 LOAD,LOADA,LAMDA,LAMDA1
CHARACTER*2 ITRACE
COMMON /BLOCK1/ SS(70,70),DQ(70),DQ1(70),DQI70),DISTR(70),
LOAO,LAMOA,LAMDA1
COMMON /BLOCK2/ A(70),CG1(70),CG2(70),CG3(70),E(70),XL(70),XLD(70),
MCODE(70,6),MINC(70,2)
COMMON /BLOCK3/ X(31),Y(31),Z(31),C1(31),C2(31),JCODE(31,3)
COMMON /BLOCK4/ D(6),DLAMD,DLAMD1,DS,ELONG,EPCLN1,EPCLN2,EPCLN3,
DETO,DET,DETOLD,DETNEW,LAMDA,LAMDA1,DSBR,
IHBR,IFLAG,IFHBR,L,LHAT,LMAX,ITRACE(2000),
NCOUNT,NEQS,NJ,NM,NNEG,NPOINT,SCALE,SYSTEM,K15,
UPDATE,CONT,ENDLIM,ENDSTB,INTEST,METHOD,BRANCH,
EXP,EXP0,EXPO,EXPOLD,EXPNEW,IMAX,JMAX,MPPOINT,NDIGIT,
IHELP,BPOINT,LFLAG,ICOORD,NUMBER,LOA,ISOLVE

C$$$$ H $$$
K15=K15+1
ITRACE(K15)="H"

C UPDATE FOR THE FIRST ITERATION.
L=1
CALL DEFORM
CALL RESID
GO TO 50

C COMPUTE THE STIFFNESS MATRIX IF 1) IT IS THE FIRST ITERATION
C OR 2) UPDATING IS DESIRED.
20 IF(L.NE.1.AND.UPDATE.NE.1) GO TO 30
C
K15=K15+1
ITRACE(K15)='I'
C UPDATE THE STIFFNESS MATRIX.
CALL STIFF
IF(ISOLVE.EQ.1)CALL SOLVE1
IF(ISOLVE.EQ.2)CALL SOLVE2
30 CALL STORE(LOAD,R,R,O.DO)
C SOLVE K DQKI = R.
IF(ISOLVE.EQ.1)CALL SOLVE1
IF(ISOLVE.EQ.2)CALL SOLVE2
C UPDATE THE VARIABLES.
CALL STORE(DQ,LOAD,Q,Q,O.DO)
CALL STORE(Q,Q,DQ,1.0,DO)
L=L+1
CALL DEFORM
CALL RESID
C CHECK FOR CONVERGENCE.
50 CALL CONVG
IF(IFLAG.EQ.1) RETURN
C CHECK TO SEE IF THE MAXIMUM NUMBER OF ITERATIONS HAS BEEN
C REACHED.
IF(L.NE.LMAX) GO TO 20
IF(ISOLVE.EQ.1)CALL DTNEG1
IF(ISOLVE.EQ.2)CALL DTNEG2(SS, LDA, NEQS, KPVT, DET, EXP, NNEG)
WRITE(3,11) NNEG, EXP, DET
11 FORMAT(* NUMBER OF NEGATIVE EIGENVALUES =', I3, ',', DETERMIN*',
*ANT * 1.0', ',13, ' =', E24.16, ')
WRITE(2,1001)
1001 FORMAT(//,* TRACE OF PROGRAM PATH*/)
WRITE(2,1002) (ITRACE(I),I=1,K15)
1002 FORMAT(1X,10A2,5X,10A2,5X,10A2,5X,10A2,5X,10A2,5X)
WRITE(2,1) NCOUNT
1 FORMAT(///, 'NUMBER OF POINTS =',14,///, '***** ITERATION ',
1      'LIMIT EXCEEDED *****')
STOP
SUBROUTINE NORMAL
IMPLICIT REAL*8 (A-H,O-Z)
INTEGER SCALE, SYSTEM, UPDATE, CONT, ENDLIM, ENSTB, BRANCH,
1 EXP, EXP0, EXPOLD, EXPNEW, BPOINT
REAL*8 LOAD, LAMDA, LAMDA1
CHARACTER*2 ITRACE
COMMON /BLOCK1/ SS(70,70), DQ(70), DQ1(70), DQ2(70), DISTR(70),
1 LOAD(70), Q(70), R(70), Q1(70), R1(70), KPVT(70)
COMMON /BLOCK2/ A(70), CG(70), CG0(70), CG1(70), CG2(70), CG3(70), E(70), XL(70), XLD(70)
1 MCODE(70,6), MINC(70,2)
COMMON /BLOCK3/ X(31), Y(31), Z(31), C1(31), C2(31), JCODE(31,3)
COMMON /BLOCK4/ D(6), DLAM0, DLAM01, DS, ELONG, EPSLN1, EPSLN2, EPSLN3,
1 DET0, DET, DETOLD, DETNEW, LAMDA, LAMDA1, DSBR,
2 IHBW, JFLAG, JHBW, IFLAG, LMAX, LMAX, ITRACE(2000)
3 NCOUNT, NEQS, NJ, NM, NNEG, NPOINT, SCALE, SYSTEM, K15
4 UPDATE, CONT, ENDLIM, ENSTB, INTMETHGO, BRANCH,
5 EXP, EXP0, EXPOLD, EXPNEW, IMAX, JMAX, NPOINT, NODIGIT,
6 IHELP, BPOINT, LFLAG, LG0, NUMBER, LDA, ISOLVE
DLAMD=-F(DQ1,DQ,DQ,0)/[F(DQ1,DQ,DQ,0)+DLAM01)
CALL STORE(DQ,DQ,DQ,DLAM0)
C UPDATE THE VARIABLES
LAMDA=LAMDA+DLAMD
CALL STORE(Q,Q,DQ,1.0D0)
RETURN
SUBROUTINE POWINV

IMPLICIT REAL*8 (A-H, O-Z)
INTEGER SCALE, SYSTEM, UPDATE, CONT, ENOLIM, ENSTB, BRANCH,
1 EXP, EXPO, EXPOLD, EXPNEW, BPOINT
REAL*8 LOAD, LAMOA, LAMDAI
CHARACTER*2 ITRACE
COMMON /BLOCK1/ SS(70,70), DQ(70), DQ1(70), DQ1(70), DISTR(70),
1 LOAD(70), Q(70), R(70), Q1(70), R1(70), KPVT(70)
COMMON /BLOCK2/ A(70), CG1(70), CG2(70), CG3(70), E(70), XL(70), XLD(70)
1 MCODE(70,6), MINC(70,2)
COMMON /BLOCK3/ X(31), Y(31), Z(31), CI(31), C2(31), JCODE(31,3)
COMMON /BLOCK4/ D(6), DLAMD, DLAMDI, DS, ELUNG, EPSLN1, EPSLN2, EPSLN3,
1 DET, DET, DETOLD, DETNEW, LAMDA, LAMDAI, DSBR,
2 IHBW, JFLAG, JHBW, L, IFLAG, LHAT, LMAX, ITRACE(2000),
3 NCOUNT, NEQS, NJ, NM, NNEG, NPOINT, SCALE, SYSTEM, K15,
4 UPDATE, CONT, ENOLIM, ENSTB, INTEST, METHOD, BRANCH,
5 EXP, EXPO, EXPOLD, EXPNEW, IMAX, JMAX, MPOINT, NDIGIT,
6 IHELP, BPOINT, LFLAG, ICOORD, NUMBER, LDA, ISOLVE

C
$$$
K15=K15+1
ITRACE(K15)="U"
IF(ISOLVE.EQ.1)CALL SOLVE1
IF(ISOLVE.EQ.2)CALL SOLVE2
C THIS SUBROUTINE USES THE INVERSE POWER METHOD TO ESTIMATE THE
C EIGENVECTOR CORRESPONDING TO THE SMALLEST EIGENVALUE.
KOUNT=0
C CALCULATE THE INITIAL EIGENVALUE APPROXIMATION.
DO 10 I=1, NEQS
   LOAD(I)=1.00/DFLOAT(NEQS)
R1(I)=LOAD(I)
10 CONTINUE
   IF(ISOLVE.EQ.1)CALL SOLVE1
   IF(ISOLVE.EQ.2)CALL SOLVE2
   TEMP=F(LOAD,R1,R1,0)
   YSCALE=DSQRT(F(LOAD,LOAD,LOAD,0))
   ESTOLD=1.0/TEMP
C INVERSE POWER METHOD ITERATION WITH SCALING (ALPH=0).
15 KOUNT=KCOUNT+1
   DO 20 I=1,NEQS
      LOAD(I)=LOAD(I)/YSCALE
      R1(I)=LOAD(I)
20 CONTINUE
   IF(ISOLVE.EQ.1)CALL SOLVE1
   IF(ISOLVE.EQ.2)CALL SOLVE2
   TEMP=F(LOAD,R1,R1,0)
   YSCALE=DSQRT(F(LOAD,LOAD,LOAD,0))
   ESTNEW=1.0/TEMP
C SEE IF THE NUMBER OF ITERATIONS IS THE MAXIMUM ALLOWED.
   IF(KOUNT.GE.JMAX) GO TO 50
      ESTEMP=ESTNEW
201   IF(IABS(ESTNEW).GT.0.D0) GO TO 202
      ESTNEW=ESTNEW*10.D0
      ESTOLD=ESTOLD*10.D0
      GO TO 201
202   I1=IFIX(SNLG(ESTNEW*10**NDIGIT))
      I2=IFIX(SNLG(ESTOLD*10**NDIGIT))
      IF(I1.EQ.I2) GO TO 30
      ESTOLD=ESTEMP
      GO TO 15
30   WRITE(2,1) ESTEMP
1   FORMAT(/,' EIGENVALUE =',G24.16,/, 'EIGENVECTOR')
DO 40 I=1,NEQS
   R1(I)=LOAD(I)/YScale
40  CONTINUE
G10=0.00
WRITE(2,2) (R1(I),I=1,NEQS),G10
2  FORMAT(5G24.16)
WRITE(2,39) KOUNT
39  FORMAT(///,' NUMBER OF ITERATIONS TO FIND THE EIGENVECTOR =',
1    I3,///)
   RETURN
50 WRITE(2,1001)
1001 FORMAT(///,' TRACE OF PROGRAM PATH',//)
WRITE(2,1002) (ITRACE(I),I=1,K15)
1002 FORMAT(1X,10A2,5X,10A2,5X,10A2,5X,10A2,5X,10A2,5X)
WRITE(2,3)
3  FORMAT(///,' ITERATION LIMIT EXCEEDED TO FIND THE EIGENVECTOR')
   STOP
SUBROUTINE RESIO

IMPLICIT REAL*8 (A-H,O-Z)
INTEGER SCALE,SYSTEM,UPDATE,CONT,ENDLIN,ENDSTB,BRANCH,
EXP,EXPO,EXPOLD,EXPNW,BPOINT
REAL*8 LOAD,LAMDA,LAMDA1,
CHARACTER*2 ITRACE
COMMON /BLOCK1/ SS(70,70),DQ(70),DQI(70),QII(70),DISTR(70),
1 LOAD(70),QI(70),RI(70),QII(70),R1(70),KPVT(70)
COMMON /BLOCK2/ A(70),CG1(70),CG2(70),CG3(70),E(70),XL(70),XLD(70)
1 MCOOE(70),MINC(70,2)
COMMON /BLOCK3/ X(31),Y(31),Z(31),C1(31),C2(31),JCODE(31,3)
COMMON /BLOCK4/ D(6),DLAMD,DLAMD1,DS,ELONG,EPSLNI,EPSLN2,EPSLN3,
1 DETO,DET,DETOLD,DETNEW,LAMDA,LAMDA1,DSBR,
2 IHBW,IFLAG,JHBW,L,IFLAG,LHAT,LMAX,TRACE(2000),
3 NCOUNT,NEQS,NJ,NM,NNEG,NPOINT,SCALE,SYSTEM,K15,
4 UPDATE,CONT,ENDLIN,ENDSTB,INTEST,METHOD,BRANCH,
5 EXP,EXPO,EXPOLD,EXPNW,IMAX,JMAX,MPOINT,NDIGIT,
6 IHELP,BPOINT,LFLAG,COORD,NUMBER,LDA,ISOLVE

C INITIALIZE THE RESIDUAL FORCES TO THE CURRENT LOAD LEVEL.
DO 10 I=1,NEQS
  R(I)=LAMDA*DISTR(I)
10 CONTINUE
C SUBTRACT THE FORCES THAT COME FROM THE JOINT DISPLACEMENTS FROM
C THE RESIDUAL FORCE VECTOR.
DO 20 I=1,NM
  ELONG=XLD(I)-XL(I)
  PP1=A(I)*E(I)*ELONG*CG1(I)/XL(I)
  PP2=A(I)*E(I)*ELONG*CG2(I)/XL(I)
  PP3=A(I)*E(I)*ELONG*CG3(I)/XL(I)
20 CONTINUE
D(3)=-1.000*PP3
D(6)=PP3
GOTO(11,12,13)

11
D(1)=-1.000*PP1
D(2)=-1.000*PP2
D(4)=PP1
D(5)=PP2
GOTO 13

12
I1=MING(I,1)
I2=MING(I,2)
C1I=C1(I1)
C2I=C2(I1)
C1HI=C1(I2)
C2HI=C2(I2)
D(1)=-1.000*PP1*C1I-PP2*C2I
D(2)=PP1*C2I-PP2*C1I
D(4)=PP1*C1HI+PP2*C2HI
D(5)=PP2*C1HI-PP1*C2HI

13
DO 30 J=1,6
    IF(MCODE(I,J).EQ.0) GO TO 30
    ND=MCODE(I,J)
    R(ND)=R(ND)-D(J)
30 CONTINUE
20 CONTINUE
RETURN
SUBROUTINE RWC
IMPLICIT REAL*8 (A-H,O-Z)
INTEGER SCALE,SYSTEM,UPDATE,CONT,ENDLIM,ENDSTB,BRANCH,
1 EXP,EXPO,EXPOLD,EXPNEW,BPOINT
REAL*8 LOAD,LAMDA,LAMDA1
CHARACTER*2 ITRACE
COMMON /BLOCK1/ SS(70,70),DQI(70),DQ1(70),DQII(70),DQIII(70),DISTR(70),
1 LOAD(70),Q(70),R(70),QI(70),RI(70),KPVT(70)
COMMON /BLOCK2/ A(70),CG1(70),CG2(70),CG3(70),E(70),XL(70),XLD(70)
1 ,M CODE(70,6),M INC(70,2)
COMMON /BLOCK3/ X(31),Y(31),Z(31),C1(31),C2(31),JCODE(31,3)
1 DETO,DET,DETOLD,DETNEW,LAMDA,LAMDA1,DSBR,
2 IHBW,JFLAG,JHBW,L,IFLAG,LHAT,LMAX,ITRACE(2000),
3 NCOUNT,NEQS,NJ,NM,NNEG,NPOINT,SCALE,SYSTEM, K15,
4 UPDATE,CONT,ENDLIM,ENDSTB,INTEST,METHOD,BRANCH,
5 EXP,EXPO,EXPOLD,EXPNEW,IMAX,JMAX,MPOINT,NDIGIT,
6 IHELP,BPOINT,LFLAG,ICOORD,NUMBER,LDA,ISOLVE

C
$$
K15=K15+1
ITRACE(K15)='F'
C
SAVE THE FIRST LAMDA AND Q VECTOR FOR THE NORMAL PLANE ITERATION
CALL STORE(DQ1,DQ,DQ,0.00)
DLAMDI=DLAMD
L=1
CALL DEFORM
CALL RESID
GO TO 50
C
COMPUTE THE STIFFNESS MATRIX AND SOLVE K DQKI = DISTR IF IT IS
THE FIRST ITERATION OR IF UPDATING IS WANTED.

20 IF(L.NE.1.AND.UPDATE.NE.1) GO TO 30

C $$S G $$$
K15=K15+1
ITRACE(K15)=*G*
C UPDATE THE STIFFNESS MATRIX.
CALL STIFF
IF(ISOLVE.EQ.1)CALL SOLVE1
IF(ISOLVE.EQ.2)CALL SOLVE2
CALL STORE(LOAD,DISTR,LOAD,0.D0)
C SOLVE K DQKI = DISTR.
IF(ISOLVE.EQ.1)CALL SOLVE1
IF(ISOLVE.EQ.2)CALL SOLVE2
C TEMPORARILY STORE DQKI IN DQI.
CALL STORE(DQI,LOAD,DQI,0.D0)
30 CALL STORE(LOAD,R,R,0.D0)
C SOLVE K DQKII = R.
IF(ISOLVE.EQ.1)CALL SOLVE1
IF(ISOLVE.EQ.2)CALL SOLVE2
C TEMPORARILY STORE DQKII IN DQ.
CALL STORE(DQ,LOAD,DQ,0.D0)
C CALL THE APPROPRIATE SUBROUTINE TO GET DLAMD.
IF(METHOD.EQ.1) CALL NORMAL
IF(METHOD.EQ.2) CALL SPHERE
L=L+1
CALL DEFORM
CALL RESID
C CHECK FOR CONVERGENCE.
50 CALL CONVG
IF(IFLAG.EQ.1) RETURN
C CHECK TO SEE IF THE MAXIMUM NUMBER OF ITERATIONS HAS BEEN
C REACHED.
IF(L.NE.LMAX) GO TO 20
IF(ISOLVE.EQ.1) CALL DTM1G1
IF(ISOLVE.EQ.2) CALL DTM2G2(SS, LDA, NEQ, KPVT, DET, EXP, NNEG)
WRITE(3,1) NNEG, EXP, DET
11 FORMAT(' NUMBER OF NEGATIVE EIGENVALUES = ', I3, ' ', DETERM', ')
WRITE(2,1001)
1001 FORMAT(' TRACE OF PROGRAM PATH ', /)
WRITE(2,1002) (ITRACE(I), I=1,K15)
1002 FORMAT(1X, 1OA, 5X, 1OA, 5X, 1OA, 5X, 1OA, 5X, 1OA, 5X)
WRITE(2,1) NCOUNT
1 FORMAT(' NUMBER OF POINTS = ', I4, ' ', LIMIT EXCEEDED *****)
STOP
SUBROUTINE SOLVE1
IMPLICIT REAL*8 (A-H, O-Z)
INTEGER SCALE, SYSTEM, UPDATE, CONT, ENDLIM, ENDBT, BRANCH,
EXP, EXP0, EXPD, EXPNEW, BPINT
REAL*8 LOAD, LAMDA, LAMDA1
CHARACTER*2 ITRACE
COMMON / BLOCK1/ SS(70, 70), DQ(70), DQ1(70), DQ2(70), DISTR(70),
LOAD(70), Q(70), R(70), Q1(70), R1(70), KPVT(70)
COMMON / BLOCK2/ A(70), CG1(70), CG2(70), CG3(70), E(70), XL(70), XLD(70)
COMMON / BLOCK3/ X(31), Y(31), Z(31), C1(31), C2(31), JCODE(31, 3)
COMMON / BLOCK4/ D(6), DLA0, DLM), DS, ELONG, EPSLN1, EPSLN2, EPSLN3,
DET0, DET, DETOLD, DETNEW, LAMDA, LAMDA1, DSBR,
IHBF, JFLAG, JHBW, IFLAG, LHAT, LMAX, ITRACE(2000),
NCOUNT, NEQS, NJ, NM, NNEG, NPOINT, SCALE, SYSTEM, K15,
UPDATE, CONT, ENDLIM, ENDBT, INTEST, METHOD, BRANCH,
EXP, EXP0, EXPD, EXPNEW, IMAX, JMAX, MPOINT, NDIGIT,
IHELP, BPOINT, LFLAG, ICOORD, NUMBER, LDA, ISOLVE
C
FORWARD REDUCTION OF MATRIX
IF (JFLAG.EQ.1) GO TO 40
DO 10 N=1, NEQS
DO 20 LL=2, IHBW
IF (SS(N, LL).EQ.0.D0) GO TO 20
I=N+LL-1
C=SS(N, LL)/SS(N, 1)
J=0
DO 30 K=LL, IHBW
J=J+1
SS(I, J)=SS(I, J)-C*SS(N, K)
10 CONTINUE
20 CONTINUE
30 CONTINUE
40 CONTINUE
...
C  FORWARD REDUCTION OF CONSTANTS
40  DO 50  N=1,NEQS
   DO 60  LL=2,NBW
      IF(SS(N,LL).EQ.0.D0) GO TO 60
      I=N+LL-1
      LOAD(I)=LOAD(I)-SS(N,LL)*LOAD(N)
   CONTINUE
50  CONTINUE
   LOAD(N)=LOAD(N)/SS(N,1)
50  CONTINUE
C  SOLVE FOR UNKNOWNS BY BACK-SUBSTITUTION
DO 80  M=2,NEQS
   N=NEQS+1-M
   DO 70  LL=2,NBW
      IF(SS(N,LL).EQ.0.D0) GO TO 70
      K=N+LL-1
      LOAD(N)=LOAD(N)-SS(N,LL)*LOAD(K)
   CONTINUE
70  CONTINUE
80  CONTINUE
RETURN
SUBROUTINE SOLVE2
IMPLICIT REAL*8 (A-H,O-Z)
INTEGER SCALE,SYSTEM,UPDATE,CONT,ENDLIM,ENOSTB,BRANCH,
EXP,EXPO,EXPNEW,BPOINT
REAL*8 LOAD,LAMDA,LAMDA1
CHARACTER*2 ITRACE
COMMON /BLOCK1/ SS(70,70),DQ(70),DQ1(70),DQ1(70),DISTR(70),
LOAD(70),Q1(70),R(70),Q1(70),R(70),KPVT(101)
COMMON /BLOCK2/ A(70),CG1(70),CG2(70),CG3(70),E(70),XL(70),XLD(70)
COMMON /BLOCK3/ X(31),Y(31),Z(31),C1(31),C2(31),JCODE(31,3)
COMMON /BLOCK4/ D(6),DLAMD,DLAMD1,DS,ELONG,EPSLN1,EPSLN2,EPSLN3,
DET,DET,DETOLD,DETNEW,LAMDA,LAMDA1,DSBR,
IHBN,JFLAG,JHBW,L,IFLAG,LHAT,LMAX,ITRACE
NCOUNT,NEQS,NJ,NM,NNEG,NPOINT,SCALE,SYSTEM,K15,
UPDATE,CONT,ENDLIM,ENDSTB,INTEST,METHOD,BRANCH,
EXP,EXPO,EXPNEW,IMAX,JMAX,MPOINT,MPOINT,NDIGIT,
IHHELP,BPOINT,LFLAG,ICOORD,NUMBER,LOA,ISOLVE
IF(JFLAG.EQ.1)GOTO 40
FORWARD REDUCTION OF COEFFICIENT MATRIX
CALL SSIFA(SS,LOA,NEQS,KPVT,INFO)
IF(INFO.EQ.0)GOTO 30
WRITE(2,50)
50 FORMAT(///,'* * * MATRIX IS SINGULAR--EXECUTION STOPPED TO AVOID E
ERROR * * *')
STOP
30 JFLAG=1
RETURN

DETERMINATION OF SOLUTION VECTOR

40 CALL SSISL(SS, LDA, NEQS, KPVT, LOAD)
RETURN
SUBROUTINE SCLVE2
IMPLICIT REAL*8 (A-H,O-Z)
INTEGER SCALE,SYSTEM,UPDATE,CONT,ENDLIM,ENDSTB,BRANCH,
EXP,EXPO,EXPOLD,EXPNEW,BPOINT
REAL*8 LOAO,LAMDA,LAMDAL
CHARACTER*2 ITRACE
COMMON /BLOCK1/ SS(70,70),DQ(70),DQI(70),DQII(70),DISTR(70),
LOAD(70),Q(70),QI(70),QII(70),R(70),R1(70),R2(70),R6(70),R6(70)
COMMON /BLCK2/ A(70),CG1(70),CG2(70),CG3(70),E(70),XL(70),XLD(70)
COMMON /BLCK3/ M(31),Y(31),Z(31),C1(31),C2(31),JCODE(31,3)
COMMON /BLCK4/ D(6),DLAMDO,DLAMDO,DSS,ELONG,EPOLL,EPOLL2,EPOLL3,
DET0,DET,DETOLD,DETNEW,LAMDA,LAMDAll,DSBR,
IHBw,JFLAG,JSPW,L,IFLAG,LHAT,LMAX,ITRACE(2000),
NCOUNT,NEQS,NJ,NM,NNEG,NPOINT,SCALE,SYSTEM,K15,
UPDATE,CONT,ENDLIM,ENDSTB,INTEST,METHOD,BRANCH,
EXP,EXPO,EXPOLD,EXPNEW,IMAX,JMAX,NPOINT,NDIGIT,
HELP,BPOINT,LFLAG,ICOORO,NUMBER,LOA,ISOLVE
IF(JFLAG.EQ.1)GOTO 80
FORWARD REDUCTION OF COEFFICIENT MATRIX
CALL SSIFA(SS,LDA,NEQS,KPVT,INFO)
IF(INFO.EQ.0)GOTO 80
WRITE(2,50)GO TO 80
50 FORMAT(/**,*** ** MATRIX IS SINGULAR—EXECUTION STOPPED TO AVOID E
ERROR ** **)
STOP
30 JFLAG=1
RETURN

DETERMINATION OF SOLUTION VECTOR

CALL SSISL(SS, LOA, NEQS, KPVT, LOAD)
RETURN
SUBROUTINE SSIFA(SS,LOA,NEQS,KPVT,INFO)
IMPLICIT REAL*8 (A-H,O-Z)
INTEGER KPVT(NEQS)
REAL*8 SS(LDA,NEQS),MULK,MULKM1
LOGICAL SWAP

ALPHA=(1.0+DSQRT(17.0))/8.0
INFO=0
K=NEQS
10 CONTINUE
IF(K.EQ.0)GOTO 200
IF(K.GT.1)GOTO 20
KPVT(1)=1
IF(SS(1,1).EQ.0.DO)INFO=1
GOTO 200
20 CONTINUE
KM1=K-1
ABSACK=DABS(SS(K,K))
IMAX=ISAMAX(K-1,SS(1,K),1,NEQS)
COLMAX=DABS(SS(IMAX,K))
IF(ABSACK.LT.ALPHA*COLMAX)GOTO 30
KSTEP=1
SWAP=.FALSE.
GOTO 40
30 CONTINUE
ROWMAX=0.DO
IMAXP1=IMAX+1
DO 40 J=IMAXP1,K
   ROWMAX=DMAX1(ROWMAX,DABS(SS(IMAX,J)))
40 CONTINUE
IF(IMAX.EQ.1)GOTO 50
JMAX=ISAMAX(IMAX-1,SS(1,IMAX),1,NEQS)
ROWMAX=DMAX1(ROWMAX,DABS(SS(JMAX,IMAX)))

50 CONTINUE
IF(DABS(SS(IMAX,IMAX)).LT.ALPHA*ROWMAX)GOTO 60
KSTEP=1
SWAP=.TRUE.
GOTO 80

60 CONTINUE
IF(ABS(AK).LT.ALPHA*COLUMNMAX*(COLUMNMAX/ROWMAX))GOTO 70
KSTEP=1
SWAP=.FALSE.
GOTO 80

70 CONTINUE
KSTEP=2
SWAP=IMAX.NE.KM1

80 CONTINUE
90 CONTINUE
IF(DMAX1(ABS(AK),COLUMNMAX).NE.0.0)GOTO 100
KPVT(K)=K
INFO=K
GOTO 190

100 CONTINUE
IF(KSTEP.EQ.2)GOTO 140
IF(.NOT.SWAP)GOTO 120
CALL SS(WAP(IMAX,SS(1,IMAX),1,SS(1,K),1,NEQS)
DO 110 JJ=IMAX,K
J=K+IMAX-JJ
T=SS(J,K)
SS(J,K)=SS(IMAX,J)
SS(IMAX,J)=T

110 CONTINUE
120 CONTINUE
   DO 130 JJ=1,KM1
      J=K-JJ
      MULK=-SS(J,K)/SS(K,K)
      T=MULK
      CALL SAXPY(J,T,SS(1,K),1,SS(1,J),1,NEQS)
      SS(J,K)=MULK
   130 CONTINUE
   KPVT(K)=K
   IF(SWAP)KPVT(K)=IMAX
   GOTO 190
140 CONTINUE
   IF(.NOT.SWAP)GOTO 160
   CALL SWAP(IMAX,SS(1,IMAX),1,SS(1,K-1),1,NEQS)
   DO 150 JJ=IMAX,KM1
      J=KM1+IMAX-JJ
      T=SS(J,K-1)
      SS(J,K-1)=SS(IMAX,J)
      SS(IMAX,J)=T
   150 CONTINUE
   T=SS(K-1,K)
   SS(K-1,K)=SS(IMAX,K)
   SS(IMAX,K)=T
160 CONTINUE
   KM2=K-2
   IF(KM2.EQ.0)GOTO 180
   AK=SS(K,K)/SS(K-1,K)
   AKM1=SS(K-1,K-1)/SS(K-1,K)
   DENOM=1.DO-AK*AKM1
   DO 170 JJ=1,KM2
      J=KM1-JJ
      BK=SS(J,K)/SS(K-1,K)
   170 CONTINUE
BKML = SS(J, K-1) / SS(K-1, K)

MULK = (AKM1 * BK - BKM1) / DENOM

MULKML = (AKM1 * BKM1 - BK) / DENOM

T = MULK

CALL SAXPY(J, T, SS(1, K), 1, SS(1, J), 1, NEQS)

T = MULKML

CALL SAXPY(J, T, SS(1, K-1), 1, SS(1, J), 1, NEQS)

SS(J, K) = MULK

SS(J, K-1) = MULKML

170 CONTINUE

180 CONTINUE

KPVT(K) = 1 - K

IF(SWAP) KPVT(K) = -IMAX

KPVT(K-1) = KPVT(K)

190 CONTINUE

K = K - KSTEP

GOTO 10

200 CONTINUE

RETURN
SUBROUTINE SSISL

SUBROUTINE SSISL(SS,LDA,NEQS,KPVT,LOAD)

IMPLICIT REAL*8 (A-H,O-Z)
INTEGER KPVT(NEQS)
REAL*8 SS(LDA,NEQS),LOAD(NEQS)

K=NEQS

10 IF(K.EQ.0)GOTO 80
   IF(KPVT(K).LT.0)GOTO 40
   IF(K.EQ.1)GOTO 30
   KP=KPVT(K)
   IF(KP.EQ.K)GOTO 20
   TEMP=LOAD(K)
   LOAD(K)=LOAD(KP)
   LOAD(KP)=TEMP

20 CONTINUE
   CALL SAXPY(K-1,LOAD(K),SS(1,K),1,LOAD(1),1,NEQS)

30 CONTINUE
   LOAD(K)=LOAD(K)/SS(K,K)
   K=K-1
   GOTO 70

40 CONTINUE
   IF(K.EQ.2)GOTO 60
   KP=IABS(KPVT(K))
   IF(KP.EQ.K-1)GOTO 50
   TEMP=LOAD(K-1)
   LOAD(K-1)=LOAD(KP)
   LOAD(KP)=TEMP

50 CONTINUE
   CALL SAXPY(K-2,LOAD(K),SS(1,K),1,LOAD(1),1,NEQS)
   CALL SAXPY(K-2,LOAD(K-1),SS(1,K-1),1,LOAD(1),1,NEQS)
60 CONTINUE
    AK = SS(K, K) / SS(K-1, K)
    AKM1 = SS(K-1, K-1) / SS(K-1, K)
    BK = LOAD(K) / SS(K-1, K)
    BKMI = LOAD(K-1) / SS(K-1, K)
    DENOM = AK * AKM1 - 1.0
    LOAD(K) = (AKM1 * BK - BKMI) / DENOM
    LOAD(K-1) = (AK * BKMI - BK) / DENOM
    K = K-2
70 CONTINUE
    GOTO 10
80 CONTINUE
    K = 1
90 IF(K.GT.NEQS) GOTO 160
    IF(KPVT(K).LT.0) GOTO 120
    IF(K.EQ.1) GOTO 110
    LOAD(K) = LOAD(K) + SDOT(K-1, SS(1, K), 1, LOAD(1), 1, NEQS)
    KP = KPVT(K)
    IF(KP.EQ.K) GOTO 100
    TEMP = LOAD(K)
    LOAD(K) = LOAD(KP)
    LOAD(KP) = TEMP
100 CONTINUE
110 CONTINUE
    K = K + 1
    GOTO 150
120 CONTINUE
    IF(K.EQ.1) GOTO 140
    LOAD(K) = LOAD(K) + SDOT(K-1, SS(1, K), 1, LOAD(1), 1, NEQS)
    LOAD(K+1) = LOAD(K+1) + SDOT(K-1, SS(1, K+1), 1, LOAD(1), 1, NEQS)
    KP = IABS(KPVT(K))
    IF(KP.EQ.K) GOTO 130
TEMP = LOAD(K)
LOAD(K) = LOAD(KP)
LOAD(KP) = TEMP
130 CONTINUE
140 CONTINUE
K = K + 2
150 CONTINUE
GOTO 90
160 CONTINUE
RETURN
FUNCTION SDOT

FUNCTION SDOT(N, SX, INCX, SY, INCY, NEQS)
IMPLICIT REAL*8 (A-H, O-Z)
REAL*8 SX(NEQS), SY(NEQS)
STEP=0.000
SDOT=0.000
IF (N.LE.0) GO TO 70
IF (INCX.EQ.1 .AND. INCY.EQ.1) GO TO 20
IX=1
IY=1
IF (INCX.LT.0) IX=(-N+1)*INCX+1
IF (INCY.LT.0) IY=(-N+1)*INCY+1
DO 10 I=1,N
STEP=STEP+SX(IX)*SY(IY)
IX=IX+INCX
IY=IY+INCY
10 CONTINUE
SDOT=STEP
GO TO 70
20 M=MOD(N,5)
IF (M.EQ.0) GO TO 40
DO 30 I=1,M
STEP=STEP+SX(I)*SY(I)
30 CONTINUE
IF (N.LT.5) GO TO 60
40 M1=M+1
DO 50 I=M1,N,5
STEP=STEP+SX(I)*SY(I)+SXI(I+1)*SY(I+1)+SXI(I+2)*SY(I+2)+SXI(I+3)*SY(I+3)+SXI(I+4)*SY(I+4)
50 CONTINUE
60   SOOT=STEMP
70   CONTINUE
     RETURN
SUBROUTINE SAXPY(N, SA, SX, INCX, SY, INCY, NEQS)
IMPLICIT REAL*8 (A-H,O-Z)
REAL*8 SX(NEQS), SY(NEQS), SA
IF (N.LE.0) RETURN
IF (SA.EQ.0) RETURN
IF (INCX.EQ.1 .AND. INCY.EQ.1) GO TO 20
IX = 1
IY = 1
IF (INCX.LT.0) IX = (-N+1)*INCX + 1
IF (INCY.LT.0) IY = (-N+1)*INCY + 1
DO 10 I = 1, N
SY(IY) = SY(IY) + SA*SX(IX)
IX = IX + INCX
IY = IY + INCY
10 CONTINUE
M = MOD(N, 4)
IF (M.EQ.0) GO TO 40
DO 30 I = 1, M
SY(I) = SY(I) + SA*SX(I)
30 CONTINUE
IF (N.LT.4) RETURN
MP = M + 1
DO 50 I = MP, N, 4
SY(I) = SY(I) + SA*SX(I)
SY(I+1) = SY(I+1) + SA*SX(I+1)
SY(I+2) = SY(I+2) + SA*SX(I+2)
SY(I+3) = SY(I+3) + SA*SX(I+3)
50 CONTINUE
RETURN
SUBROUTINE SSWAP(N,SX,INCX,SY,INCY,NEQS)
IMPLICIT REAL*8 (A-H,O-Z)
REAL*8 SX(NEQS),SY(NEQS)
IF(N.LE.0)RETURN
IF(INCX.EQ.1.AND.INCY.EQ.1)GOTO 20
IX=1
IY=1
IF(INCX.LT.0)IX=(-N+1)*INCX+1
IF(INCY.LT.0)IY=(-N+1)*INCY+1
DO 10 I=1,N
STEMP=SX(IX)
SX(IX)=SY(IY)
SY(IY)=STEMP
IX=IX+INCX
IY=IY+INCY
10 CONTINUE
RETURN
20 M=MOD(N,3)
IF(M.EQ.0)GOTO 40
DO 30 I=1,M
STEMP=SX(I)
SX(I)=SY(I)
SY(I)=STEMP
30 CONTINUE
IF(N.LT.3)RETURN
40 MPL=M+1
DO 50 I=MPL,N,3
STEMP=SX(I)
SX(I)=SY(I)
50 CONTINUE
SY(I) = TEMP
TEMP = SX(I+1)
SX(I+1) = SY(I+1)
SY(I+1) = TEMP
TEMP = SX(I+2)
SX(I+2) = SY(I+2)
SY(I+2) = TEMP
50 CONTINUE
RETURN
C***********************************************************
C**FUNCTION ISAMAX**
C***********************************************************

INTEGER FUNCTION ISAMAX(N,SX,INCX,NEQS)
IMPLICIT REAL*8 (A-H,O-Z)
REAL*8 SX(NEQS)
ISAMAX=0
IF(N.LT.1)RETURN ISAMAX=1
IF(N.EQ.1)RETURN IF(INCX.EQ.1)GOTO 20
IX=1
SMAX=DAABS(SX(1))
IX=IX+INCX
   DO 10 I=2,N
      IF(DAABS(SX(I)).LE.SMAX)GOTO 5
      ISAMAX=I
      SMAX=DAABS(SX(I))
  5  IX=IX+INCX
10  CONTINUE
RETURN
20 SMAX=DAABS(SX(1))
   DO 30 I=2,N
      IF(DAABS(SX(I)).LE.SMAX)GOTO 30
      ISAMAX=I
      SMAX=DAABS(SX(I))
30  CONTINUE
RETURN
C*********************************************************************
C*
C SPHERE
C*********************************************************************
SUBROUTINE SPHERE
Implicit REAL*8 (A-H,O-Z)
INTEGER SCALE, SYSTEM, UPDTE, CONT, ENDLIM, ENSTB, BRANCH,
1 EXP, EXPNEW, EXPOLD, BPOINT
REAL*8 LOAD, LAMDA, LAMDA1
CHARACTER*2 !TRACE
COMM
/BLOCK1/ SS(70,70), DQ(70), DQ1(70), DQ1(70), Q(70), R(70), Q(70), R(70), KPVT(70)
1 COMMON /BLOCK2/ A(70), CG1(70), CG2(70), CG3(70), E(70), X(70), XL(70), X(LD(70))
1 COMMON /BLOCK3/ MCODE(70,6), MNC(70,2)
1 COMMON /BLOCK4/ X(31), Y(31), Z(31), CL(31), C2(31), JCODE(31,3)
1 COMMON /BLOCK4/ D(6), DLAMD, DLAMD1, DS, ELONG, EPSLN1, EPSLN2, EPSLN3,
1 DETO, DET, DETOLD, DETNEW, LAMDA, LAMDA1, DSBR,
2 IHBW, JFLAG, JHBW, L, IFLAG, LMAX, LMAX, ITRACE(2000),
3 NCOUNT, NEQS, NJ, NM, NNEG, NPOINT, SCALE, SYSTEM, K15,
4 UPDATE, CONT, ENSTB, INTEST, METHOD, BRANCH,
5 EXP, EXPNEW, EXPOLD, EXP, IMAX, JMAX, NPOINT, NDIGIT,
6 HELP, BPOINT, IFLAG, ICOORD, NUMBER, LOA, ISOLVE
C FIND THE CONSTANTS FOR THE QUADRATIC EQUATION.
CALL STORE(LUAD, Q, Q1, -1.00)
A1=1.00+F(DQ1, DQ1, DQ1, 0)
B1=(LAMDA-LAMDA1+F(DQ1, DQ, LOAD, 1)) * 2.0
D1=F(DQ, DQ, LOAD, 2)
C FIND THE ROOTS OF THE QUADRATIC EQUATION.
G2=B1*B1-4.00*A1*D1
IF(G2.GT.0.00) GO TO 20
WRITE(2,1) G2
1 FORMAT(/**, ' NEGATIVE VALUE IN SPHERE IS ', G24.16,
1 ' PROGRAM IS STOPPED TO AVOID ERROR. */)
WRITE(2,1001)
1001 FORMAT(/,' Trace of program path/',)
WRITE(2,1002) (ITRACE(I),I=1,K15)
1002 FORMAT(1X,10A2,5X,10A2,5X,10A2,5X,10A2,5X,10A2,5X)
STOP
20 G2=DSQRT(G2)
   G1=(-B1+G2)/(2.0D0*A1)
   G2=(-B1-G2)/(2.0D0*A1)
C CHOOSE THE RCOT THAT LEADS TO THE SMALLEST INCREMENTAL VECTOR.
   DLAMD=G1
   CALL STORE(LOAD,DQ,DQI,G2)
   CALL STORE(DQ,DQ,DQI,G1)
   IF(DSQRT(G1*G1+F(DQ,DQ,DQ,0)).LT.
      DSQRT(G2*G2+F(LOAD,LOAD,LOAD,0))) GO TO 10
   CALL STORE(DQ,LOAD,LOAD,0.0D0)
   DLAMD=G2
C UPDATE THE VARIABLES.
10 LAMDA=LAMDA+DLAMD
   CALL STORE(Q,Q,DQ,1.0D0)
RETURN
SUBROUTINE STIFF
IMPLICIT REAL*8 (A-H,O-Z)
INTEGER SCALE,SYSTEM,UPDATE,CONT,ENDLIM,ENDSTB,BRANCH,
1 EXP,EXPO,EXPOLD,EXPNEW,BPOINT
REAL*8 LOAD,LAMDA,LAMDA1
CHARACTER*2 ITRACE
COMMON /BLOCK1/ SS(70,70),DQ(70),DQ1(70),DQI(70),DISTR(70),
1 LOAD(70),Q(70),R(70),Q1(70),R1(70),KPVT(70)
COMMON /BLOCK2/ A(70),CG1(70),CG2(70),CG3(70),E(70),XL(70),XLD(70)
1 ,MCODE(70,6),MINC(70,2)
COMMON /BLOCK3/
1 ,MCODE(70,6),MINC(70,2)
COMMON /BLOCK4/
1 ,MCODE(70,6),MINC(70,2)
COMMON /BLOCK4/ X(31),Y(31),Z(31),C1(31),C2(31),JCODE(31,3)
COMMON /BLOCK4/ X(31),Y(31),Z(31),C1(31),C2(31),JCODE(31,3)
COMMON /BLOCK4/ X(31),Y(31),Z(31),C1(31),C2(31),JCODE(31,3)
COMMON /BLOCK4/ X(31),Y(31),Z(31),C1(31),C2(31),JCODE(31,3)
COMMON /BLOCK4/ X(31),Y(31),Z(31),C1(31),C2(31),JCODE(31,3)
DIMENSION H(15),INDEX(6,6,2)
DATA INDEX /1,2,4,-1,-2,-4,2,3,5,-2,-3,-5,4,5,6,-4,-5,-6,
1 -1,-2,-4,1,2,4,-2,-3,-5,2,3,5,-4,-5,-6,4,5,6,
2 1,2,3,4,5,-3,2,6,7,8,9,-7,3,7,10,11,12,-10,4,8,
3 11,13,14,-11,5,9,12,14,15,-12,-3,-7,-10,-11,-12,4
4 10/
JFLAG=0
IF(ISOLVE.EQ.2)IHBW=NEQS
DO 10 II=1,NEQS
DG 20 JJ=1,IHBW
SS(II,JJ)=0.DO
20 CONTINUE
10 CONTINUE
DO 30 I=1,NM
  AI=A(I)
  EI=E(I)
  XLI=XL(I)
  XLDI=XLD(I)
  CG1I=CG1(I)
  CG2I=CG2(I)
  CG3I=CG3(I)
  ELONG=XLDI-XLI
  H(1)=AI*EI*(CG1I*CG1I*(ELONG/XLDI)*((1.00-CG1I*CG1I))/XLI
  H(2)=AI*EI*CG1I*CG2I*(1.00-ELONG/XLDI)/XLI
  H(3)=AI*EI*CG2I*CG2I*(1.00-ELONG/XLDI)*((1.00-CG2I*CG2I))/XLI
  H(4)=AI*EI*CG1I*CG3I*(1.00-ELONG/XLDI)/XLI
  H(5)=AI*EI*CG2I*CG3I*(1.00-ELONG/XLDI)/XLI
  H(6)=AI*EI*CG3I*CG3I*(1.00-ELONG/XLDI)*((1.00-CG3I*CG3I))/XLI
  IF(ICCQRD.EQ.1)GOTO 70
  H1=H(1)
  H2=H(2)
  H3=H(3)
  H4=H(4)
  H5=H(5)
  H6=H(6)
  I1=MINC(1,1)
  I2=MINC(1,2)
  C1I=C1(I1)
  C2I=C2(I1)
  C1HI=C1(I2)
  C2HI=C2(I2)
  H(1)=C1I**2*H1+2.00*C1I*C2I*H2+C2I**2*H3
  H(2)=-1.00*C1I*C2I*H1+(C1I**2-C2I**2)*H2+C1I*C2I*H3
\[ H(3) = C_{11}H_4 + C_{21}H_5 \]
\[ H(4) = -1.0D0 \times C_{11}H_1C_{11}H_1H_1 - (C_{11}C_{21}H_1 + C_{21}C_{11}H_1)H_2 - C_{21}C_{21}H_1H_3 \]
\[ H(5) = C_{11}C_{21}H_1H_1 - (C_{11}C_{11}H_1C_{21}C_{21}H_1)H_2 - C_{21}C_{11}H_1H_3 \]
\[ H(6) = C_{21}H_2H_1 - 2.0D0 \times C_{11}C_{21}H_1H_2 + C_{11}H_2H_3 \]
\[ H(7) = -1.0D0 \times C_{21}H_4 + C_{11}H_5 \]
\[ H(8) = C_{21}C_{11}H_1H_1 - (C_{11}C_{11}H_1C_{21}C_{21}H_1)H_2 - C_{11}C_{21}H_1H_3 \]
\[ H(9) = -1.0D0 \times C_{21}C_{21}H_1H_1 + (C_{11}C_{21}H_1C_{21}C_{11}H_1)H_2 - C_{21}C_{11}H_1H_3 \]
\[ H(10) = H_6 \]
\[ H(11) = -1.0D0 \times C_{11}H_1H_4 - C_{21}H_5 \]
\[ H(12) = C_{21}H_4 - C_{11}H_5 \]
\[ H(13) = C_{11}H_1H_2H_1 + 2.0D0 \times C_{11}H_1C_{21}H_1H_2 + C_{21}H_2H_3 \]
\[ H(14) = -1.0D0 \times C_{21}H_1C_{21}H_1H_1 + (C_{11}H_1H_2C_{21}H_2H_3)H_2 + C_{11}H_1C_{21}H_1H_3 \]
\[ H(15) = C_{21}H_2H_1 - 2.0D0 \times C_{11}H_1C_{21}H_1H_2 + C_{11}H_1H_3 \]

GOTO(70,170), ISOLVE

170  DO 40 JM=1,6
    J=MCODE(I,JM)
    IF(J.EQ.0) GO TO 40
    DO 50 KM=JM,6
        K=MCODE(I,KM)
        IF(K.EQ.0) GO TO 50
        KB=K-J+1
        IN=INDEX(JM,KM,INDEX)
        IF(IN.LE.0) GO TO 60
        SS(J,KB)=SS(J,KB)+H(IN)
        GO TO 50
    60  CONTINUE
40  CONTINUE
GOTO 30

70  DO 40 JM=1,6
    J=MCODE(I,JM)
    IF(J.EQ.0) GO TO 40
    DO 50 KM=JM,6
        K=MCODE(I,KM)
        IF(K.EQ.0) GO TO 50
        KB=K-J+1
        IN=INDEX(JM,KM,INDEX)
        IF(IN.LE.0) GO TO 60
        SS(J,KB)=SS(J,KB)-SS(J,KB)-H(IN)
        GO TO 50
    60  CONTINUE
40  CONTINUE
GOTO 30
DO 150 KM = JM, 6
    K = MCODE(I, KM)
    IF(K.EQ.0) GO TO 150
    IN = INDEX(JM, KM, ICOORD)
    IF(IN.LT.0) GO TO 160
    SS(J, K) = SS(J, K) + H(IN)
    GO TO 150
   160  SS(J, K) = SS(J, K) - H(IN)
   150  CONTINUE
   140 CONTINUE
   30 CONTINUE
RETURN
C**************************************************************
C*           STORE                                           *
C**************************************************************

SUBROUTINE STORE(X1,Y1,Z1,V)
IMPLICIT REAL*8 (A-H,O-Z)
INTEGER SCALE,SYSTEM,UPDATE,CONT,ENDLIM,ENDSTB,BRANCH,
1 EXP,EXPO,EXPOLD,EXPNEW,BPOINT
REAL*8 LOAD,LAMDA,LAMDA1
CHARACTER*2 ITRACE
COMMON /BLOCK1/
1 LOAO,LAMDA,LAMDA1
COMMON /BLOCK2/
1 LOAD(70),Q(70),R(70),Q1(70),R1(70),KPVT(70)
COMMON /BLOCK3/
1 MCODE(70,6),HINC(70,2)
COMMON /BLOCK4/
1 MCODE(70,6),HINC(70,2)

CSTORE Y1 + V * Z1 IN X1.
IF(V.EQ.0.00) GO TO 20
DO 10 I=1,NEQS
   X1(I)=Y1(I)+V*Z1(I)
10 CONTINUE
RETURN
DO 30 I=1,NEQS
   X1(I)=Y1(I)
30 CONTINUE
RETURN
SUBROUTINE VECTOR

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

INTEGER SCALE, SYSTEM, UPDATE, CONT, ENDLIM, ENDSB, BRANCH,
           EXP, EXPO, EXPNEW, BPOINT

REAL*8 LOAD, LAMDA, LAMDA1

CHARACTER*2 ITRACE

COMMON /BLOCK1/
             SS(70,70), DQ(70), DQ1(70), DQI(70), DISTR(70),
             LOAD(70), Q(70), R(70), QI(70), R1(70), KPVT(70)

COMMON /BLOCK2/
             A(70), CG1(70), CG2(70), CG3(70), E(70), XL(70), XLD(70)

               MCODE(70,6), MNC(70,2)

COMMON /BLOCK3/
             X(31), Y(31), Z(31), C1(31), C2(31), JCODE(31,3)

COMMON /BLOCK4/
             D(6), DLAMD, DLAMD1, DS, ELONG, EPSLN1, EPSLN2, EPSLN3,

               DETO, DET, DETOLD, DETNEW, LAMDA, LAMDA1, DBSBR,
               IHB, JFLAG, JHBW, L, IFLAG, LHAT, LMAX, ITRACE(2000),
               NCOUNT, NEQS, NJ, NM, NNEG, NPOINT, SCALE, SYSTEM, K15,
               UPDATE, CONT, ENDLIM, ENDSB, INTEST, METHOD, BRANCH,
               EXP, EXPNEW, EXPOLD, EXPNEW, IMAX, JMAX, NPOINT, NDIGIT,
               IHELP, BPOINT, LFLAG, ICOORD, NUMBER, LOA, ISOLVE

C *** R ***
K15=K15+1
ITRACE(K15)=RBRANCH=0
CALL POWNV
C

FIND THE TANGENT VECTOR TO GET THE NORMAL VECTOR.
CALL :STORE(LOAD, DISTR, LOAD, 0.DO)
IF(ISOLVE.EQ.1) CALL SOLVE1
IF(ISOLVE.EQ.2) CALL SOLVE2
IF(NCOUNT.EQ.0) GO TO 70
DLAMD=DS/DSQRT(F(LOAD, LOAD, LOAD, 0)*1.DO)
GO TO 80
70 DS=DLAMD*DSQRT(F(LOAD,LOAD,LOAD,0)+1.00)
80 DO 90 I=1,NEQS
   DQ(I)=LOAD(I)*DLAMD
   CONTINUE
90 CONTINUE
   LAMDA1=LAMDA
   CALL STORE(Q1,Q,Q,0.00)
   WRITE(2,32)
32 FORMAT(/,' TANGENT VECTOR' ,/)
   WRITE(2,2) (DQ(I), I=1,NEQS), DLAMD
   DL TEMP=DLAMD
   CALL STORE(LOAD,DQ,DQ,0.00)
   C CALCULATE THE NORMAL VECTOR.
   ALPH=-F(DQ,DQ,DQ,0)+DLAMD*DLAMD)/F(DQ,R1,R1,0)
   CALL STORE(DQ,DQ,R1,ALPH)
   DS=DSQRT(F(DQ,DQ,DQ,0)+DLAMD*DLAMD)
   G10=DSBR/DS
   DS=DSBR
   DO 65 I=1,NEQS
      DQ(I)=DQ(I)*G10
   CONTINUE
65 CONTINUE
   DLAMD=DLAMD*G10
   C UPDATE FOR THE NORMAL VECTOR.
   CALL STORE(Q,Q,DQ,1.00)
   LAMDA=LAMDA+DLAMD
   NPOINT=MPOINT+NCOUNT-1
   WRITE(2,31)
31 FORMAT(/,' NORMAL VECTOR',/)
   WRITE(2,2) (DQ(I), I=1,NEQS), DLAMD
   WRITE(2,49) DS
49 FORMAT(/,' DS =',G24.16)
C   CALCULATE THE DOT PRODUCT OF THE NORMAL AND TANGENT VECTORS.
C
G10 = F(LOAD, DQ, DQ, O) + DLAMD * DLTEMP
WRITE(2, 37) G10
37 FORMAT(///, ' DOT PRODUCT =', G24.16, ///)
RETURN
END
The vita has been removed from the scanned document.
LINEAR AND NONLINEAR ANALYSIS OF SPACE TRUSSES
RELATIVE TO CYLINDRICAL COORDINATES

by

Jeffrey A. Perrier

(ABSTRACT)

The element and system models for linear and nonlinear space trusses were formulated and implemented relative to cylindrical coordinates. Nonlinear equilibrium paths were traced using the modified Riks/Wempner method. Two radially symmetric reticulated domes which exhibit snap-through instability and which were subjected to radially symmetric loads were analyzed both linearly and nonlinearly. Along with the entire structures, "pie slices" of the structures were also analyzed. Accuracy was observed by comparison of element forces or of critical load values to those previously presented for these trusses.

Reduction of computational effort from analysis of "pie slices" was studied, as were suppression and generation of bifurcation buckling modes.

Structural models in cylindrical coordinates were found to greatly reduce computational effort by analysis of "pie slices" of radially symmetric structures with radially symmetric loads, and analysis of "pie slices" provided qualitative information regarding buckling mode geometries when techniques for branching onto bifurcation paths failed.