A COMPARISON OF TWO MODELS FOR GEOMETRICALLY NONLINEAR FINITE ELEMENT ANALYSIS OF PLANE FRAMES

by

Michael Joseph Butler

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, 1983

Blacksburg, Virginia

A COMPARISON OF TWO MODELS FOR GEOMETRICALLY NONLINEAR FINITE ELEMENT ANALYSIS OF PLANE FRAMES

by

Michael Joseph Butler

(ABSTRACT)

A well structured computer program has been developed to implement and compare two models for geometrically nonlinear finite element analysis of plane frames. One model is developed using classical beam-column theory, and the other is formulated using standard finite element techniques. The beam-column model is shown to converge to a more accurate equilibrium path using less elements to model a structure than the second model for frames. The models are also compared on the basis of their limitations and on the basis of the degree of difficulty for their implementation. The modified Riks/Wempner method is used to perform postbuckling analysis and to study the effect of varying the composition of the tangent stiffness matrix on its ability to detect instability. The importance of including all contributions in the incrementation of a model derived using convected coordinates is also studied.

ACKNOWLEDGEMENTS

The author expresses his sincere appreciation to Dr. S. M. Holzer for his guidance, suggestions, and many helpful discussions. Gratitude is also extended to Dr. A. E. Somers and Dr. R. M. Barker for reviewing this thesis and for serving on the committee.

The author would like to thank his wife and his family for their love and support during his education.

TABLE OF CONTENTS

TABLE OF CONTENTS (cont.)

TABLE OF CONTENTS (cont.)

LIST OF FIGURES

LIST OF FIGURES (cont.)

LIST OF FIGURES (cont.)

LIST OF FIGURES (cont.)

LIST OF FIGURES (cont.)

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Purpose and Scope

The primary goal of this study is to implement and compare two models for geometrically nonlinear finite element analysis of plane frames. The first of these models is founded on conventional beam-column theory (36) and is developed by Oran (24). The beam-column model is compared to a model which is formulated by standard finite element procedures. This model uses Hermite interpolation (7) in the discretization process and formulates conditions of equilibrium based on the principle of virtual work. Although both models are actually implemented as finite element models, the former is referred to as the beam-column model and the latter as the finite element model. The models are compared on the basis of their efficiency, accuracy, and convergence characteristics, the degree of difficulty for their implementation, and their limitations.

Postbuckling analysis is performed because of the insight that is gained into the behavior of a model in a region of instability. Particularly, a satisfactory model should detect structural instability. Additionally, differences in accuracy often become more pronounced as the solution process moves farther away from the undeformed state and, for example, beyond a snap-through condition.

A final goal is to develop a well structured computer program. In a well structured program, the major emphasis is placed on making the program easily understandable. A program code that does not sacrifice readability for computational efficiency does not restrict its user to the program's present capabilities.

Some of the ideas and techniques that are necessary for nonlinear analysis are reviewed in chapter two because they are generally not available from one source.

1.2 Literature Review

The topic of nonlinear finite element analysis is treated by many authors. A very basic and comprehensive review is given by Cook (5), and additional information on certain topics is available in references 1 and 41. Performing nonlinear analysis using the modified Riks/Wempner method is covered by Holzer, Watson, and Vu (13) and in references 19 and 37.

The model developed using classical beam-column theory is based on that presented by Oran (24,25). Intermediate steps in the development are shown in reference 23.

The finite element model is presented in the literature using at least two separate sets of notation. The so-called B-notation is used by Zienkiewicz (41), and the N-notation is used by Mallet and Marcal (21) and by Rajasekaran and Murray (29). A useful correlation between

the two sets of notation is presented by Wood and Schrefler (39).

CHAPTER 2

PREPARATION FOR NONLINEAR ANALYSIS

2.1 Sources of Nonlinearity

Nonlinearity may result from the use of a nonlinear model or from the specification of nonlinear boundary conditions. A nonlinear model may contain geometric nonlinearity, material nonlinearity, or both. Geometric nonlinearity may result from the use of a nonlinear strain-displacement relation, the formulation of equilibrium for the deformed state or both (11).

A source of geometrically nonlinear behavior exists in the specification of a compatibility or strain-displacement relation for the finite element model. The conventional linear compatibility relation for a frame element is

$$\epsilon(x,y) = \frac{du}{dx} - y\frac{d^2v}{dx^2} \quad . \tag{2.1}$$

where x and y are coordinates and u and v represent displacements of a point. Nonlinearity can be introduced into the model by using a nonlinear strain-displacement relation such as Green's strain tensor which is given by the equation

$$\epsilon(x,y) = \frac{du}{dx} - y\frac{d^2v}{dx^2} + \frac{1}{2}\left[\frac{du}{dx}\right]^2 + \frac{1}{2}\left[\frac{dv}{dx}\right]^2 \tag{2.2}$$

(see section 2.2).

Geometric nonlinearity also occurs when equilibrium is formulated for the deformed state of a structure. For this formulation, the applied forces are resisted by the deformed structure, and the deformations are caused by the applied forces. Equilibrium is achieved when the internal resisting forces due to the deformations balance the external loads.

The use of a nonlinear stress-strain or constitutive relation results in a materially nonlinear formulation. The most commonly used linear constitutive relation is Hooke's law, which defines stress as a linear function of strain. Material nonlinearity results when stress is not a linear function of strain.

Nonlinearity can also be introduced through the specification of nonlinear boundary conditions. An example of a nonlinear boundary condition is a contact problem in which a part of a structure can deflect a given distance before making contact with some other member or boundary. When this occurs, a degree of freedom becomes constrained in some manner.

This study addresses geometric nonlinearity due to a nonlinear strain-displacement relation and to the formulation of equilibrium for the deformed state.

2.2 Green's Strain

Green's strain tensor is given by

$$\epsilon_x = \frac{du}{dx} + \frac{1}{2}\left\{\left[\frac{du}{dx}\right]^2 + \left[\frac{dv}{dx}\right]^2\right\} \tag{2.3}$$

where ϵ_x is the strain parallel to the x-axis for a planar element. For small strains and small rotations, the strain at a point (x,y) is approximated by

$$\epsilon(x,y) = \frac{du}{dx} - y\frac{d^2v}{dx^2} + \frac{1}{2}\left[\frac{dv}{dx}\right]^2 \tag{2.4}$$

The derivation of the square term in Green's strain is presented in appendix C, and the development of equation 2.4 from equation 2.3 is shown in appendix D.

The term

$$\frac{1}{2}\left[\frac{du}{dx}\right]^2 \tag{2.5}$$

is neglected in the development of equation 2.4 because it is small with respect to du/dx for small strains, and it is negligible for small rotations. Neglecting this term causes the strain-displacement relation to produce nonzero strains for a rigid body rotation of an element (15,16,17).

It is important to realize the limitations of the components of a model in order to use the model correctly, so it is helpful to gain some insight into the relationship

between the rotation of an element and the resulting errors in the representation of the strain. The element in figure 2.1 undergoes a rigid body rotation. It is easily seen that

$$u(x) = x \cos\theta - x \tag{2.6}$$

and

$$v(x) = x \sin\theta \quad . \tag{2.7}$$

Thus,

$$\frac{du}{dx} = \cos\theta - 1 \tag{2.8}$$

and

$$\frac{dv}{dx} = \sin\theta \tag{2.9}$$

From equations 2.3, 2.8, and 2.9, the strain can be expressed as a function of the rotation.

$$\epsilon_x = (\cos\theta - 1) + \frac{1}{2}(\cos\theta - 1)^2 + \frac{1}{2}(\sin\theta)^2 \tag{2.10}$$

Table 2.1 shows the values of each of the terms in equation 2.10 for various values of the angle of rotation. Since the member undergoes a rigid body rotation, the total strain remains zero for all values of the angle when all contributions are included. Thus the contribution of each term represents the amount of strain that is neglected for a given rotation when that term is neglected in the strain-displacement relation.

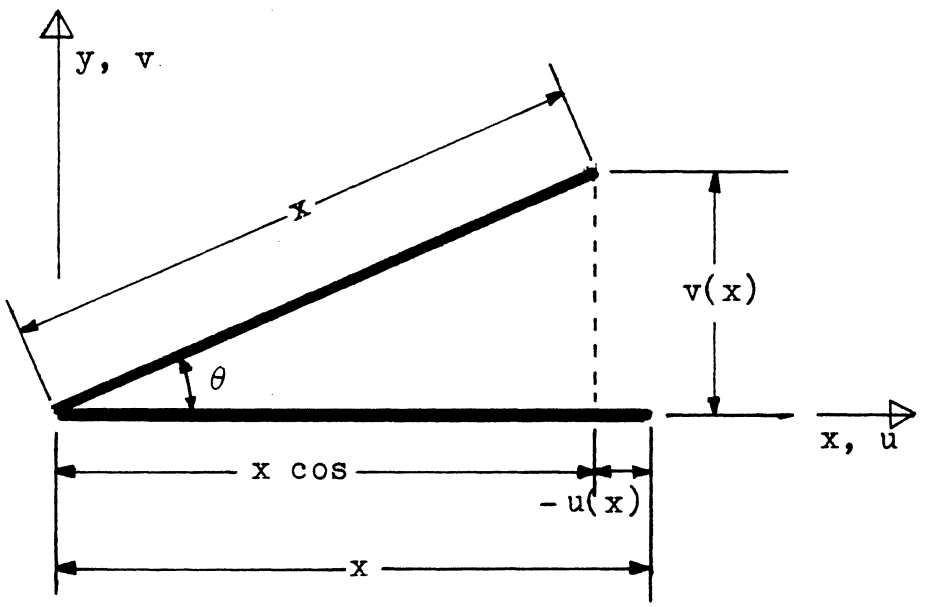


Figure 2.1 : Rigid Body Rotation of an Element

Table 2.1 : Contributions of Terms in Green's Strain For Rigid Body Rotation

θ (degrees)	θ (radians)	$\frac{du}{dx}$	$\frac{1}{2}\left[\frac{du}{dx}\right]^2$	$\frac{1}{2}\left[\frac{dv}{dx}\right]^2$
0	0.0000	0.0000	0.0000	0.0000
1	0.0175	-0.0002	0.0000	0.0002
5	0.0873	-0.0038	0.0000	0.0038
10	0.1745	-0.0152	0.0001	0.0151
20	0.3491	-0.0603	0.0018	0.0585
30	0.5236	-0.1340	0.0090	0.1250
40	0.6981	-0.2340	0.0274	0.2066
50	0.8727	-0.3572	0.0638	0.2934
60	1.0472	-0.5000	0.1250	0.3750
70	1.2217	-0.6580	0.2165	0.4415
80	1.3963	-0.8264	0.3414	0.4849
90	1.5708	-1.0000	0.5000	0.5000

2.3 Total and Updated Lagrangian Formulations

A geometrically nonlinear analysis can be performed using a total Lagrangian, a Eulerian, or an updated Lagrangian formulation (5,10). The reference frame remains the same throughout the analysis for the total Lagrangian formulation. Displacements, differentiations, and integrations are referenced to the reference frame of the original, undeformed configuration (5). In the Eulerian formulation, the reference frame follows the deformations of the structure in such a way that the coordinates of every point in the structure remain the same (5). Usually the Eulerian formulation is implemented as an updated Lagrangian formulation (5). The updated Lagrangian formulation references the displacements of the current configuration to the last equilibrium configuration. Thus, each equilibrium configuration becomes the initial state from which the next equilibrium configuration is determined. A total Lagrangian formulation is used in this study.

2.4 Equilibrium Path

An equilibrium path is a set of points defined by the load level and the corresponding displacements for which the internal forces balance the externally applied loads. Some common terminology used in nonlinear analysis

is presented in figure 2.2 (14). In the figure, loading and unloading are indicated for tracing the path from left to right.

2.5 Incrementing Applied Loads

For a multi-degree of freedom system, a load distribution vector, $\bar{Q}$, is prescribed. For proportional loading, the total applied load vector, Q, is incremented by incrementing a load multiplier, λ, so that the relation between the applied load and the load distribution is

$$Q = \lambda \bar{Q} \tag{2.11}$$

The ratios between different applied loads remain constant during proportional loading (5). For example, the loads in figures 2.3a and b could be represented by those in figure 2.3c for $\lambda = 10$ and $\lambda = 20$ respectively. For the numbering scheme for the degrees of freedom represented in figure 2.3d,

$$\bar{Q} = \begin{bmatrix} 3.0 \\ -1.0 \\ 0.5 \end{bmatrix} \tag{2.12}$$

In nonlinear analysis, the load is usually applied in small increments to prevent the solution process from jumping to an unstable configuration (41).

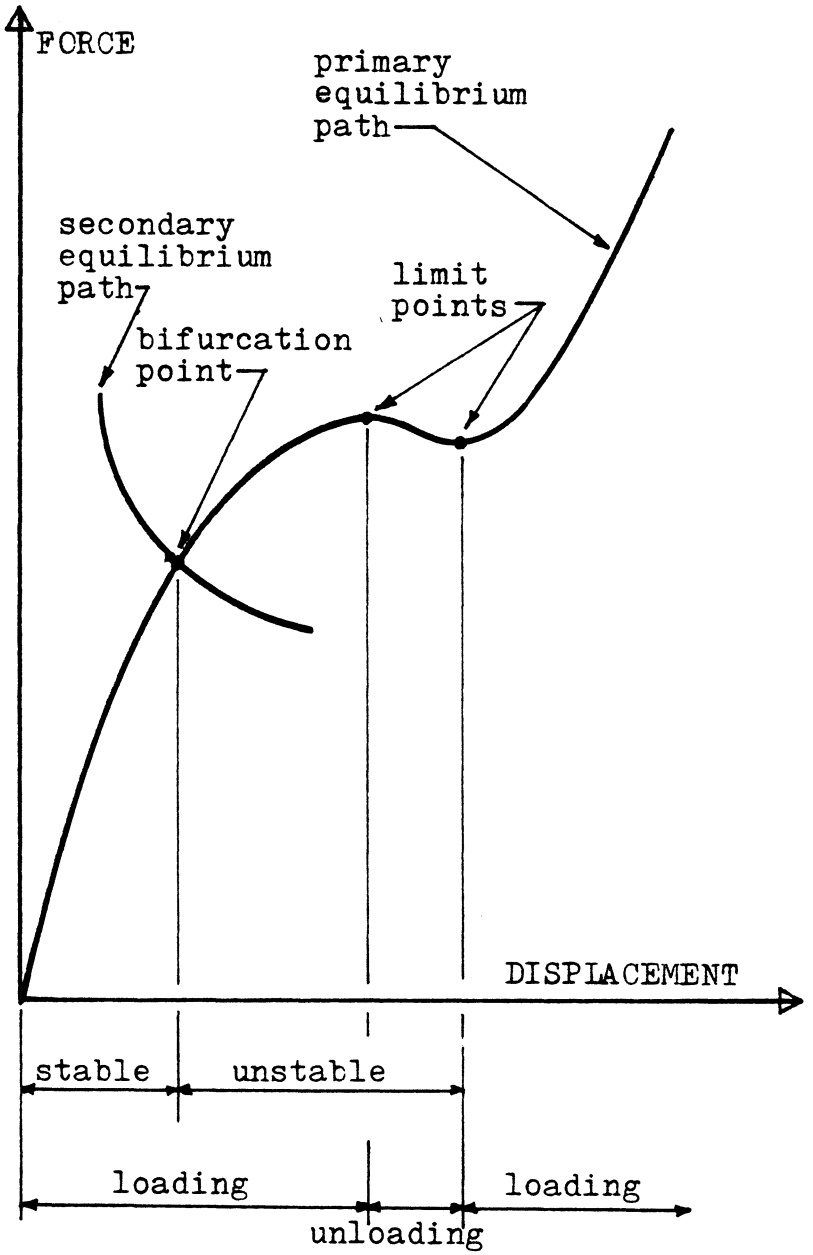


Figure 2.2 : Equilibrium Path

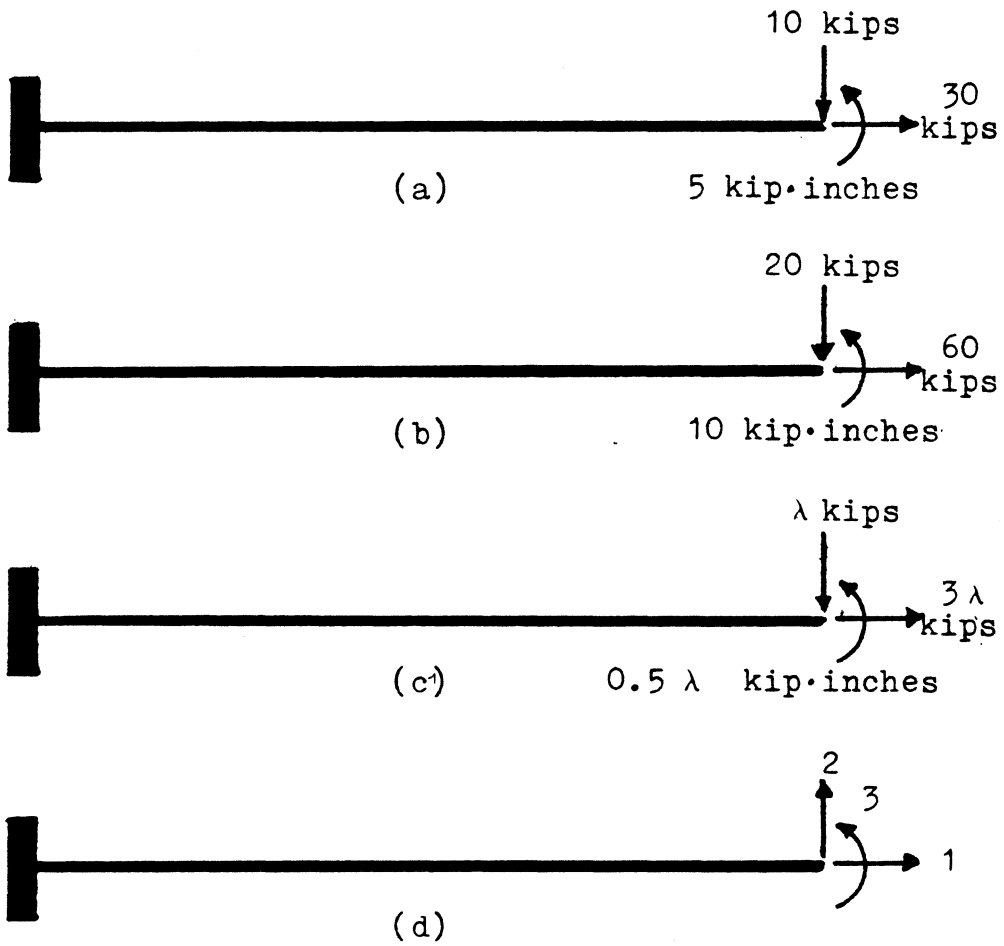


Figure 2.3 : Incremented Loads

2.6 Newton-Raphson Method

A nonlinear analysis is generally performed using a linear system stiffness matrix, and thus becomes a piecewise linear analysis in its implementation. The Newton-Raphson method is one technique used to perform a nonlinear analysis. The Nassi-Schneiderman diagram(7,43) in figure 2.4 shows the general algorithm for the Newton-Raphson method.

It is very helpful to consider a one degree of freedom example to gain physical insight into the method. In figure 2.5 the present equilibrium point is point 0, and the next equilibrium point, N, is sought. The configuration of the structure is known at point 0, so it is possible to generate the tangent stiffness, K_T^0, at that point. For a one degree of freedom example, the tangent stiffness defines the slope of the line tangent to the equilibrium path at the point where the stiffness is generated (see section 2.8). The residual force is computed by

$$R^k = Q^N - F^k \tag{2.13}$$

and

$$K_T^k \Delta q^k = R^k \tag{2.14}$$

is solved for Δq^k. The total displacement at point

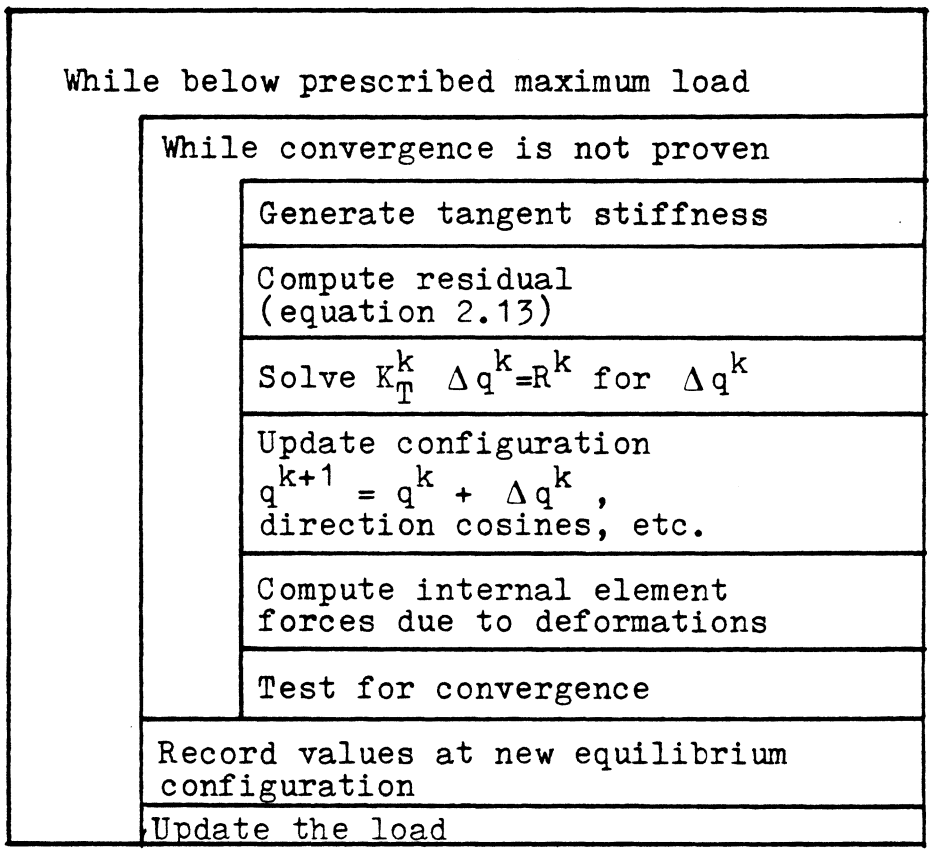


Figure 2.4 : Algorithm for the Newton-Raphson Method

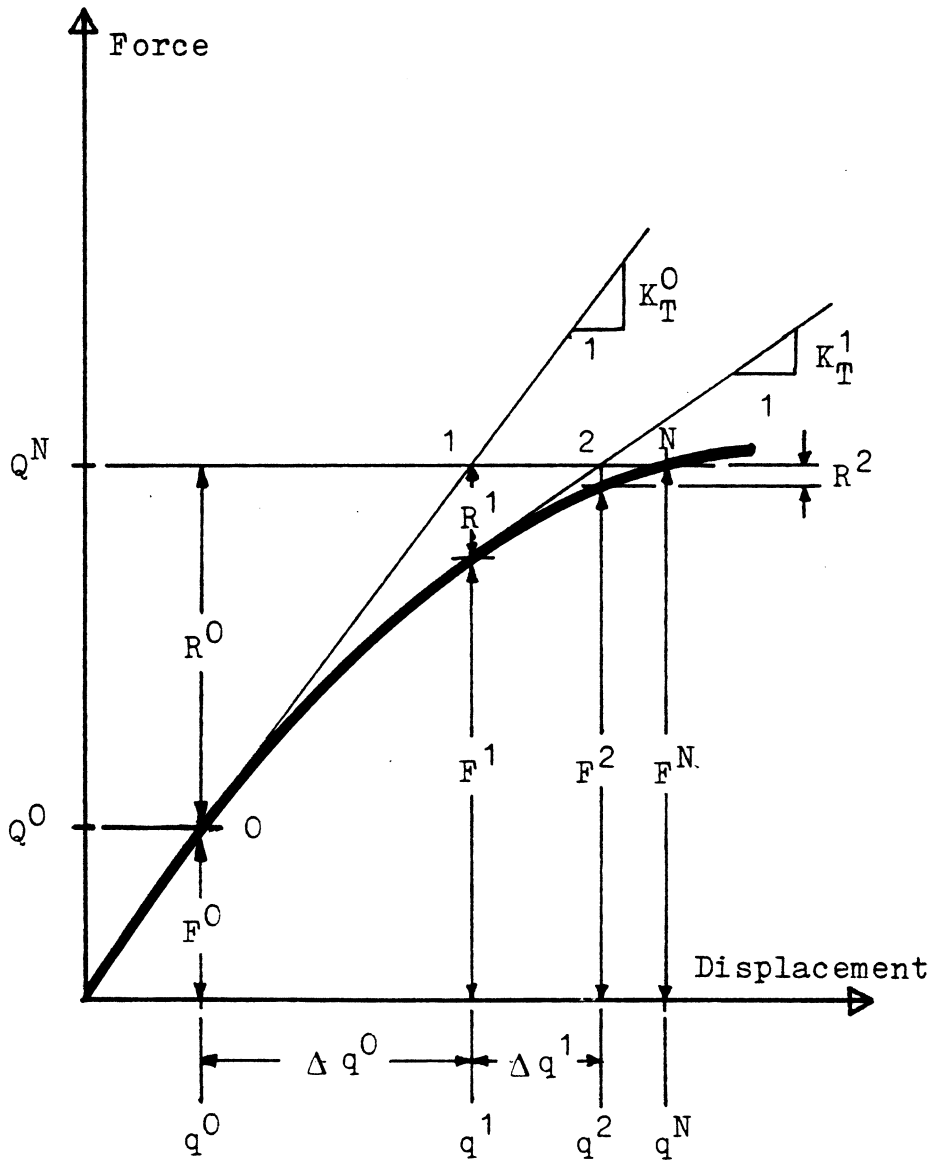


Figure 2.5 : Physical Interpretation of the Newton-Raphson Algorithm

k+1 is determined by

$$q^{k+1} = q^k + \Delta q^k \tag{2.15}$$

The internal element force due to the deformation, F^{k+1}, can be computed since the configuration at point k+1 is known and is given by q^{k+1}. If point k+1 is not sufficiently close to point N as determined by the convergence criteria (see section 2.14), the preceding steps are repeated from point k+1 to find point k+2.

For systems with more than one degree of freedom, the tangent stiffness is a square matrix, and the external loads, internal forces, and residual forces are column vectors. Equation 2.14 becomes

$$K_T \Delta q = R \tag{2.16}$$

where K_T is an n by n matrix, Δq and R are each column vectors of length n, and n is the number of degrees of freedom in the system.

The solution of equation 2.16 is the solution of a set of linear equations. The solution process often involves a factorization of K_T (see section 2.12). The generation and factorization of the tangent stiffness matrix can be costly for large systems, so the Newton-Raphson method is often modified so that K_T is only updated and factored after a predetermined number of steps are taken or after each new equilibrium point is found.

Figure 2.6 shows a physical interpretation of the modified Newton-Raphson method for one degree of freedom. It is obvious in this figure that the modified Newton-Raphson method generally requires that more steps be taken to reach a new equilibrium point, so some efficiency may be lost while some is gained by using the modified method rather than the Newton-Raphson method. Additionally, the modified Newton-Raphson method may not converge to the next equilibrium point at all for some structures (5).

The Newton-Raphson method is often unreliable when a postbuckling path must be traced. It may (figure 2.7a) or may not (figure 2.7b) converge to a point beyond a limit point. In either case, however, the response in the vicinity of a limit point is often poorly represented (see figure 2.8) unless the incrementation of the load is small (in comparison to the incrementation required to accurately represent straighter portions of the equilibrium path). The Newton-Raphson method cannot generally be used to trace an equilibrium path from a region of loading through a limit point into a region of unloading or vice versa. A method which overcomes some of these problems is the modified Riks/Wempner method.

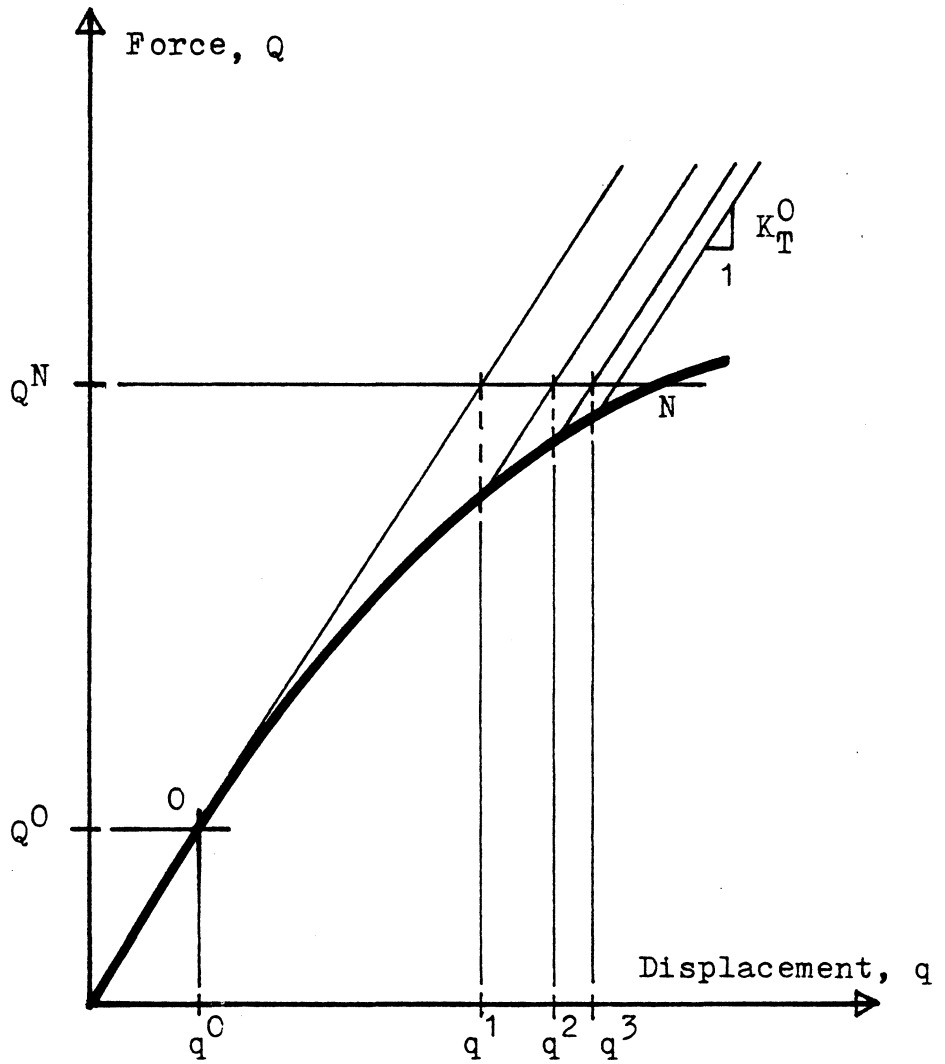


Figure 2.6 : Physical Interpretation of the Modified Newton-Raphson Method

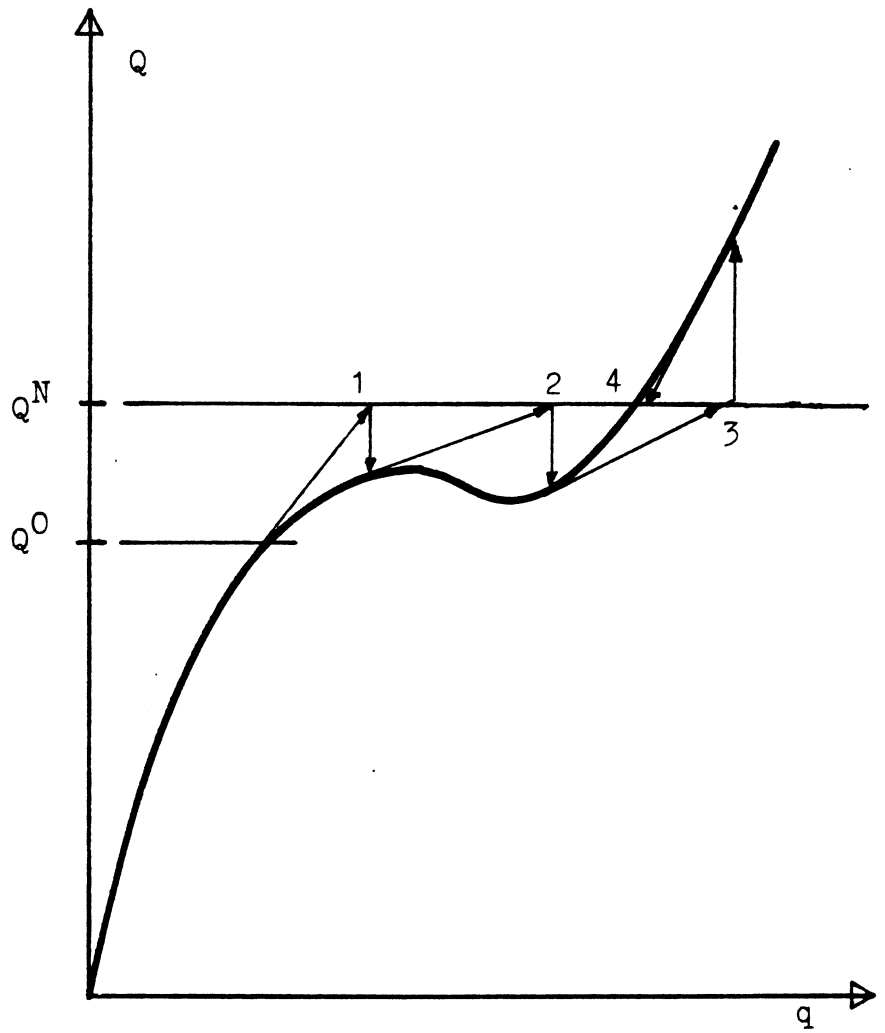


Figure 2.7 a : Newton-Raphson Method Converging Beyond a Limit Point

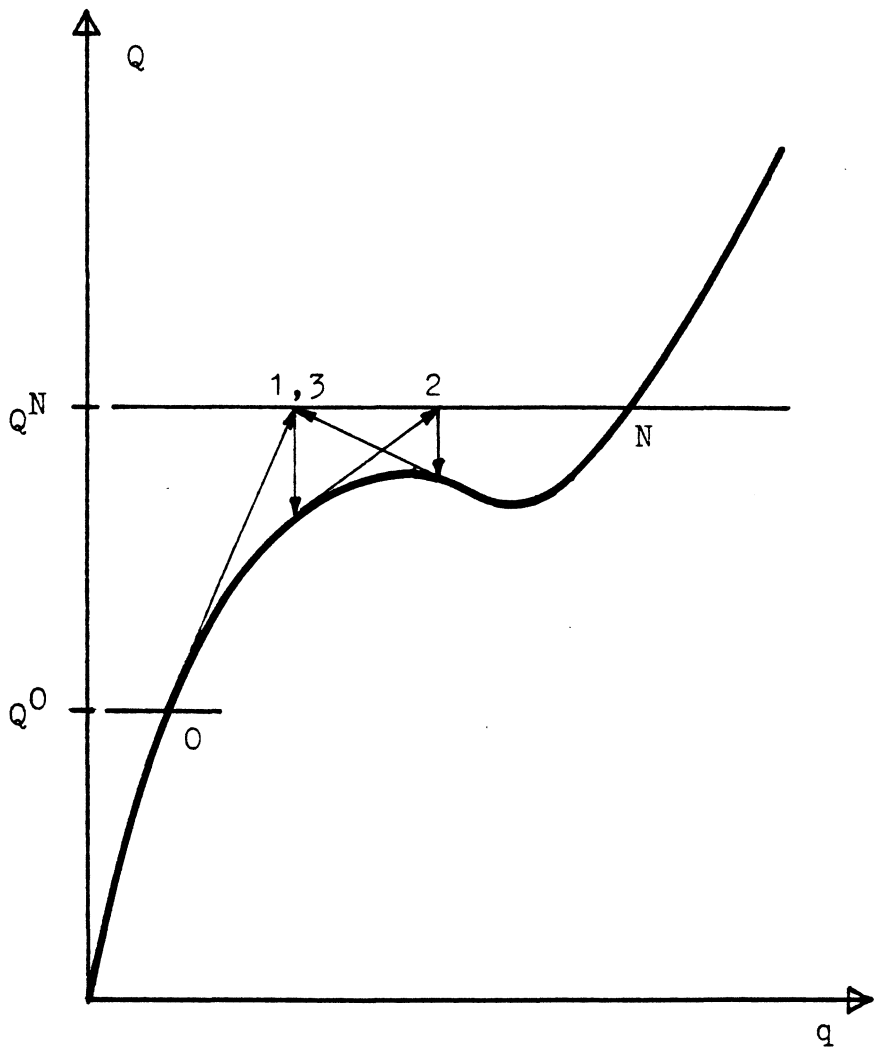


Figure 2.7 b : Newton-Raphson Method Not Converging Beyond a Limit Point

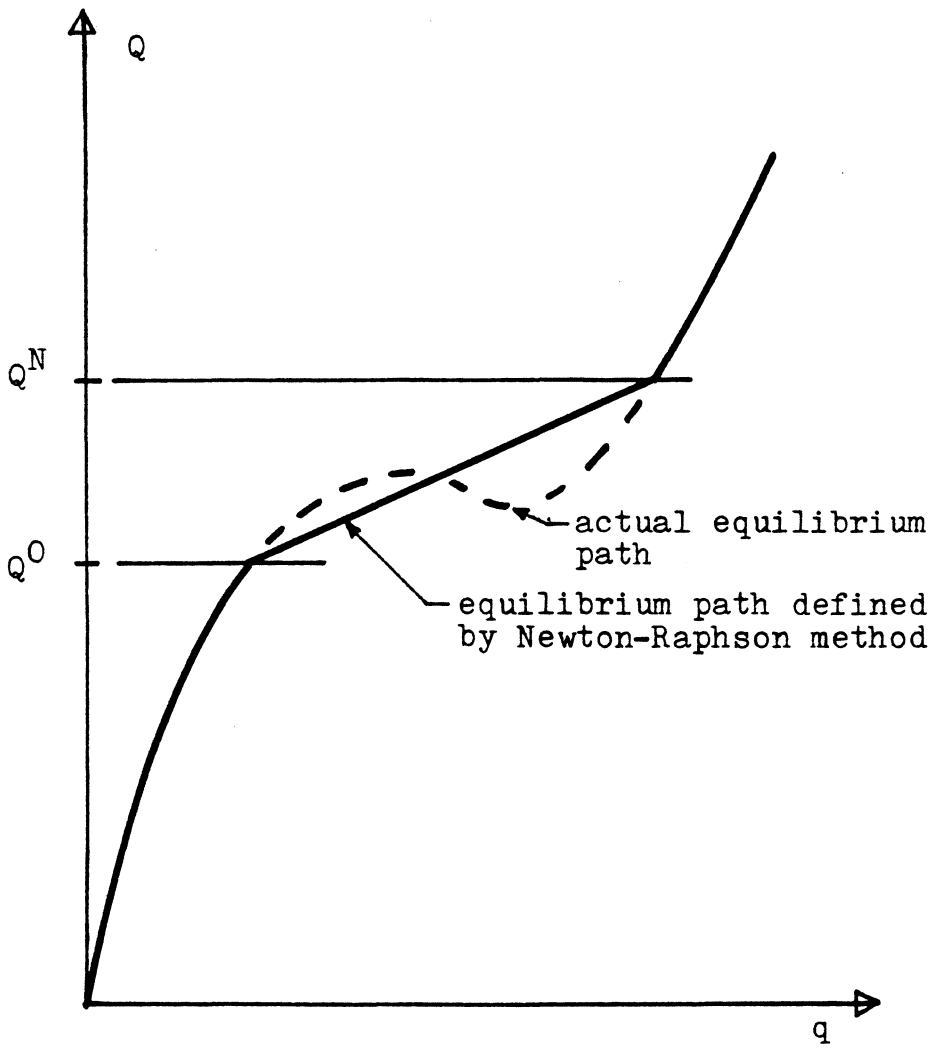
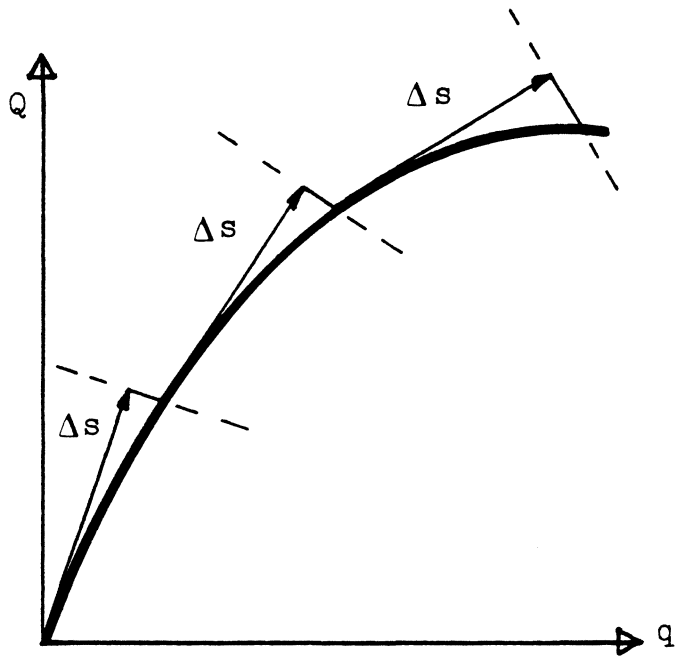


Figure 2.8 : Equilibrium Path Defined By the Newton-Raphson Method in the Vicinity of Limit Points

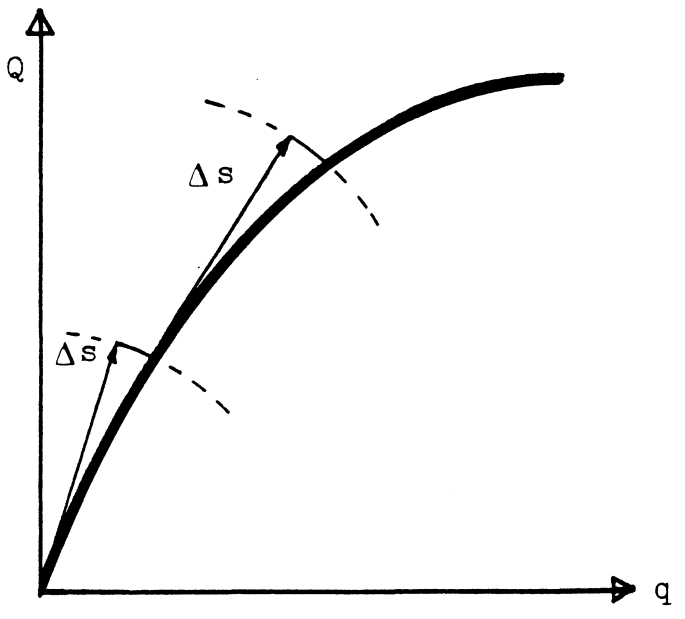
2.7 Modified Riks/Wempner Method

The modified Riks/Wempner method has been shown to be an effective technique for tracing an equilibrium path beyond a limit point into a postbuckling range (13). The modified Riks/Wempner method searches for a new equilibrium point by moving away from the previous equilibrium point along a tangent to the equilibrium path for a specified distance, Δs. The method then iterates along a normal to the tangent or along a sphere (a circle for a one degree of freedom problem) until an intersection with the equilibrium path is attained. For a one degree of freedom problem, figure 2.9a shows iteration along a normal and figure 2.9b shows iteration along a circle with the previous equilibrium point as its center. Iteration along a normal is shown in more detail with updating of the tangent stiffness in figure 2.10a and without updating in figure 2.10b. Only iteration along a normal will be considered here as it is sufficient for the purposes of this study. More information about iteration on the sphere is available from other sources (13, 19, 37).

The general algorithm of the modified Riks/Wempner method is presented in figure 2.11. The notation used in this figure corresponds to the notation of reference 13.

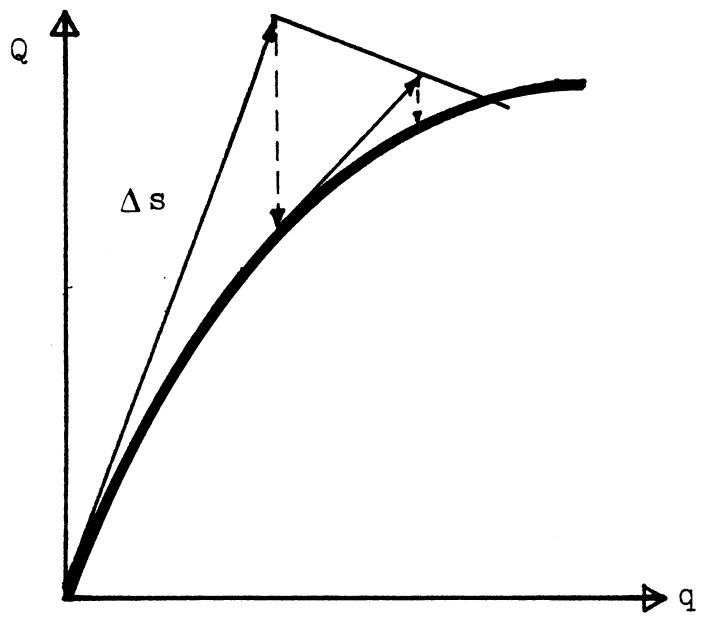


(a) Iteration on the Normal

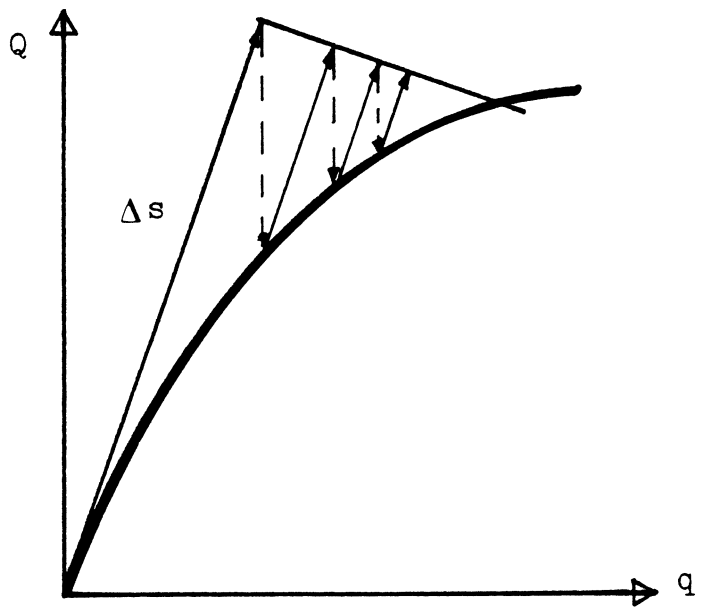


(b) Iteration on the Circle

Figure 2.9 : Modified Riks/Wempner Method



(a) With Updating



(b) Without Updating

Figure 2.10 : Modified Riks/Wempner Iteration Along a Normal

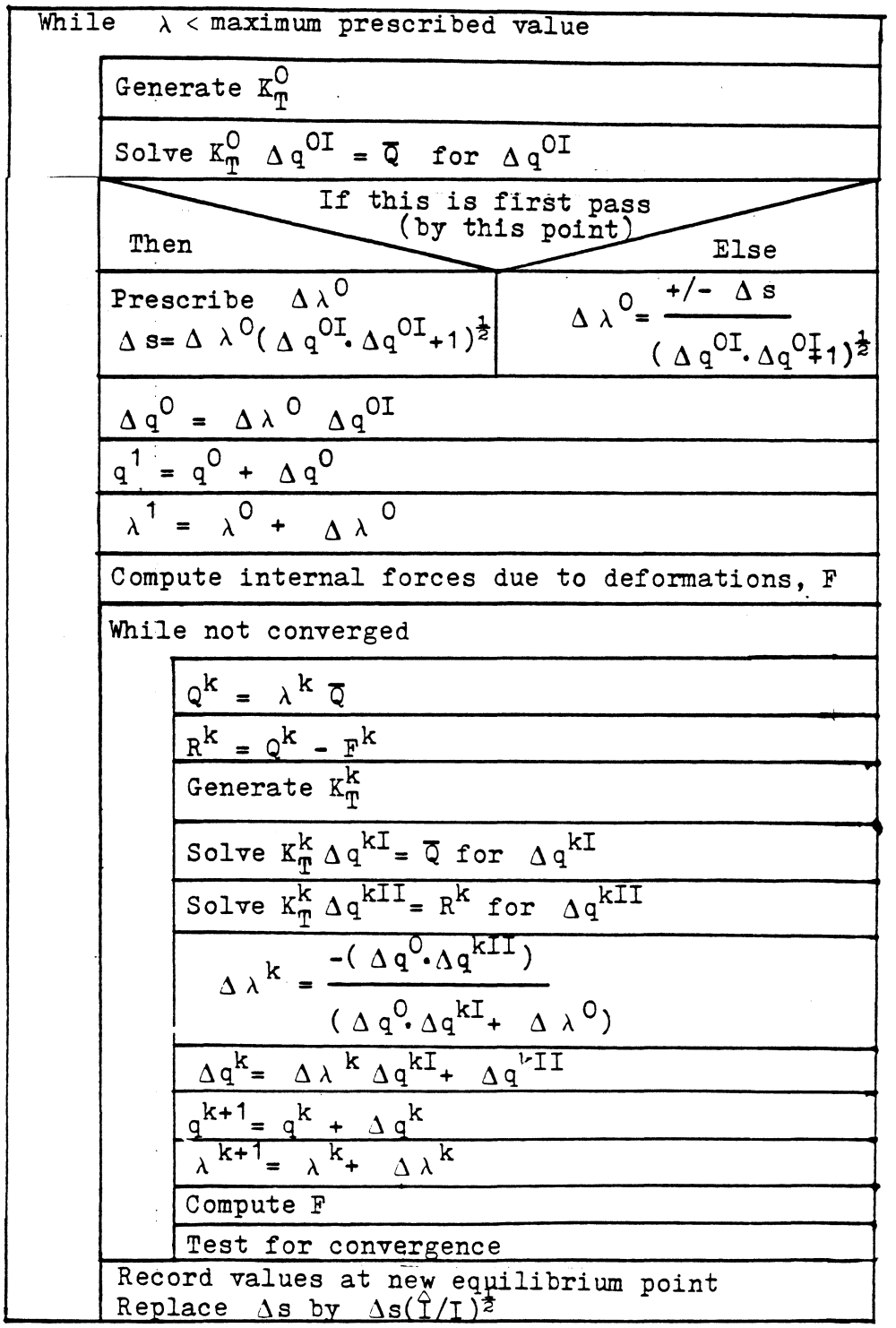


Figure 2.11 : Modified Riks/Wempner Algorithm

In figure 2.11, the iteration "while $\lambda <$ maximum presribed value" represents the iteration from one equilibrium point to the next. The iteration "while not converged" represents the iteration along the normal. Note that in the first pass through the algorithm, $\Delta\lambda$ is prescribed, and Δs is computed from it. Thereafter, $\Delta\lambda$ is computed from Δs, with $\Delta\lambda$ positive for loading and negative for unloading. This procedure is a key difference between the Newton-Raphson and the modified Riks/Wempner methods. While the Newton-Raphson method proceeds along the curve by a specified load increment, the modified Riks/Wempner method moves along the equilibrium path by a specified arc length, Δs. Another key difference is that the Newton-Raphson method iterates toward a new equilibrium point along a path of constant loading, while the modified Riks/Wempner method iterates along a line normal to the initial tangent. These differences are the characteristics that permit the modified Riks/Wempner method to trace the path through limit points.

Note that Δq is separated into two contributions such that

$$\Delta q = \Delta\lambda \; \Delta q^{kI} + \Delta q^{kII} \qquad (2.17)$$

A physical interpretation of this separation is given in appendix E. This separation preserves the symmetry and bandedness of the stiffness matrix (see reference 13).

The modified Riks/Wempner iteration is forced to proceed along a normal to the initial tangent by the constraint equation:

$$\Delta\lambda^{k}=-(\Delta q^{0}\,\Delta q^{kII})/(\Delta q^{0}\,\Delta q^{kI}+\Delta\lambda^{0}) \qquad (2.18)$$

This equation is developed in appendix E, which gives some insight into the actual procedure involved in the iteration along the normal.

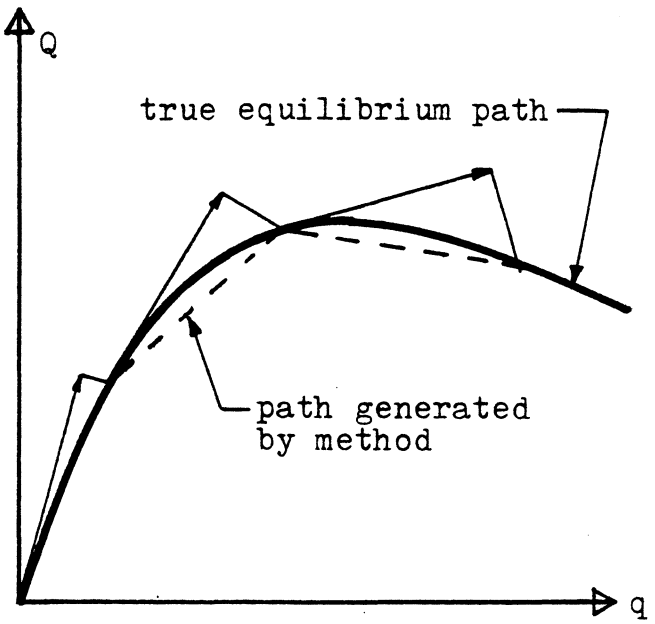
Notice that the arc length, Δs, is scaled after convergence to a new equilibrium point is achieved (figure 2.11). The purpose of scaling is to decrease the arc length in regions of high curvature so that the equilibrium path obtained is well-defined (see figures 2.12 a and b). The scaling equation is

$$\Delta s' = \Delta s\,(\,\hat{I}\,/\,I\,)^{\frac{1}{2}} \qquad (2.19)$$

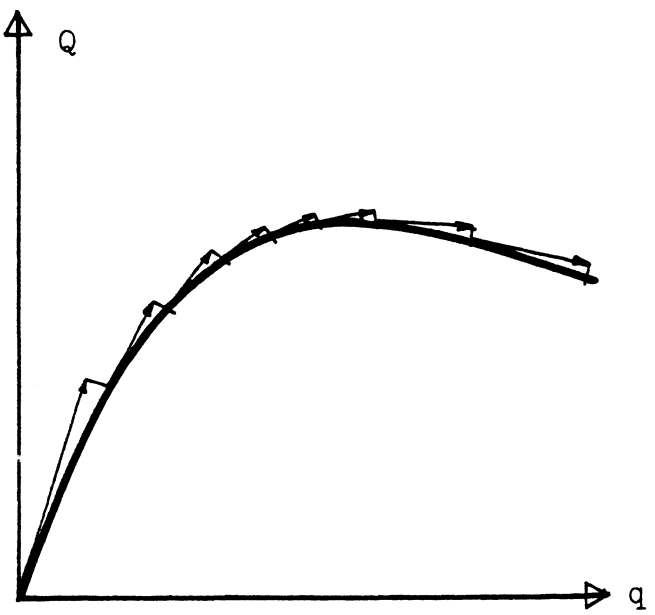
where $\Delta s'$ replaces Δs for the next iteration, $\hat{I}$ is the number of iterations that is desired to converge to a new equilibrium point, and I is the actual number of iterations that was required to converge to the last equilibrium point.

2.8 Definition of the Tangent Stiffness

Several components of the solution techniques for nonlinear analysis should be studied in more detail. The tangent stiffness matrix, for example, merits a more rigorous treatment than has already been given.



(a) Without Updating Arc Length



(b) With Updating Arc Length

Figure 2.12 : Effect of Updating the Arc Length in the Modified Riks/Wempner Method

The definition of the system tangent stiffness arises from a Taylor series expansion of the generalized forces about the generalized displacements (5, 7, 10). Recall that for geometric nonlinearity, the forces are nonlinear functions of the displacements.

$$F = F(q) \tag{2.20}$$

The Taylor series expansion gives

$$F(q+\Delta q)=F(q)+F_{,q}\,\Delta q + \text{higher order terms} \tag{2.21}$$

where $F_{,q}$ is the partial derivative of F with respect to q. Since

$$\Delta F = F(q+\Delta q) - F(q) \ , \tag{2.22}$$

$$\Delta F = F_{,q}\,\Delta q \tag{2.23}$$

(neglecting higher order terms). In equation 2.23, ΔF is the residual force and $F_{,q}$ is the tangent stiffness matrix.

For a multi-degree of freedom system,

$$F_i = F_i(q_1, q_2, q_3, \cdots , q_n) \tag{2.24}$$

and the Taylor series expansion gives

$$\Delta F_i = \sum_{j=1}^{n} F_{i,q_j}\,\Delta q_j \tag{2.25}$$

and the tangent stiffness matrix is merely the transpose of the Jacobian (9, 24).

$$\Delta F = J^T\,\Delta q \tag{2.26}$$

where the entry in the i^{th} row and j^{th} column of J^T is

$$J^T_{ij} = F_{i,q_j} \tag{2.27}$$

The preceding argument is easily correlated with the Newton-Raphson method (5).

2.9 Characteristics of the Tangent Stiffness Matrix For Stability

The inertia of a matrix is a three valued column vector whose entries are the numbers of positive, negative, and zero eigenvalues of the matrix (13). A matrix that has all positive eigenvalues is positive definite; a matrix that has all eigenvalues greater than or equal to zero is positive semidefinite; a matrix having positive, negative, and zero eigenvalues is indefinite; a matrix with only negative and zero eigenvalues is negative semidefinite; a matrix with all negative eigenvalues is negative definite. Limit and bifurcation points are collectively called critical points. (1, 14, 20).

At each critical point, at least one eigenvalue of the tangent stiffness matrix becomes zero. As a critical point is passed, an eigenvalue changes sign either from positive to negative or from negative to positive. For stability to exist, the tangent stiffness matrix is positive definite, as is proven in appendix F. Conversely,

if the stiffness matrix is not positive definite, the structure is unstable.

When the stiffness matrix is factored in the solution process, the signs of the entries of the diagonal matrix, D, often correspond to the signs of the eigenvalues (the LDU factorization is not mathematically guaranteed for a nonpositive definite matrix; see section 2.12 and ref. 13). Thus, the stability characteristics of a structure may often be determined from D for a given equilibrium point. A change in sign of an entry of D signals the crossing of a critical point as an equilibrium path is traced.

2.10 Tangent Stiffness as a Predictor

The tangent stiffness matrix is used in the Newton-Raphson and the modified Riks/Wempner methods as a predictor of the structural behavior. The tangent stiffness provides an initial estimate of the next equilibrium point, but the point itself is defined by internal element force equations, not by the tangent stiffness. The secant stiffness matrix is a representation of internal element forces due to member deformations in matrix form (see figure 2.13). Thus, the correct equilibrium path can be obtained even if the tangent stiffness matrix being used is a poor predictor, provided that the secant stiffness is correct. Figures 2.14 a and b depict the

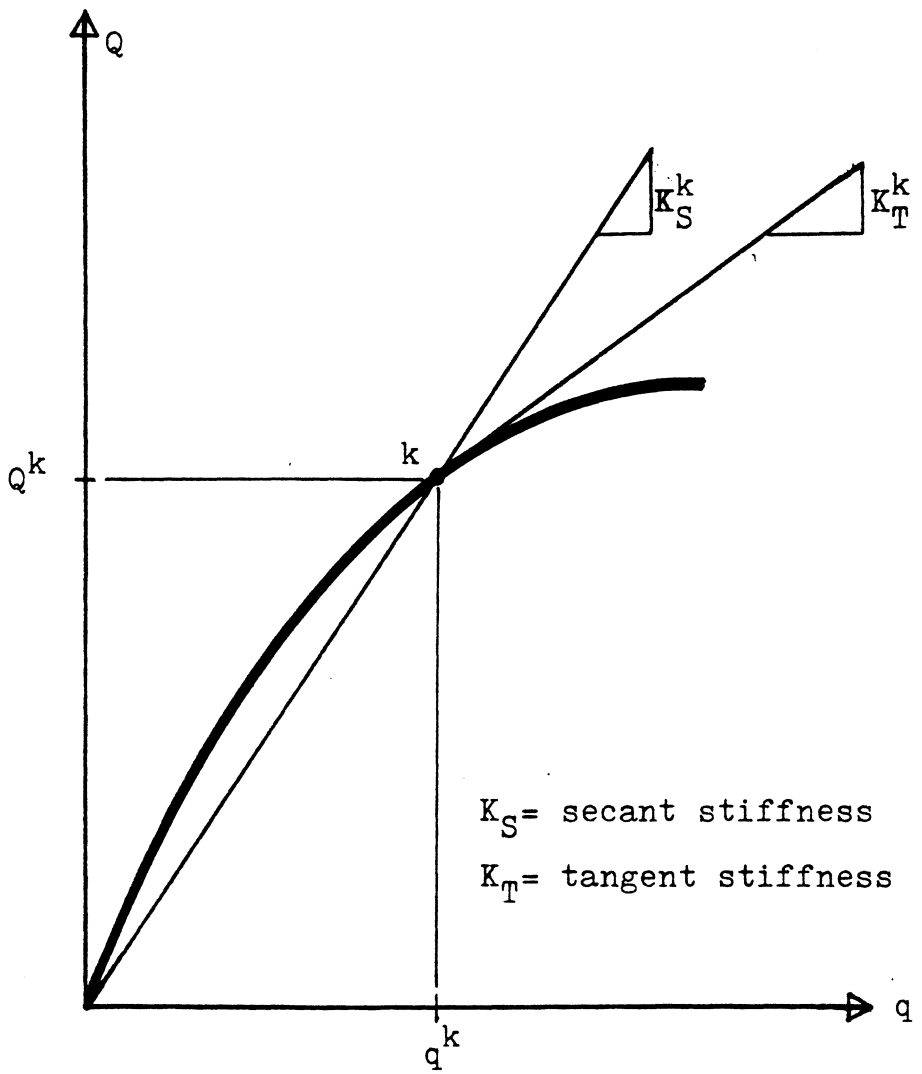
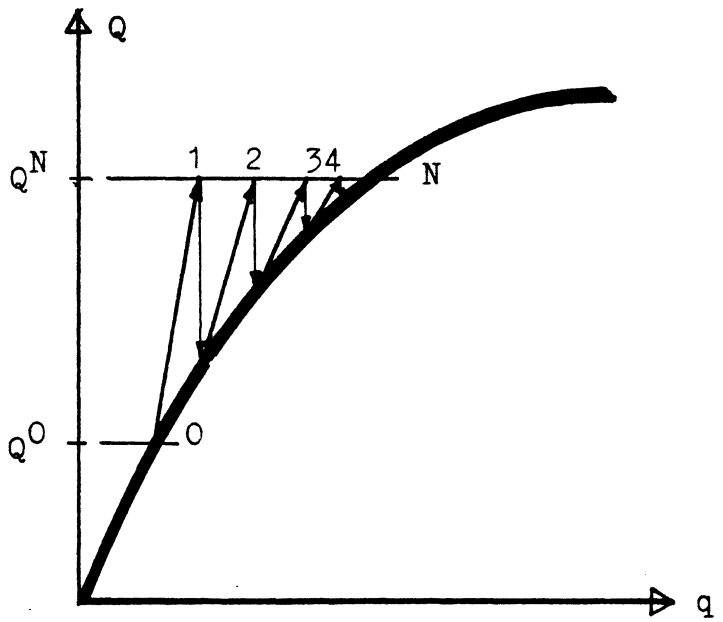
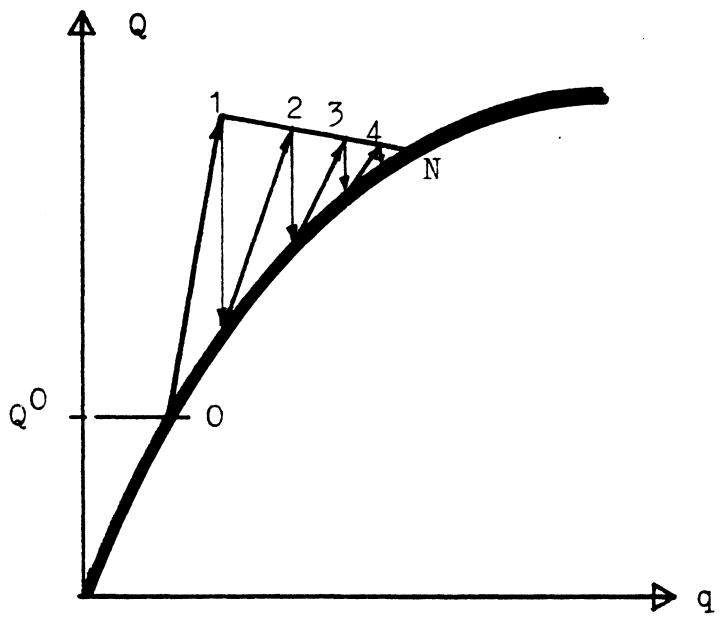


Figure 2.13 : Tangent and Secant Stiffnesses



(a) Newton-Raphson Method

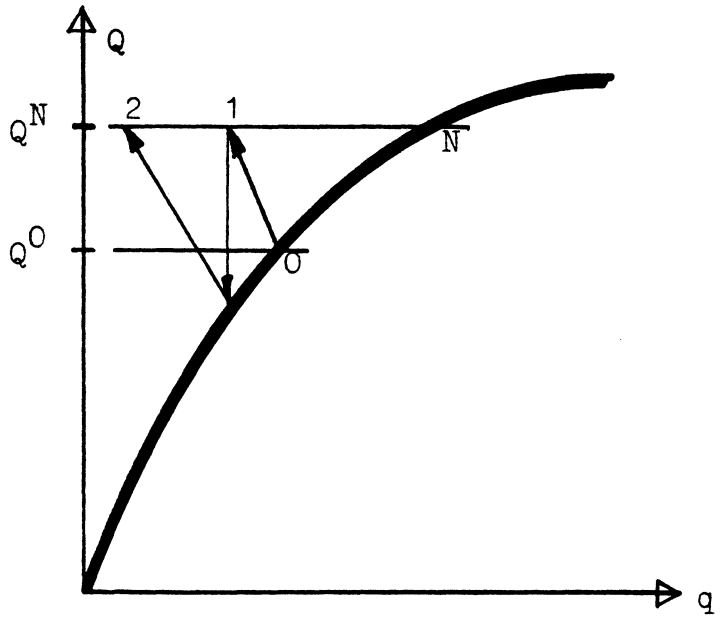


(b) Modified Riks/Wempner Method

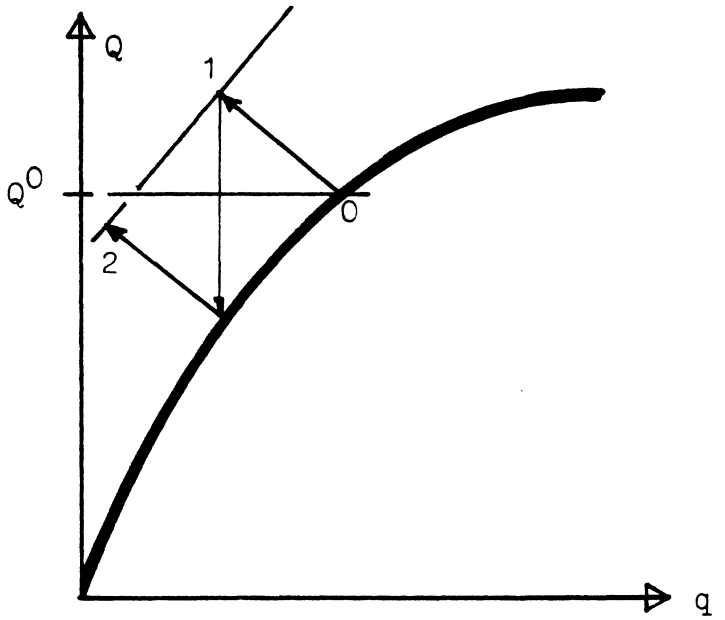
Figure 2.14 : Poor Representation of the Tangent Stiffness

possible behavior of the Newton-Raphson and the modified Riks/Wempner methods respectively when the representations of the tangent stiffnesses are poor. Notice that as long as the internal force equations are correct, both methods can converge to correct equilibrium points, although convergence is naturally slower than it would be for a good predictor. Convergence is not guaranteed, however, for a predictor that is not a true tangent. Figures 2.15 a and b show predictors which are so poor that convergence to a meaningful equilibrium point may never be achieved. Additionally, a poor predictor may cause characteristics of the equilibrium path to be missed, as is shown in figure 2.16. Note, however, that the equilibrium point obtained is valid because it is determined by element force equations and convergence criteria.

In figure 2.16, the "tangent stiffness" does not detect instability. The one dimensional representation of a tangent that predicts instability beyond a limit point has a negative slope. The characteristics of the tangent stiffness are the chief indicators of stability or lack of it (see section 2.9). The presence of a limit point is signaled by the shape of the equilibrium path, but the presence of a bifurcation point is determined by the detection of a negative eigenvalue. If the



(a) Newton-Raphson Method



(b) Modified Riks/Wempner Method

Figure 2.15 : Poor Representation of the Tangent Stiffness Without Convergence

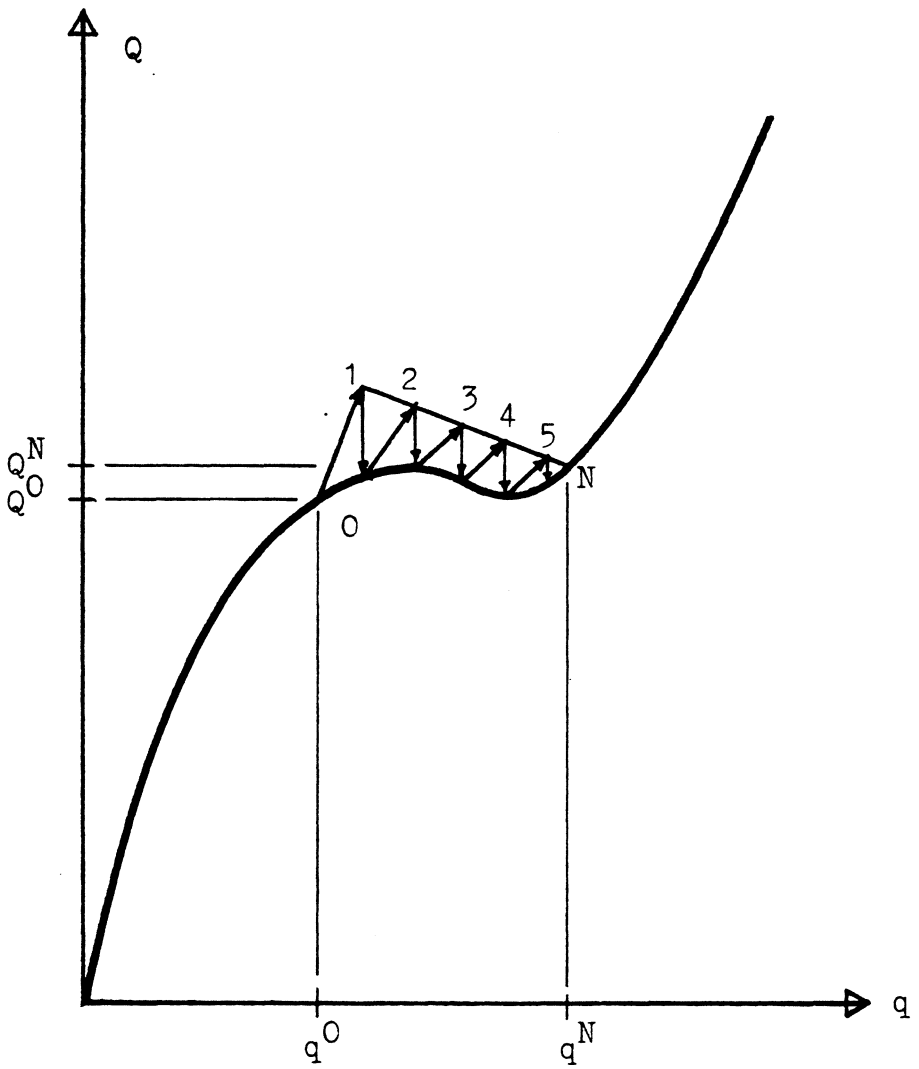


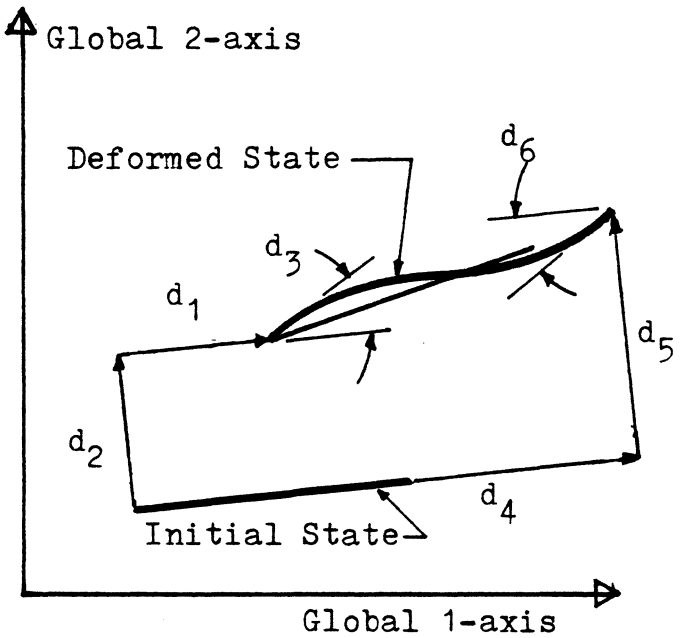
Figure 2.16 : A Poor Representation of the Tangent Stiffness for the Modified Riks/Wempner Method

full representation of the tangent stiffness matrix is not used, it is questionable whether detection of a bifurcation point is guaranteed. It is recommended procedure to use the full representation of the tangent stiffness when the behavior of the structure is not known beforehand (41).

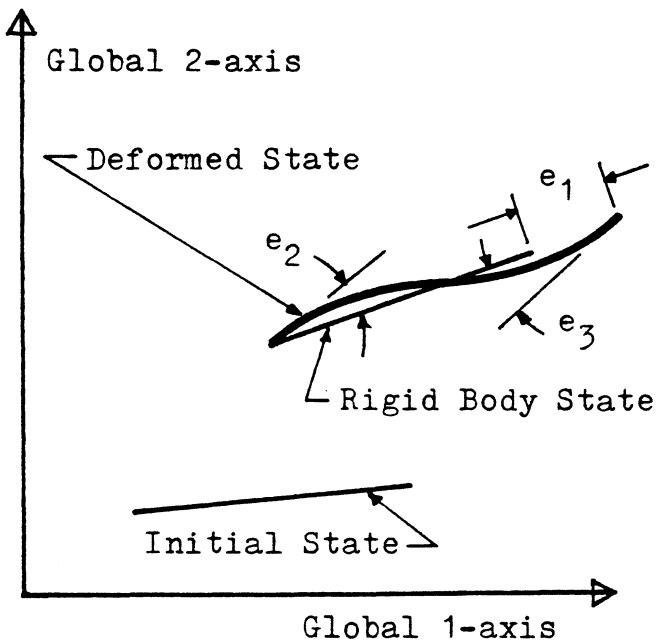
2.11 Convected Coordinate Systems

The beam-column model is developed using convected coordinates (24). For convected coordinates, also called corotational coordinates or deformation displacements, nodal displacements are referenced to a rigid body state rather than to the initial state of the member (4, 11). The rigid body state of a member is an undeformed configuration lying on the chord of the deformed member and having its a-end coincident with the a-end of the deformed state. Since there are no deformations caused by moving from the initial state to the rigid body state, the three deformation displacements of a member identify all of the contributions to the strain energy of that member. Figure 2.17 shows full local coordinates (2.17a) and convected coordinates (2.17b).

It is important to note that the standard local-to-global transformation does not apply to a tangent stiffness matrix developed using a convected coordinate



(a) Full Local Coordinates



(b) Convected Coordinates

Figure 2.17 : Local Coordinate Systems

system (35). The transformation matrix does not remain constant during deformation, so the configuration dependency of the transformation must be considered when the incremental formulation is developed. In particular, when the relation between six global forces, F, and three local forces for a convected coordinate system, p, for a frame member,

$$F = B\ p \tag{2.28}$$

is incremented, the following is obtained

$$\Delta F = B\ \Delta p + \Delta B\ p \tag{2.29}$$

In contrast, for a conventional system with six local forces, f, the relation

$$F = \bar{\lambda}\ f \tag{2.30}$$

has the incremental form

$$\Delta F = \bar{\lambda}\ \Delta f \tag{2.31}$$

where

$$\bar{\lambda} = \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}$$

and

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

where c_1 and c_2 are direction cosines as in reference 7.

2.12 Equation Solvers

Recalling that nonlinear analysis is usually implemented as a series of linear steps, a method is needed to solve systems of linear equations. It is advantageous to use a method which can exploit the symmetry and sparsity that are characteristic of the tangent stiffness matrix in many cases (7).

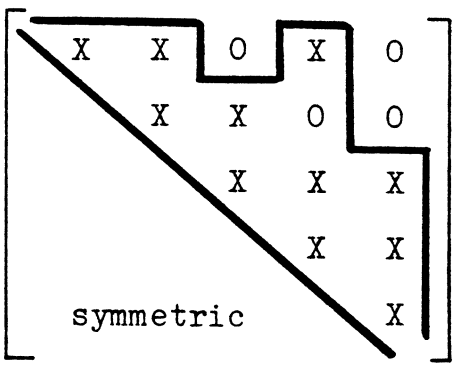
The equation solving algorithm chosen for this study and described herein is presented by Bathe (1). The solver performs Gauss elimination in three parts (which have been separated into three subroutines for the computer program used in this study). The first step consists of finding the L D U factorization of the tangent stiffness matrix, where L is a lower triangular matrix with values of unity on the diagonal, U is the transpose of L, and D is a diagonal matrix containing the pivot elements from the Gauss elimination procedure. This factorization process must be performed each time the tangent stiffness matrix is updated. The second and third steps reduce the load vector and use back substitution to solve for the displacements.

The Gauss elimination procedure is not mathematically guaranteed to give a correct result if the stiffness matrix is not positive definite, but the results are often correct nevertheless. Cholesky solution

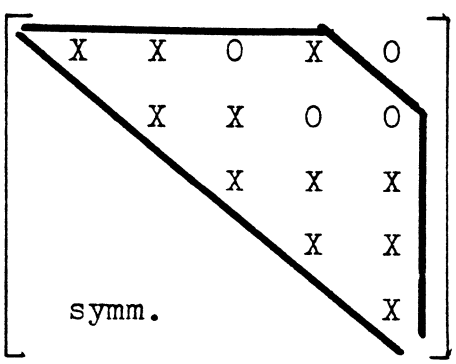
algorithms will not work for a non-positive definite stiffness matrix unless they are modified to avoid taking the square root of a negative number (5). An equation solver that is valid for a non-positive definite stiffness matrix is required for postbuckling analysis (13).

2.13 Skyline Storage

The tangent stiffnes matrix is stored using a variable band, active column, or skyline storage scheme in this study. Holzer (7) and Bathe (1) each treat this subject in detail. This scheme provides storage for all values in each column of the stiffness matrix above and including the value on the diagonal, except zeros which lie above the uppermost nonzero value. In figure 2.18a the values between the heavy lines are the values that are stored by a skyline storage scheme. An "X" represents a nonzero value, and a "0" represents a zero value. Notice that the zero value in the fourth column of the second row is stored. A zero value having a nonzero value above it in the same column must be processed by the algorithm since it may become nonzero during the solution process. It is not necessary to process zeros lying beyond the skyline, however, as they will never become nonzero (5).



(a) Skyline Storage



(b) Fixed Band Storage

Figure 2.18 : Storage of Banded Matrices

The stiffness matrix is stored in a column vector so that a minimum of storage space is used. An additional vector is needed to store the new addresses of the values which would be on the diagonal of the full tangent stiffness matrix. Figure 2.19 shows the configuration of the full stiffness matrix, where each subscript indicates the storage position of that value in the one dimensional representation of the stiffness. For example, the value in the fourth column of the second row of the full stiffness matrix is stored in the eighth position in the column vector storage (see figure 2.19). The transpose of the vector storing the addresses of the diagonals is

$$MAXA^T = \begin{bmatrix} 1 & 2 & 4 & 6 & 10 & 13 \end{bmatrix} \qquad (2.32)$$

The last entry in MAXA does not actually represent the address of a diagonal term, but it is included to indicate the number of entries contained in the last active column.

Using a skyline storage scheme obviously creates savings since zeros beyond the skyline need not be stored. In figure 2.18a twelve values are stored by the skyline storage scheme. If fixed band storage is used, fourteen values must be stored (see figure 2.18b). In order to use the column vector representation of the

$$\begin{bmatrix} K_1 & K_3 & 0 & K_9 & 0 \\ & K_2 & K_5 & K_8 & 0 \\ & & K_4 & K_7 & K_{12} \\ & & & K_6 & K_{11} \\ \text{symm.} & & & & K_{10} \end{bmatrix}$$

Figure 2.19 : Stiffness Matrix Storage

stiffness matrix, however, an additional vector containing the addresses of the diagonals must be stored.

2.14 Convergence Criteria

When either the Newton-Raphson method or the modified Riks/Wempner method iterates toward a new equilibrium point, there must be some method of determining when the current point obtained by the method is a satisfactory representation of an equilibrium point. A convergence test is conducted at each new point to determine whether that point adequately represents the equilibrium point sought.

Various convergence tests are recommended by various authors (1, 5). It is important, however, to at least check the tolerance on each of the quantities used to define the equilibrium point. In other words, the convergence of both forces and displacements must be checked. Returning to a one degree of freedom example, points A and C in figure 2.20 are used to represent the configurations at points B and D respectively. It is easily seen that the force at B may be represented with sufficient accuracy by the force at A in figure 2.20a, but the displacement at A would probably not be a satisfactory representation of the displacement at B. For a structure that behaves in a similar manner as that

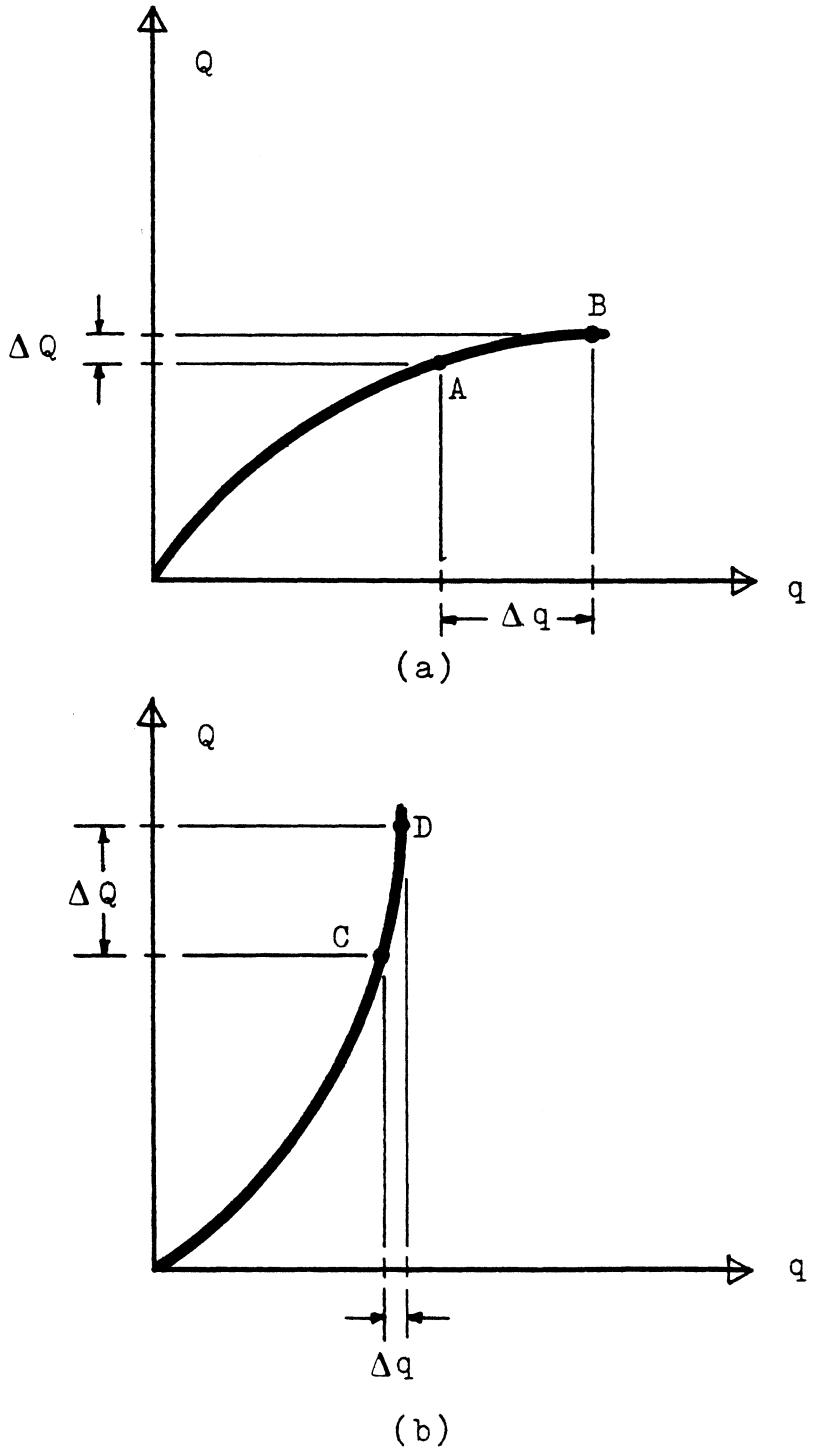


Figure 2.20 : Convergence

depicted in figure 2.20b, the displacement at C may be an adequate representation of that at D, but the force differs significantly. The convergence criteria used in this study consist of two displacement criteria and one force criterion.

It is helpful to use the following definitions of vector norms in the descriptions of the convergence criteria. The infinity vector norm of a vector, v, whose elements are

$$v_i \quad \text{for } i = 1 \text{ to } n \tag{2.33}$$

is defined as

$$\|v\|_\infty = \max_i |v_i| \tag{2.34}$$

Thus, the infinity vector norm is the element of the vector which has the largest absolute value. The Euclidean vector norm (also called the two vector norm) of a vector is

$$\|v\|_2 = \left(\sum_{i=1}^{n} (v_i)^2 \right)^{\frac{1}{2}} \tag{2.35}$$

The convergence criteria that follow are defined for the modified Riks/Wempner method. Any specializations required to apply a criterion to the Newton-Raphson method are noted in the text following the definition of that criterion.

The displacement criterion recommended by Cook (5) uses the infinity vector norm. This criterion is given as follows:

$$\frac{||\Delta q^k||_\infty}{||q^{k+1}||_\infty} \leqq \text{CPDC} \tag{2.36}$$

where the ratio is computed for the displacements and changes in displacement of one type, and CPDC is the convergence parameter for the displacement criterion recommended by Cook. Cook recommends that

$$10^{-6} \leqq \text{CPDC} \leqq 10^{-2} \tag{2.37}$$

Bathe (1) recommends a displacement criterion based on the Euclidean vector norm. This criterion is

$$\frac{||\Delta q^k||_2}{||q^{k+1}||_2} \leqq \text{CPDB} \tag{2.38}$$

where CPDB is the convergence parameter for the displacement criterion recommended by Bathe. The recommended value for this parameter (2) is

$$\text{CPDB} = 0.001 \tag{2.39}$$

Bathe (1) also recommends a convergence criterion based on the unbalanced forces. This criterion is given as

$$\frac{||Q^{k+1} - F^{k+1}||_2}{||Q^{k+1} - F^0||_2} \leqq \text{CPF} \tag{2.40}$$

where CPF is the convergence parameter for the unbalanced force criterion. For the Newton-Raphson method, Q^{k+1} is replaced by Q^N. The recommended value for the convergence parameter (2) is

$$CPF = 0.1 \tag{2.41}$$

A convergence criterion based on internal energy may also be used (1). The computer program developed for this study has the capability of implementing this criterion for the Newton-Raphson method only. A provision is included to halt iteration if divergence is detected in energy. The user should exercise caution, however, if this criterion is applied in an unstable region. The criterion is given in reference 1, and the recommended value for the convergence parameter for energy, CPE, is given in reference 2. This criterion is not discussed further here because it is not used to obtain the results of this study.

CHAPTER 3

DEVELOPMENT OF MODELS

3.1 Development of Beam-Column Model

The following development of a finite element model based on classical beam-column theory follows Oran's presentation (24) very closely. Lengthy derivations are given in appendices G, I, and J.

Using the convected coordinate system defined in figures 2.17b and 3.1, the following force-deformation relations are established from classical beam-column theory (36). See appendix G for a detailed development.

$$M_1 = \frac{EI}{L} (s_1 \theta_1 + s_2 \theta_2) \tag{3.1}$$

$$M_2 = \frac{EI}{L} (s_2 \theta_1 + s_1 \theta_2) \tag{3.2}$$

$$Q = EA (\frac{u}{L} - c_b) \tag{3.3}$$

where E, A, I, and L are material and geometric properties defined in appendix B; s_1 and s_2 are stability functions which depend on whether the axial force, Q, is compressive, tensile, or zero. The stability functions are given in appendix G and in reference 24. The length correction factor due to bowing action, c_b is defined as

$$c_b = B_1 (\theta_1 + \theta_2)^2 + B_2 (\theta_1 - \theta_2)^2 \tag{3.4}$$

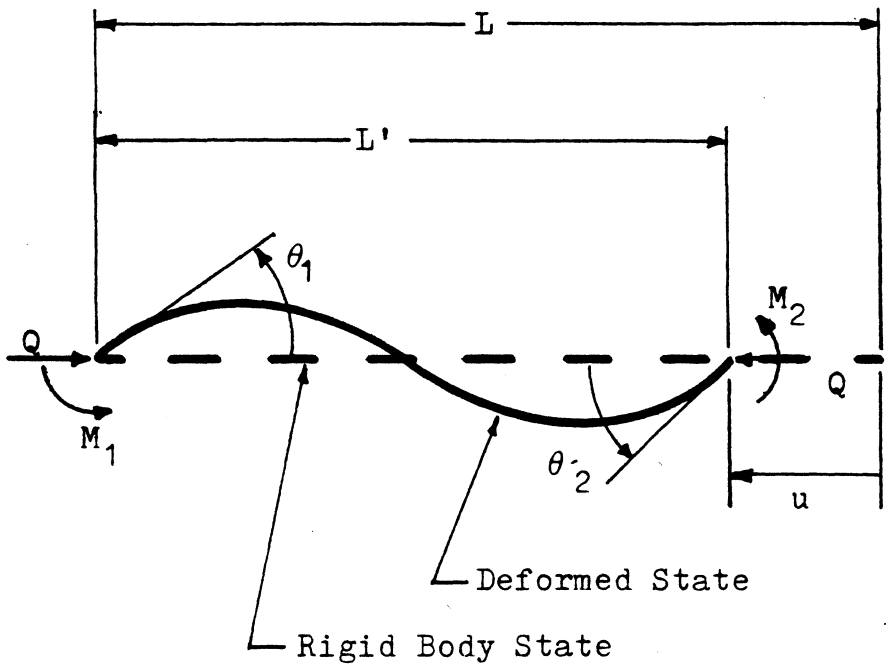


Figure 3.1 : Element Forces and Displacements for the Beam-Column Model

where B_1 and B_2 are bowing functions which are given as

$$B_1 = \frac{(s_1 + s_2)(s_2 - 2)}{8 \pi^2 QR} = - \frac{(s_1' + s_2')}{4 \pi^2} \qquad (3.5)$$

$$B_2 = \frac{s_2}{8 (s_1 + s_2)} = - \frac{(s_1' - s_2')}{4 \pi^2} \qquad (3.6)$$

where the prime superscript reresents a single differentiation with respect to QR, and QR is the axial force divided by the Euler buckling load, $\pi^2 EI / L^2$:

$$QR = \frac{QL^2}{\pi^2 EI} \quad (= \text{Oran's "q" in ref. 24}) \qquad (3.7)$$

or, from equation 3.3,

$$QR = \frac{L^2 A}{\pi^2 I} \left(\frac{u}{L} - c_b \right) \qquad (3.8)$$

Notice that the axial force, Q, is a function of the length correction factor, c_b, which is, in turn, a function of the axial force. A Newton iteration method can be used to find the root of the equation

$$QR - \frac{L^2 A}{\pi^2 I} \left(\frac{u}{L} - c_b \right) = 0 \qquad (3.9)$$

as is shown in appendix H.

Using the definition of the tangent stiffness matrix, Oran obtains a matrix which satisfies the relation :

$$t \ \Delta e = \Delta p \qquad (3.10)$$

in which

$$e = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \underline{u} \\ L \end{bmatrix} \tag{3.11}$$

$$p = \begin{bmatrix} M_1 \\ M_2 \\ QL \end{bmatrix} \tag{3.12}$$

$$t = \frac{E\,I}{L} \begin{bmatrix} s_1 + \frac{G_1^2}{\pi^2 H} & s_2 + \frac{G_1 G_2}{\pi^2 H} & \frac{G_1}{H} \\ s_2 + \frac{G_1 G_2}{\pi^2 H} & s_1 + \frac{G_2^2}{\pi^2 H} & \frac{G_2}{H} \\ \frac{G_1}{H} & \frac{G_2}{H} & \frac{\pi^2}{H} \end{bmatrix} \tag{3.13}$$

and

$$G_1 = s_1' \theta_1 + s_2' \theta_2 \tag{3.14}$$

$$G_2 = s_2' \theta_1 + s_1' \theta_2 \tag{3.15}$$

$$H = \frac{\pi^2 I}{L^2 A} + B_1' (\theta_1 + \theta_2)^2 + B_2' (\theta_1 + \theta_2)^2 \tag{3.16}$$

The prime superscript represents one differentiation with respect to QR. A complete development of t is shown in appendix I. Apparently reference 24 contains a misprint since the subscript on G in the term in the

second row of the second column of t is "1" in (24) and later given as "2" in (25). The development in appendix I verifies the value of "2" for this subscript.

The matrix, t, operates on local displacement increments to give local force increments, and it is, therefore, referred to as a local tangent stiffness matrix by Oran. Recall, however, that the matrix was developed using a convected coordinate system. Due to the reasons presented in section 2.11, the standard transformation to a global tangent stiffness matrix does not apply.

The following operations are shown in detail in appendix J. The relation between global forces and local corotational forces,

$$F = B\ p \tag{3.17}$$

has the incremental form

$$\Delta F = B\ \Delta p + \Delta B\ p \tag{3.18}$$

since the transformation matrix does not remain constant (35). From contragredience (7),

$$\Delta e = B^T\ \Delta D$$

and, recalling equation 3.10, and assembling for the system,

$$\Delta F = B\ t\ B^T\ \Delta q + \Delta B\ p \tag{3.19}$$

If $g^{(i)}$ are defined such that

$$\Delta B\ p = \left[\sum_{i=1}^{3} p_i\ (\ g^{(i)}\ )\right] \Delta q \tag{3.20}$$

A global tangent stiffness matrix, K_T, for which

$$K_T\ \Delta q = \Delta F \tag{3.21}$$

is defined by

$$K_T = B\ t\ B^T + \left[\sum_{i=1}^{3} p_i\ g^{(i)}\right] \tag{3.22}$$

See appendix J for definitions of $g^{(i)}$.

The first term on the right hand side of equation 3.22 is the conventional stiffness matrix used in linear analysis, K_0. The second term represents the initial stress stiffness matrix, K_σ (12).

The following equations are developed fully in appendix J.

$$K_T = \begin{bmatrix} G_{(1)} & G_{(2)} & G_{(4)} & -G_{(1)} & -G_{(2)} & G_{(7)} \\ & G_{(3)} & G_{(5)} & -G_{(2)} & -G_{(3)} & G_{(8)} \\ & & G_{(6)} & -G_{(4)} & -G_{(5)} & G_{(9)} \\ & & & G_{(1)} & G_{(2)} & -G_{(7)} \\ & & & & G_{(3)} & -G_{(8)} \\ \text{symm.} & & & & & G_{(10)} \end{bmatrix} \tag{3.23}$$

where the contributions to $G_{(1)}$ through $G_{(10)}$ for K_0 are :

$$
\begin{aligned}
G_{(1)} &= c_1^2\, g_1 - 2\, c_1\, c_2\, g_2 + c_2^2\, g_5 \\
G_{(2)} &= (\, c_1^2 - c_2^2\,)\, g_2 + c_1\, c_2\, (\, g_1 - g_5\,) \\
G_{(3)} &= c_1^2\, g_5 + 2\, c_1\, c_2\, g_2 + c_2^2\, g_1 \\
G_{(4)} &= c_1\, g_3 - c_2\, g_6 \\
G_{(5)} &= c_2\, g_3 + c_1\, g_6 \\
G_{(6)} &= g_8 \\
G_{(7)} &= c_1\, g_4 - c_2\, g_7 \\
G_{(8)} &= c_1\, g_7 + c_2\, g_4 \\
G_{(9)} &= g_9 \\
G_{(10)} &= g_{10}
\end{aligned}
\qquad (3.24)
$$

where G_1, G_2, and H are defined in equations 3.14, 3.15, and 3.16 respectively. The values of g_1 through g_{10} are the components of the local tangent stiffness matrix, k_0, that operates on six local element degrees of freedom (see figure 2.17a). This matrix and its components are given in equation 3.25.

$$k_0 = \begin{bmatrix} g_1 & g_2 & g_3 & -g_1 & -g_2 & g_4 \\ & g_5 & g_6 & -g_2 & -g_5 & g_7 \\ & & g_8 & -g_3 & -g_6 & g_9 \\ & & & g_1 & g_2 & -g_4 \\ & & & & g_5 & -g_7 \\ \text{symm.} & & & & & g_{10} \end{bmatrix}$$

$$g_1 = (E\ I\ \pi^2)/(H\ L^3) \qquad (3.25)$$

$$g_2 = E\ I\ (G_1 + G_2)\ /\ (H\ L^3)$$

$$g_3 = E\ I\ G_1\ /\ (H\ L^2)$$

$$g_4 = E\ I\ G_2\ /\ (H\ L^2)$$

$$g_5 = \frac{E\ I}{L}\left[\frac{2\ (\ s_1 + s_2\ )}{L^2} + \frac{G_1^2 + 2G_1G_2 + G_2^2}{\pi^2\ H\ L^2}\right]$$

$$g_6 = \frac{E\ I}{L}\left[\frac{s_1 + s_2}{L} + \frac{G_1^2 + G_1\ G_2}{\pi^2\ H\ L}\right]$$

$$g_7 = \frac{E\ I}{L}\left[\frac{s_1 + s_2}{L} + \frac{G_1\ G_2 + G_2^2}{\pi^2\ H\ L}\right]$$

$$g_8 = E\ I\ (\ s_1 + G_1^2/\pi^2 H\ )\ /\ L$$

$$g_9 = E\ I\ (\ s_2 + G_1 G_2/\pi^2 H\ )\ /\ L$$

$$g_{10} = E\ I\ (\ s_1 + G_2^2/\pi^2 H\ )\ /\ L$$

The contributions to $G_{(1)}$ through $G_{(10)}$ in K_T of equation 3.23 for K_σ are as follows.

$$G_{(1)} = -2\, c_1\, c_2 \left[\frac{M_1 + M_2}{L^2}\right] - c_2^2 \frac{Q}{L}$$

$$G_{(2)} = (c_1^2 - c_2^2) \left[\frac{M_1 + M_2}{L^2}\right] + c_1\, c_2 \frac{Q}{L} \qquad (3.26)$$

$$G_{(3)} = 2\, c_1\, c_2 \left[\frac{M_1 + M_2}{L^2}\right] - c_1^2 \frac{Q}{L}$$

and $G_{(4)}$ through $G_{(10)}$ are zero.

3.2 Development of Finite Element Model

The techniques used in this section may be found in references (11, 18, and 39).

The first step in the development is to establish the condition of equilibrium by the principle of virtual work.

$$\delta W = \delta W_e - \delta U = 0 \qquad (3.27)$$

where the variation of the work done by the external forces is

$$\delta W_e = \delta d^T f \qquad (3.28)$$

and the variation of the internal strain energy is obtained from figure 3.2 :

$$\delta U = \int \delta \epsilon^T \sigma \, dV \qquad (3.29)$$

where the integration is performed over the volume of

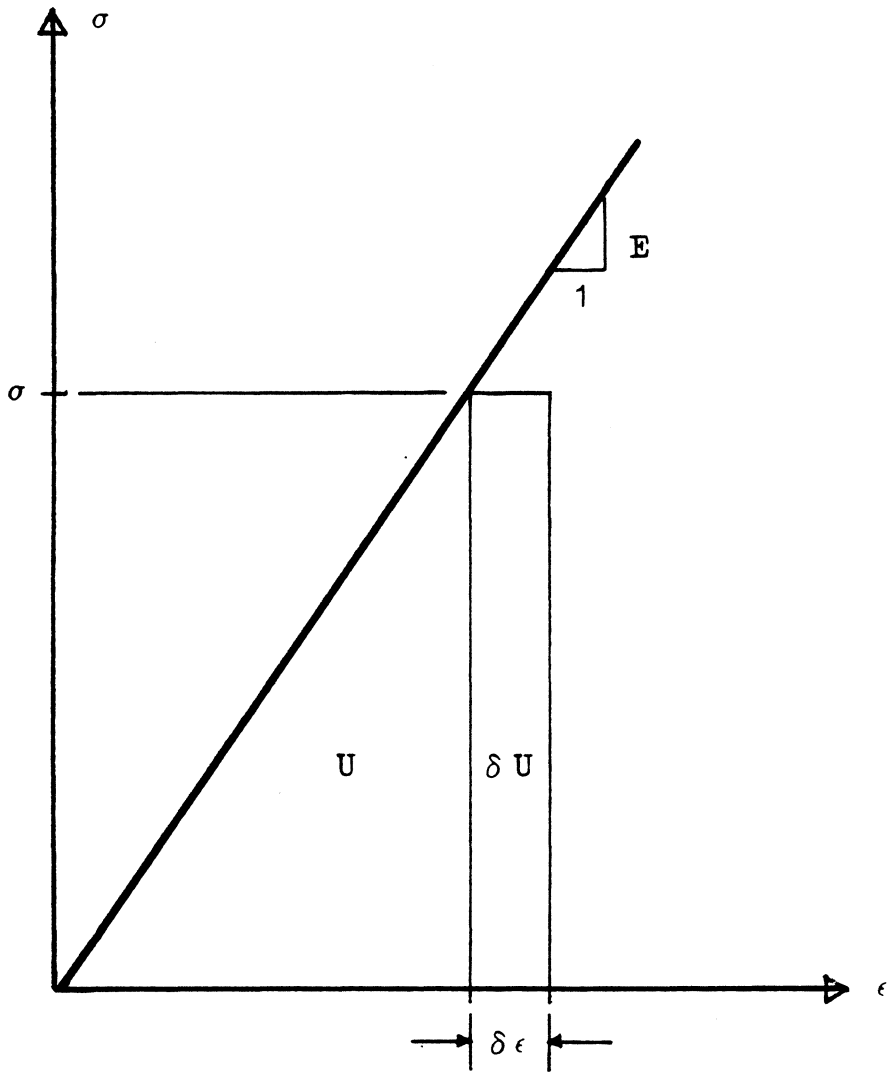


Figure 3.2 : Strain Energy

the element. Combining equations 3.27, 3.28, and 3.29,

$$\delta d^T f - \int \delta \epsilon^T \sigma \, dV = 0 \tag{3.30}$$

for equilibrium.

Green's strain contains terms which are either linearly or quadratically dependent on displacements. Thus, the strain may be expressed as

$$\epsilon = \epsilon_0 + \epsilon_L \tag{3.31}$$

where ϵ_0 is linearly dependent and ϵ_L is quadratically dependent on displacements.

For the B-notation (41)

$$\epsilon = \epsilon_0 + \tfrac{1}{2} A \theta \tag{3.32}$$

where

$$\theta = \begin{bmatrix} \frac{du}{dx} \\ \frac{dv}{dx} \\ \frac{d^2v}{dx^2} \end{bmatrix} \tag{3.33}$$

and

$$A^T = \begin{bmatrix} 0 \\ \frac{dv}{dx} \\ 0 \end{bmatrix} \tag{3.34}$$

For the N-notation (21, 29)

$$\epsilon = L^T \theta + \tfrac{1}{2} \theta^T H \theta \tag{3.35}$$

where

$$L = \begin{bmatrix} 1 \\ 0 \\ -y \end{bmatrix} \tag{3.36}$$

and

$$H = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{3.37}$$

Thus, Green's strain is given.

$$\epsilon = \frac{du}{dx} - y \frac{d^2v}{dx^2} + \frac{1}{2} \left(\frac{dv}{dx} \right)^2 \tag{3.38}$$

θ is discretized :

$$\theta = G\ d \tag{3.39}$$

where G is a matrix containing derivatives of the interpolation functions (see appendix K for a more detailed presentation). Substituting equation 3.39 into the equations for strain, and letting

$$B_O = L^T G \tag{3.40}$$

and

$$B_L = A G \tag{3.41}$$

for the B-notation, and

$$B_L = d^T G^T H G \tag{3.42}$$

for the N-notation, the strain is

$$\epsilon = (B_0 + \tfrac{1}{2} B_L) d \tag{3.43}$$

and, incidentally,

$$\delta \epsilon = (B_0 + B_L) \delta d \tag{3.44}$$

Applying the constitutive law,

$$\sigma = E \epsilon \quad , \tag{3.45}$$

$$\sigma = \sigma_0 + \sigma_L \tag{3.46}$$

where

$$\sigma_0 = E \epsilon_0 \tag{3.47}$$

$$\sigma_L = E \epsilon_L \tag{3.48}$$

where σ is the second Piola-Kirchhoff stress (1, 39).

A symmetric stress matrix is introduced such that

$$S = S_0 + S_L \tag{3.49}$$

where

$$S_0 = \sigma_0 H \tag{3.50}$$

$$S_L = \sigma_L H \tag{3.51}$$

Combining equations 3.30 and 3.44 and noting that δd^T is an arbitrary variation,

$$f = \int B^T \sigma \, dV \tag{3.52}$$

Replacing B by $B_0 + B_L$ and σ by $E(B_0 + \frac{1}{2}B_L)$ d gives

$$f = \int (B_0^T + B_L^T) E (B_0 d + \tfrac{1}{2} B_L d) \, dV \tag{3.53}$$

$$f = \left[\int (B_0^T E B_0 + B_L^T E B_0 + \tfrac{1}{2} B_0^T E B_L + \tfrac{1}{2} B_L^T E B_L) dV \right] d \tag{3.54}$$

Equation 3.54 contains the secant stiffness for the B-notation in brackets. The tangent stiffness is formed by expanding equation 3.52 in a Taylor series.

$$\Delta f = k_T \Delta d \tag{3.55}$$

where

$$k_T = \int B^T D B \, dV + \int G^T S G \, dV \tag{3.56}$$

or

$$k_T = k_0 + k_\sigma + k_L \tag{3.57}$$

where

$$k_0 = \int B_0^T E B_0 \, dV \tag{3.58}$$

$$k_\sigma = \int (G^T S_0 G + G^T S_L G) \, dV \tag{3.59}$$

$$k_L = \int (B_0^T E B_L + B_L^T E B_0 + B_L^T E B_L) \, dV \tag{3.60}$$

In the N-notation,

$$k_S = k_0 + \tfrac{1}{2} N_1 + \tfrac{1}{3} N_2 \tag{3.61}$$

$$k_T = k_0 + N_1 + N_2 \tag{3.62}$$

where k_0 is the same as k_0 for the B-notation,

$$N_1 = \int (B_0^T E B_L + B_L^T E B_0 + G^T S_0 G) \, dV \quad , \tag{3.63}$$

and

$$N_2 = \int (B_L^T E B_L + G^T S_L G) \, dV \tag{3.64}$$

Notice that

$$k_\sigma + k_L = N_1 + N_2 \tag{3.65}$$

Equations 3.58 and 3.59 give a local stiffness matrix (neglecting k_L ; see reference 5).

$$k = \begin{bmatrix} T_1 & 0 & 0 & -T_1 & 0 & 0 \\ & T_2 & T_3 & 0 & -T_2 & T_3 \\ & & T_4 & 0 & -T_3 & T_5 \\ & & & T_1 & 0 & 0 \\ & & & & T_2 & -T_3 \\ \text{symm.} & & & & & T_4 \end{bmatrix} \tag{3.66}$$

where the contributions for k_0 are the elements of the conventional stiffness matrix for linear analysis (7).

$$\begin{aligned} T_1 &= E\ A\ /\ L \\ T_2 &= 12\ E\ I\ /\ L^3 \\ T_3 &= 6\ E\ I\ /\ L^2 \\ T_4 &= 4\ E\ I\ /\ L \\ T_5 &= 2\ E\ I\ /\ L \end{aligned} \tag{3.67}$$

The contributions for k_σ , which are developed fully in appendix K, are :

$$\begin{aligned} T_1 &= 0 \\ T_2 &= 6\ Q\ /\ 5\ L \\ T_3 &= Q\ /\ 10 \\ T_4 &= 2\ Q\ L\ /\ 15 \\ T_5 &= -\ Q\ L\ /\ 30 \end{aligned} \tag{3.68}$$

Where Q is the axial force in the member which is positive in tension.

The standard transformation (7)

$$K = \overline{\lambda}^T k \overline{\lambda} \tag{3.69}$$

where

$$\overline{\lambda} = \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} \tag{3.70}$$

and

$$\lambda = \begin{bmatrix} c_1 & c_2 & 0 \\ -c_2 & c_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.71}$$

gives

$$K = \begin{bmatrix} G_{(1)} & G_{(2)} & G_{(4)} & -G_{(1)} & -G_{(2)} & G_{(7)} \\ & G_{(3)} & G_{(5)} & -G_{(2)} & -G_{(3)} & G_{(8)} \\ & & G_{(6)} & -G_{(4)} & -G_{(5)} & G_{(9)} \\ & & & G_{(1)} & G_{(2)} & -G_{(7)} \\ & & & & G_{(3)} & -G_{(8)} \\ \text{symm.} & & & & & G_{(10)} \end{bmatrix} \tag{3.72}$$

(Note the similarity to equation 3.23.)

The components of equation 3.72 are :

$$G_{(1)} = c_1^2 T_1 + c_2^2 T_2$$

$$G_{(2)} = c_1 c_2 (T_1 - T_2)$$

$$G_{(3)} = c_2^2 T_1 + c_1^2 T_2$$

$$G_{(4)} = - c_2 T_3$$

$$G_{(5)} = c_1 T_3 \qquad (3.73)$$

$$G_{(6)} = T_4$$

$$G_{(7)} = - c_2 T_3$$

$$G_{(8)} = c_1 T_3$$

$$G_{(9)} = T_5$$

$$G_{(10)} = T_4$$

CHAPTER 4

PROGRAM DEVELOPMENT

4.1 Introduction

The listing of the computer program developed for this study is given in appendix N. This listing contains definitions of variables, descriptions of subroutines, and a description of the input data. An additional explanation of the SIGN vector is presented in appendix L. The tree chart for the program, subroutine descriptions, input and output lists for the subroutines, and the Nassi-Schneiderman diagrams (7) for the subroutines are provided in this chapter.

The structured WATFIV (WATFIV-S) version of Fortran is chosen as the coding language for this study. This particular version of Fortran seems to be very easy to read and to understand. The program could easily be converted to a coding language that is cheaper to run, if cost of running became more important than readability.

Note that data which is dependent upon the structure being analyzed and the loads for which the analysis is performed are read as input data. Parameters which control the type of analysis performed are defined in the MAIN section of the program.

Subroutines FACTOR, REDUCE, and BACSUB, which are

used as an equation solving routine, are adapted from Bathe (1) and are not covered in detail in this chapter.

4.2 Program Structure

The tree chart for the program is shown in figure 4.1.

The MAIN section of the program initializes parameters that control the type of analysis that is performed, the type of output that is generated, the convergence to an equilibrium point, the convergence to a value of QR, and the maximum values for some variables. QI, QIMAX, DQI, IMP, and SIGN are read. The loads and the configuration are initialized to the undeformed state. Subroutine DATA is called; NEWRAP is called if a Newton-Raphson analysis is desired, or RIKWEM is called for the modified Riks/Wempner method. Figure 4.2 shows the Nassi-Schneiderman diagram for MAIN.

Subroutine DATA reads the numbers of elements and joints, tests these against the maximum number allowed, and calls STRUCT and LOAD. See figure 4.3.

Input: MX, MXNA, MXNEQ.

Output: A1, A2, AREA, C1, C2, CI1, CI2, DIST, DIST2, ELENG, EMOD, JCODE, MAXA, MCODE, MINC, MN, NAT, NEQ, NJ, NKT, Q, X, ZI.

STRUCT reads the member incident matrix and the

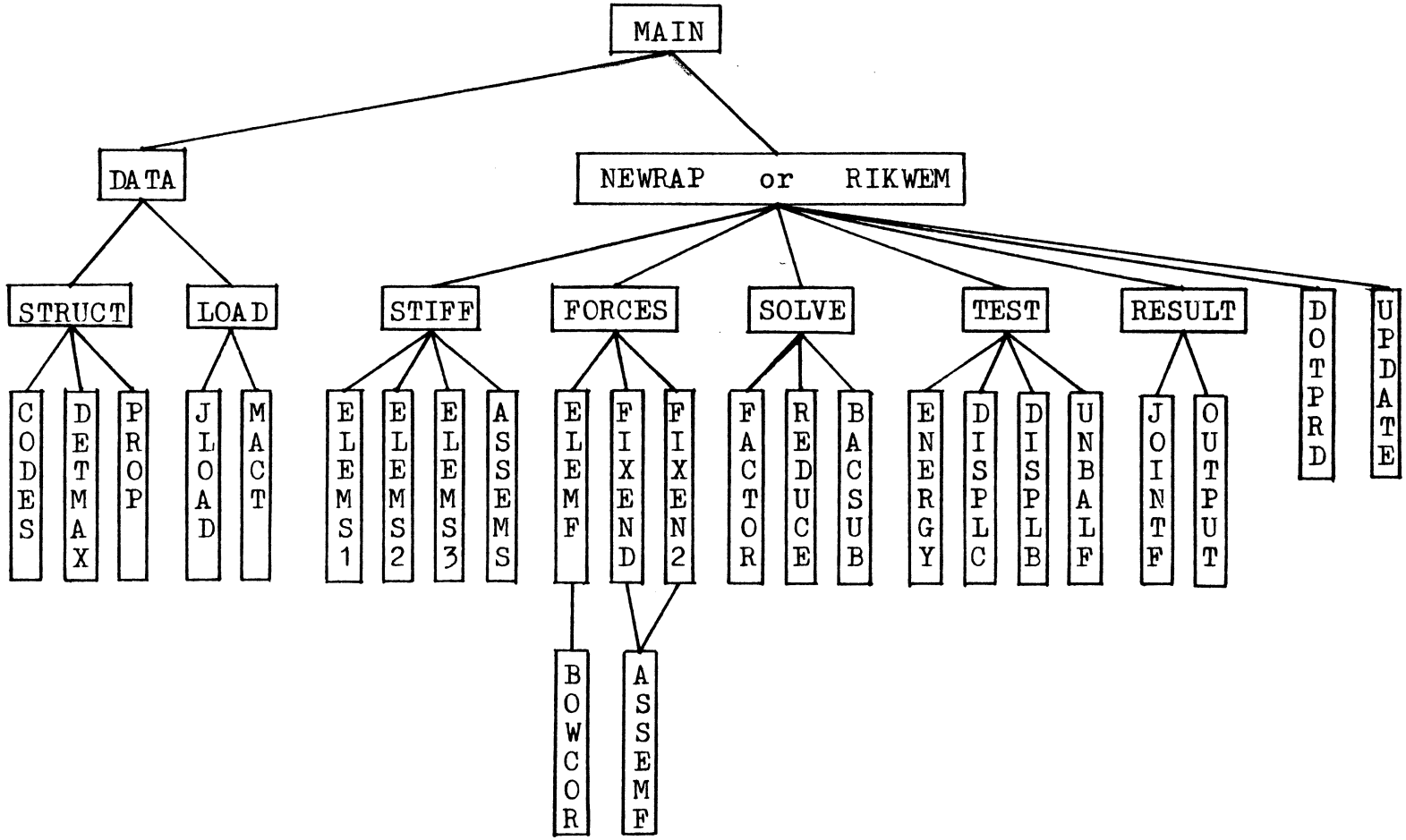


Figure 4.1 : Program Tree Chart

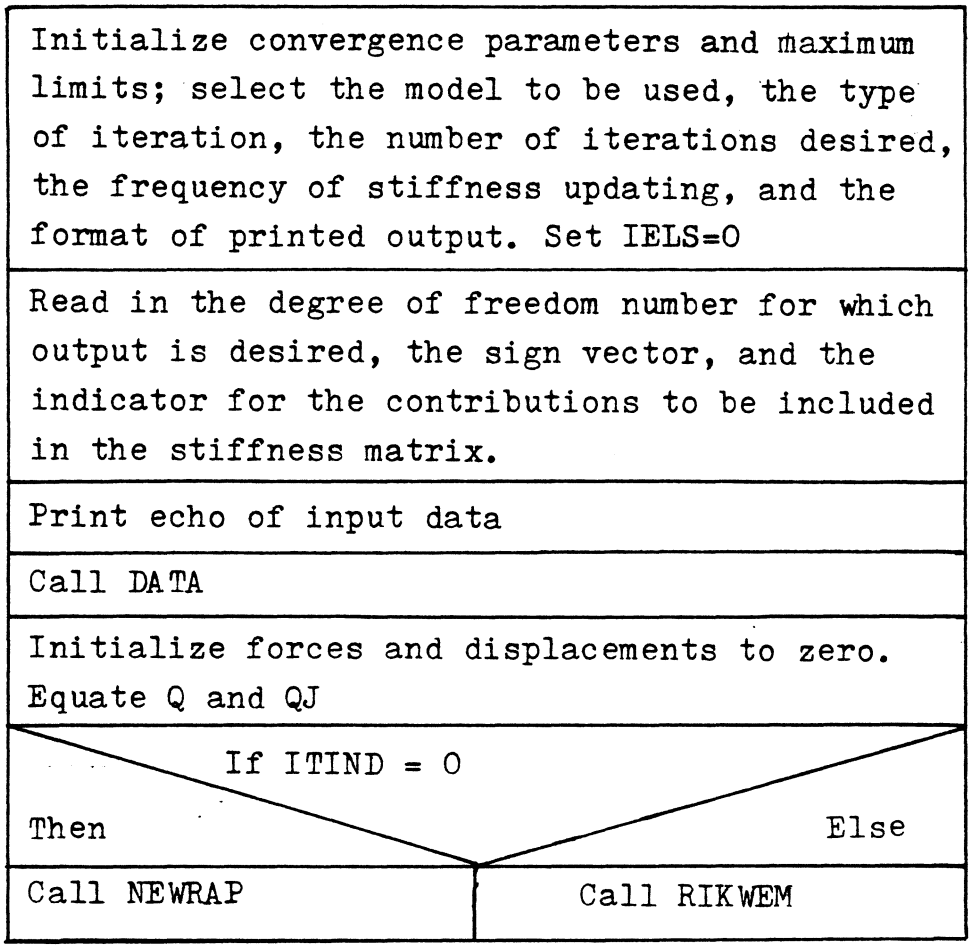


Figure 4.2 : MAIN

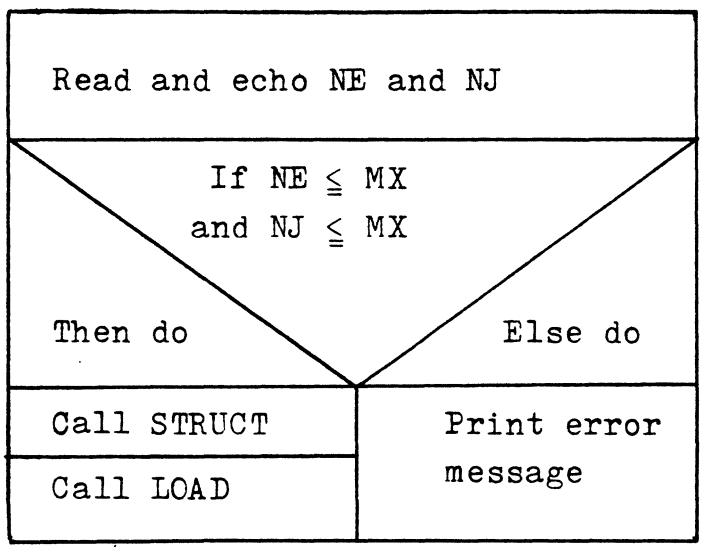


Figure 4.3 : DATA

joint constraints. The joint code matrix is initialized to unity, and then components corresponding to constraints are set equal to zero. CODES, DETMAX, and PROP are called. See figure 4.4.

Input: MXNEQ, NE, NJ.

Output: AREA, C1, C2, CI1, CI2, ELENG, EMOD, JCODE, MAXA, MCODE, MINC, NEQ, NKT, X, ZI.

CODES assigns degree of freedom numbers to JCODE and generates MCODE using JCODE and MINC. See figure 4.5.

Input: JCODE, MXNEQ, NE, NJ, MINC.

Output: JCODE, MCODE, NEQ.

DETMAX determines KHT using MCODE and determines MAXA from KHT. See figure 4.6.

Input: MCODE, NE, NEQ.

Output: MAXA, NKT.

PROP reads joint coordinates and element properties. Element lengths and direction cosines are computed. See figure 4.7.

Input: NE,NJ.

Output: AREA, C1, C2, CI1, CI2, ELENG, EMOD, MINC, X, ZI.

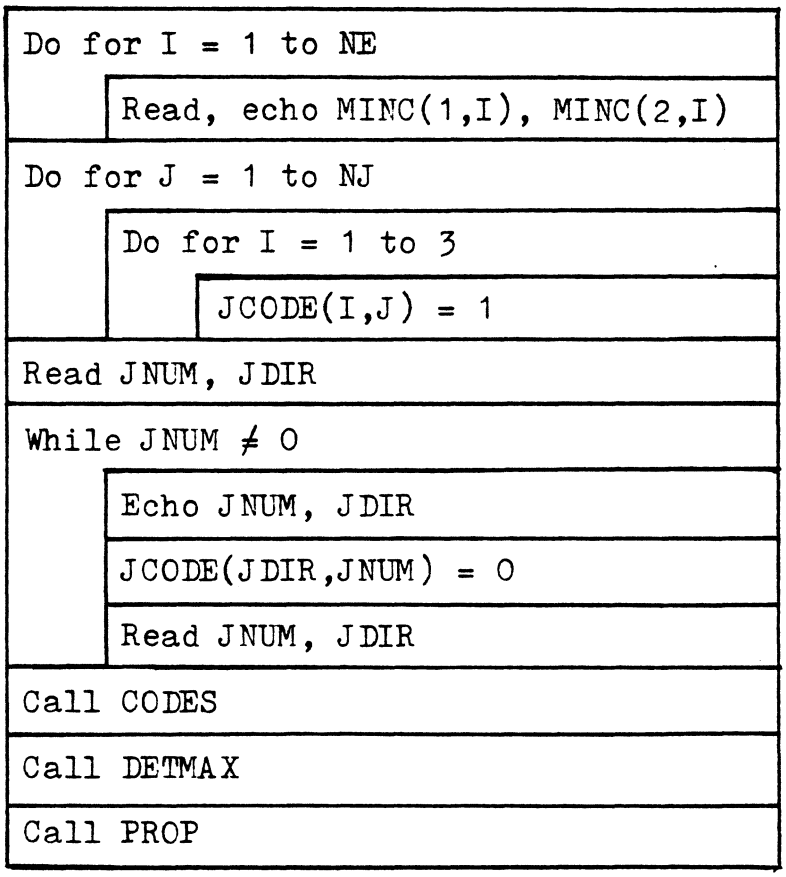


Figure 4.4 : STRUCT

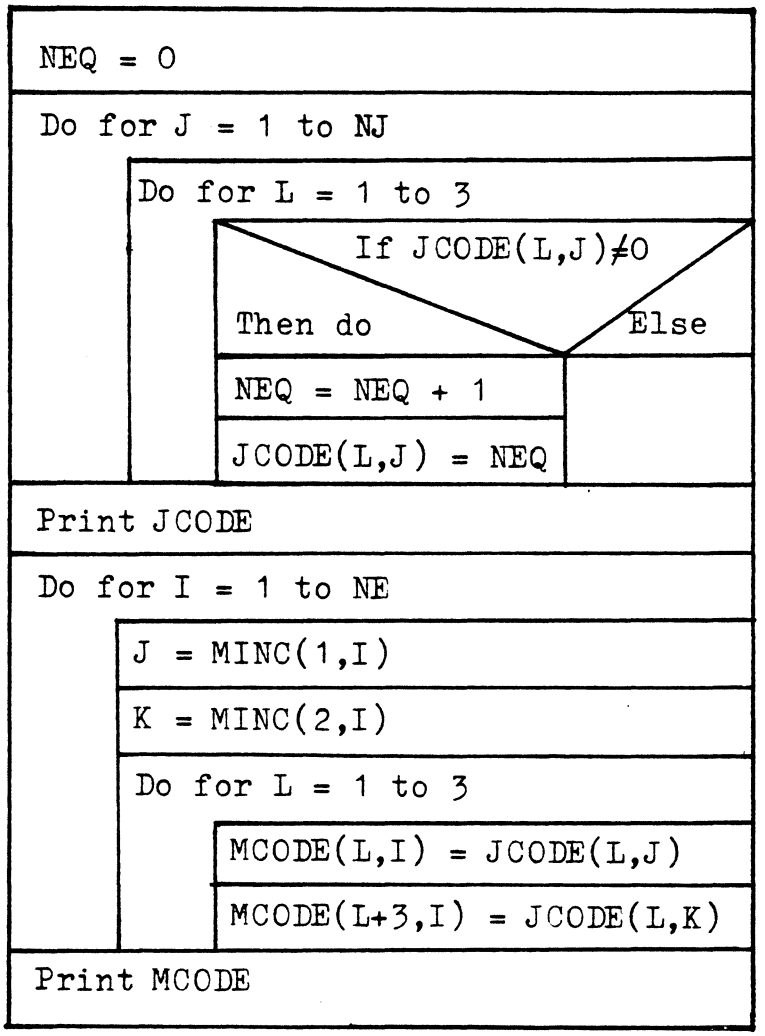


Figure 4.5 : CODES

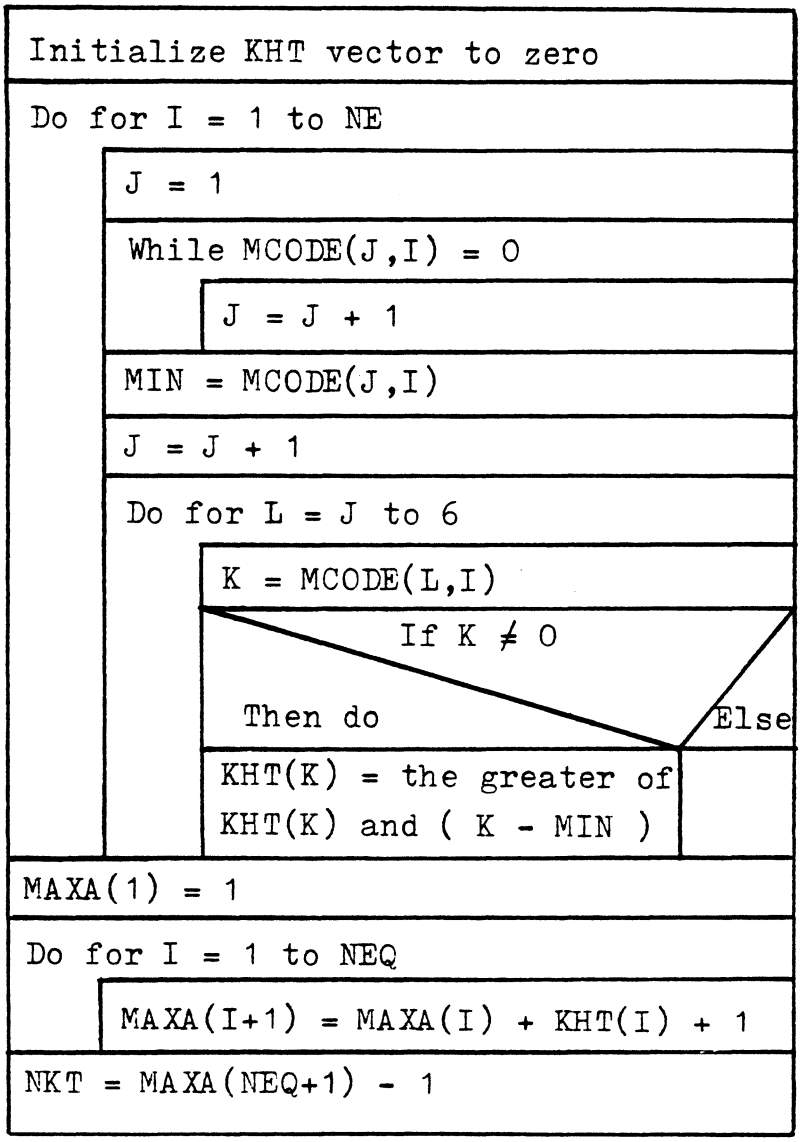


Figure 4.6 : DETMAX

Read and echo joint coordinates
Do for I = 1 to NE
J = MINC(1,I)
K = MINC(2,I)
EL1 = X(1,K) - X(1,J)
EL2 = X(2,k) - X(2,J)
ELENG(I) = $(EL1^2 + EL2^2)^{\frac{1}{2}}$
C1(I) = CI1(I) = EL1/ELENG(I)
C2(I) = CI2(I) = EL2/ELENG(I)
Read and echo AREA(I), EMOD(I), ZI(I)

Figure 4.7 : PROP

LOAD initializes Q to zero and calls JLOAD and MACT. See figure 4.8.

Input: JCODE, MXNA, NEQ.

Output: A1, A2, DIST, DIST2, MAT, MN, NAT, Q.

JLOAD reads joint load information and assigns joint load values to Q. See figure 4.9.

Input: JCODE.

Output: Q.

MACT reads and stores the element action data. See figure 4.10.

Input: MXNA.

Output: A1, A2, DIST, DIST2, MAT, MN, NAT.

NEWRAP performs the Newton-Raphson or modified Newton-Raphson method. STIFF, SOLVE, FORCES, TEST, RESULT, and UPDATE are called. See figure 4.11.

Input: AREA, C1, C2, CI1, CI2, ELENG, EMOD, JCODE, MAXA, MCODE, MINC, ITIND, INDTAN, KT, IMP, Q, QJ, CPDB, CPDC, CPF, CPE, DP, DQQR, FQR, IMAX, ITMAX, NE, NJ, NEQ, NKT, NUPD, A1, A2, DIST, DIST2, MAT, MN, NAT, QI, QIMAX, DQI, NPRINT, IELS.

Output: none.

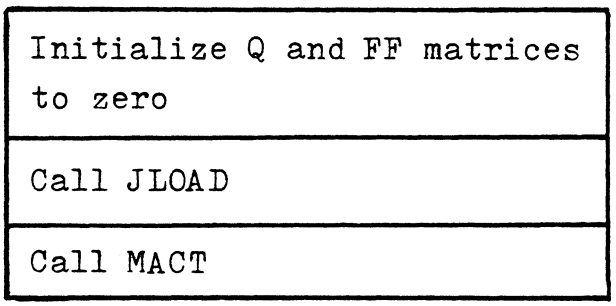


Figure 4.8 : LOAD

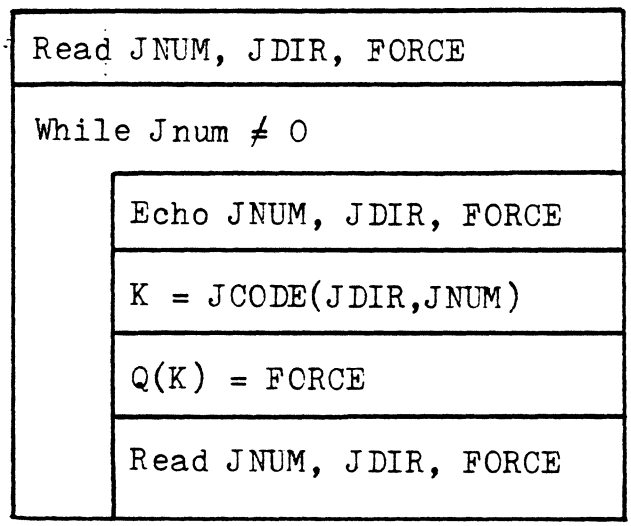


Figure 4.9 : JLOAD

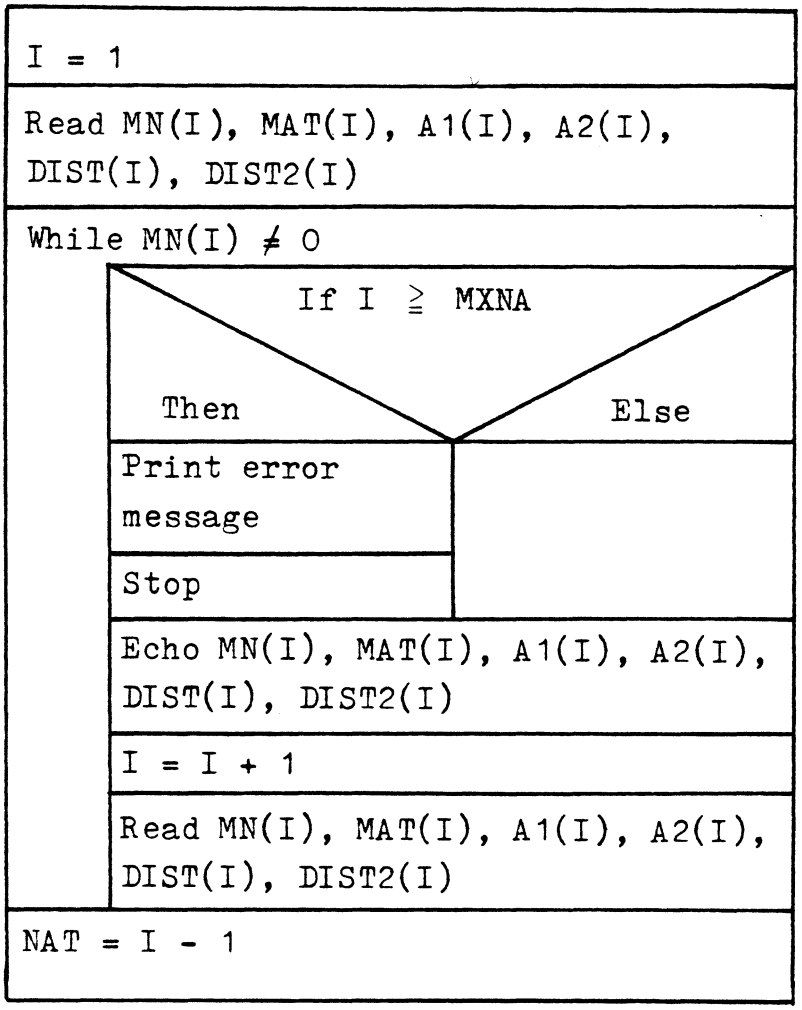


Figure 4.10 : MACT

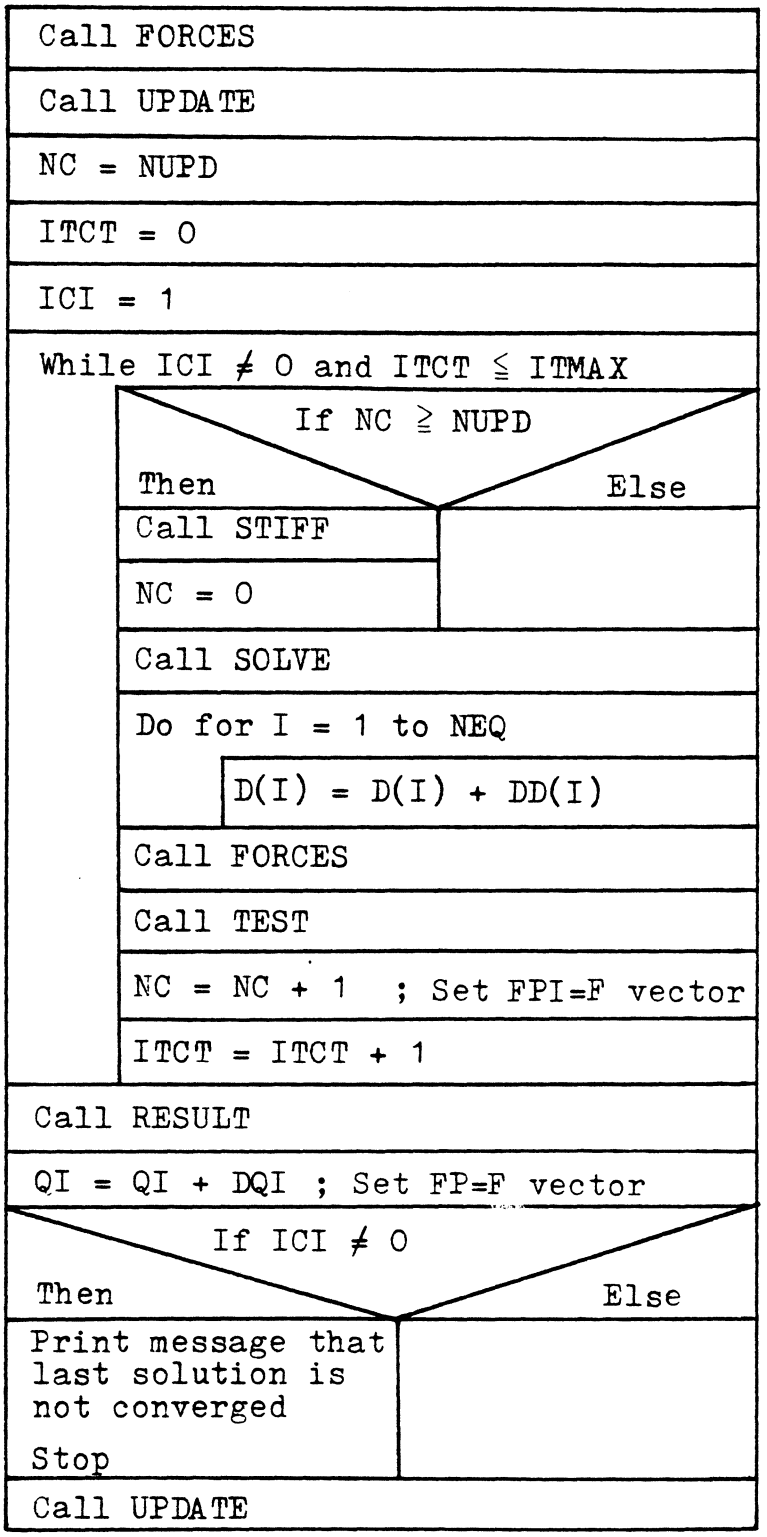


Figure 4.11 : NEWRAP

RIKWEM performs the modified Riks/Wempner method for iteration on the normal. STIFF, SOLVE, FORCES, TEST, RESULT, DOTPRD, and UPDATE are called. See figure 4.12.

Input: AREA, C1, C2, CI1, CI2, ELENG, EMOD, JCODE, MAXA, MCODE, MINC, ITIND, INDTAN, KT, IMP, Q, QJ, CPDB, CPDC, CPF, CPE, DP, DQQR, FQR, IMAX, ITMAX, NE, NJ, NEQ, NKT, NUPD, A1, A2, DIST, DIST2, MAT, MN, NAT, QI, QIMAX, DQI, NPRINT, SIGN, IELS.

Output: none.

STIFF initializes the system tangent stiffness matrix to zero and calls one of the ELEMS routines and ASSEMS for each element. See figure 4.13.

Input: AREA, BP1, BP2, C1, C2, ELENG, EMOD, MAXA, MCODE, R1, R2, SP1, SP2, ST1, ST2, RM12, QL, U, KT, ZI, IELS.

Output: SKT.

ELEMS1 computes the components of the system tangent stiffness matrix for the beam-column model for each element. See figure 4.14.

Input: AREA, BP1, BP2, C1, C2, ELENG, EMOD, R1, R2, SP1, SP2, ST1, ST2, RM12, QL, I, KT, ZI, IELS.

Output: G.

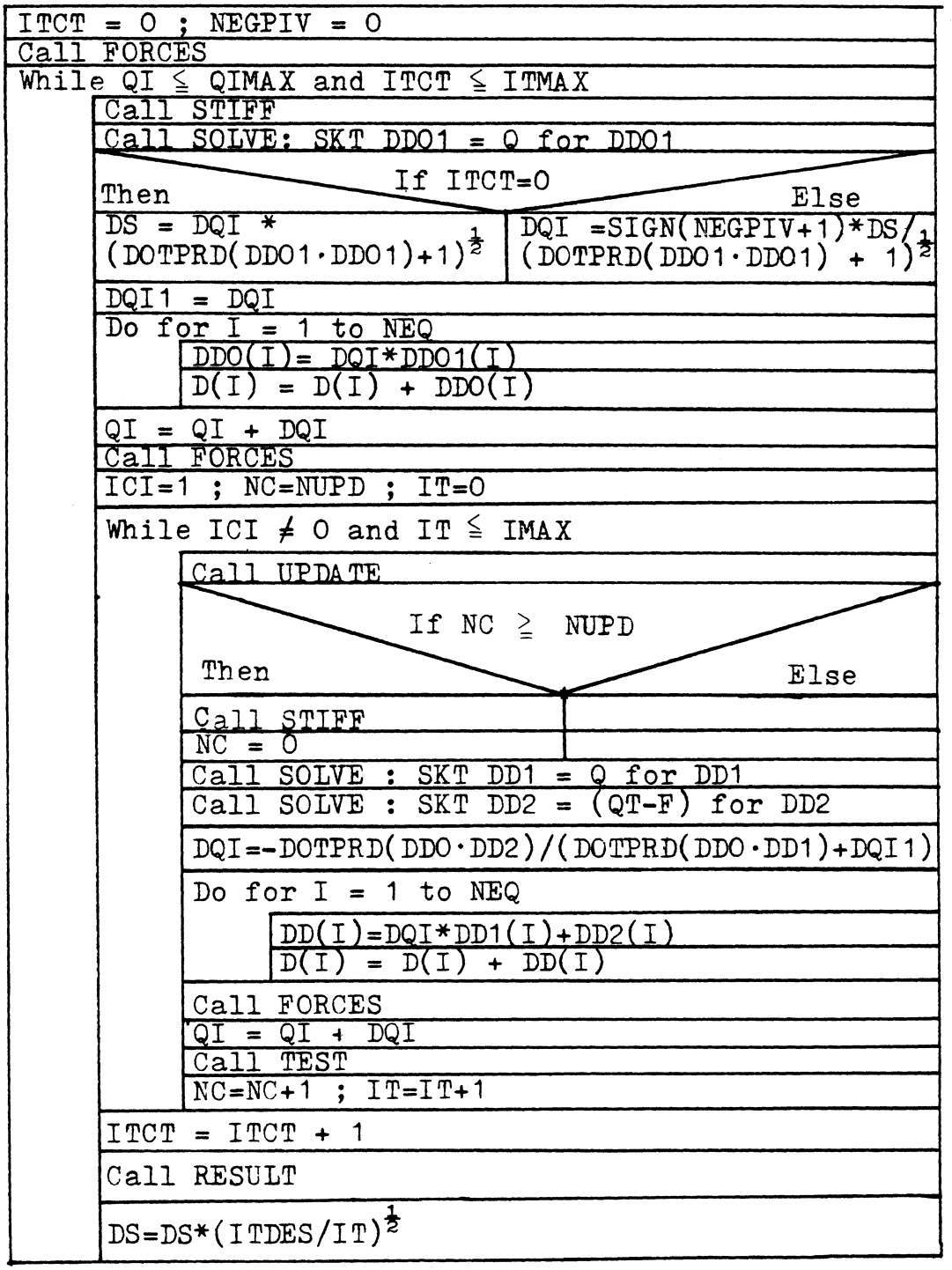


Figure 4.12 : RIKWEM

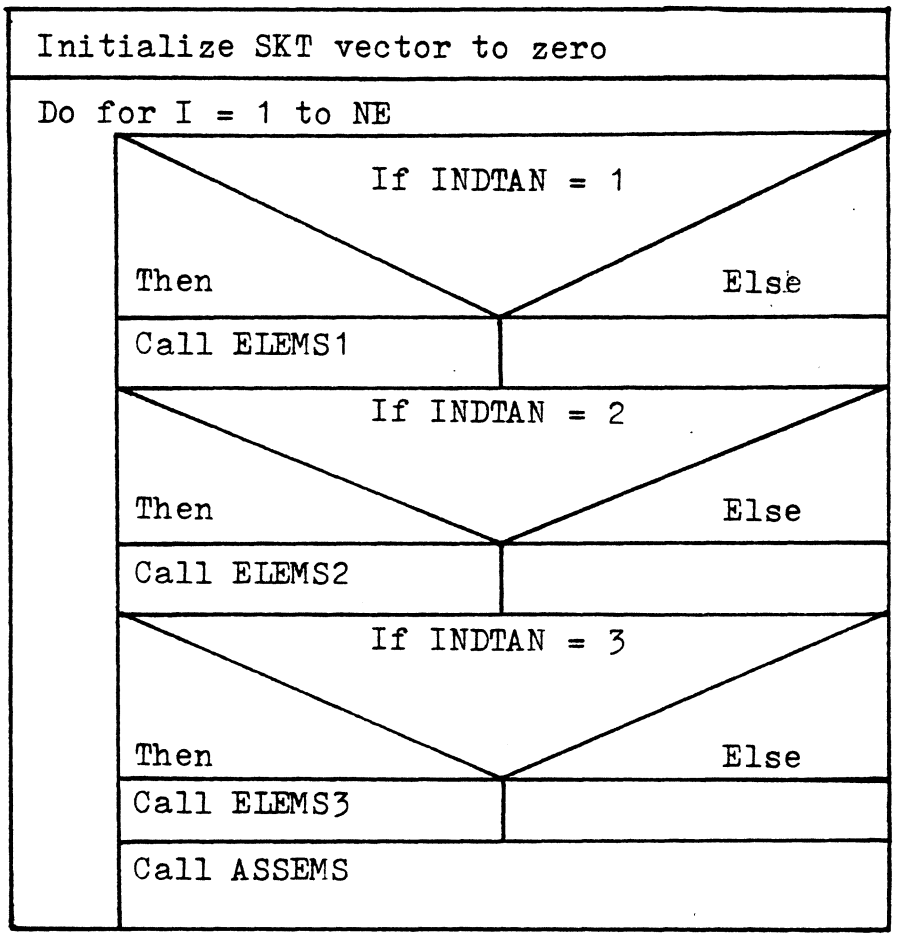


Figure 4.13 : STIFF

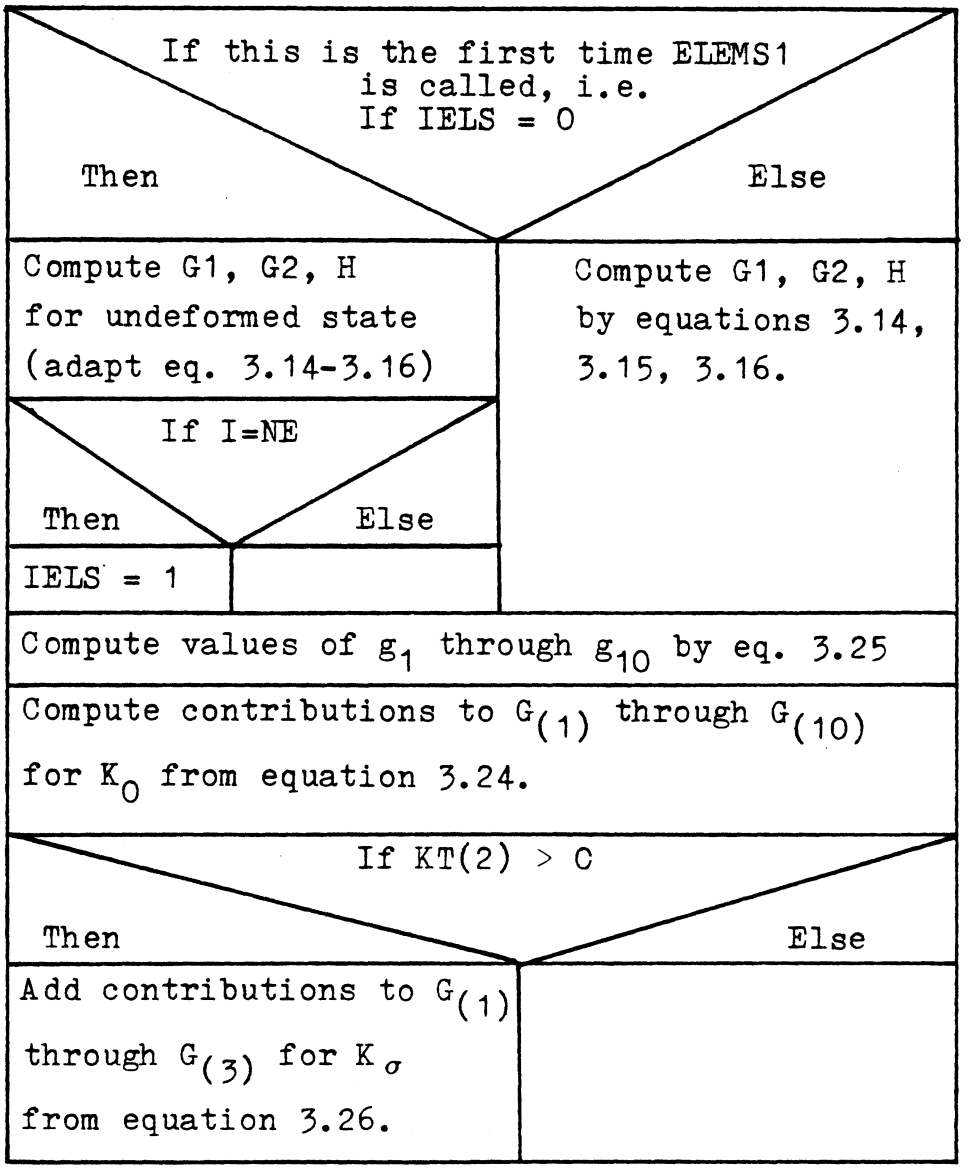


Figure 4.14 : ELEMS1

ELEMS2 computes the components of SKT for the finite element model derived using convected coordinates and neglecting the variation of the coordinate transformation matrix. This model does not always give a satisfactory representation of SKT. See figure 4.15.

Input: AREA, C1, C2, ELENG, EMOD, R1, R2, U, I, KT, ZI.

Output: G.

ELEMS3 computes the components of SKT for the finite element model developed using six local degrees of freedom. See figure 4.16.

Input: AREA, C1, C2, ELENG, EMOD, R1, R2, U, I, KT, U, QL.

Output: G.

ASSEMS assembles the contribution from each element to the system tangent stiffness matrix. See figure 4.17.

Input: G, I, MAXA, MCODE.

Output: SKT.

FORCES initializes the element force vector, F, to zero, calls ELEMF for each member, and calls FIXEND or FIXEN2 if there are member actions. See figure 4.18.

Input: AREA, C1, C2, CI1, CI2, D, ELENG, EMOD, MCODE, ZI, DQQR, FQR, IMAX, NE, NEQ, A1, A2, DIST, DIST2, MAT, MN, QJ, NAT, INDTAN.

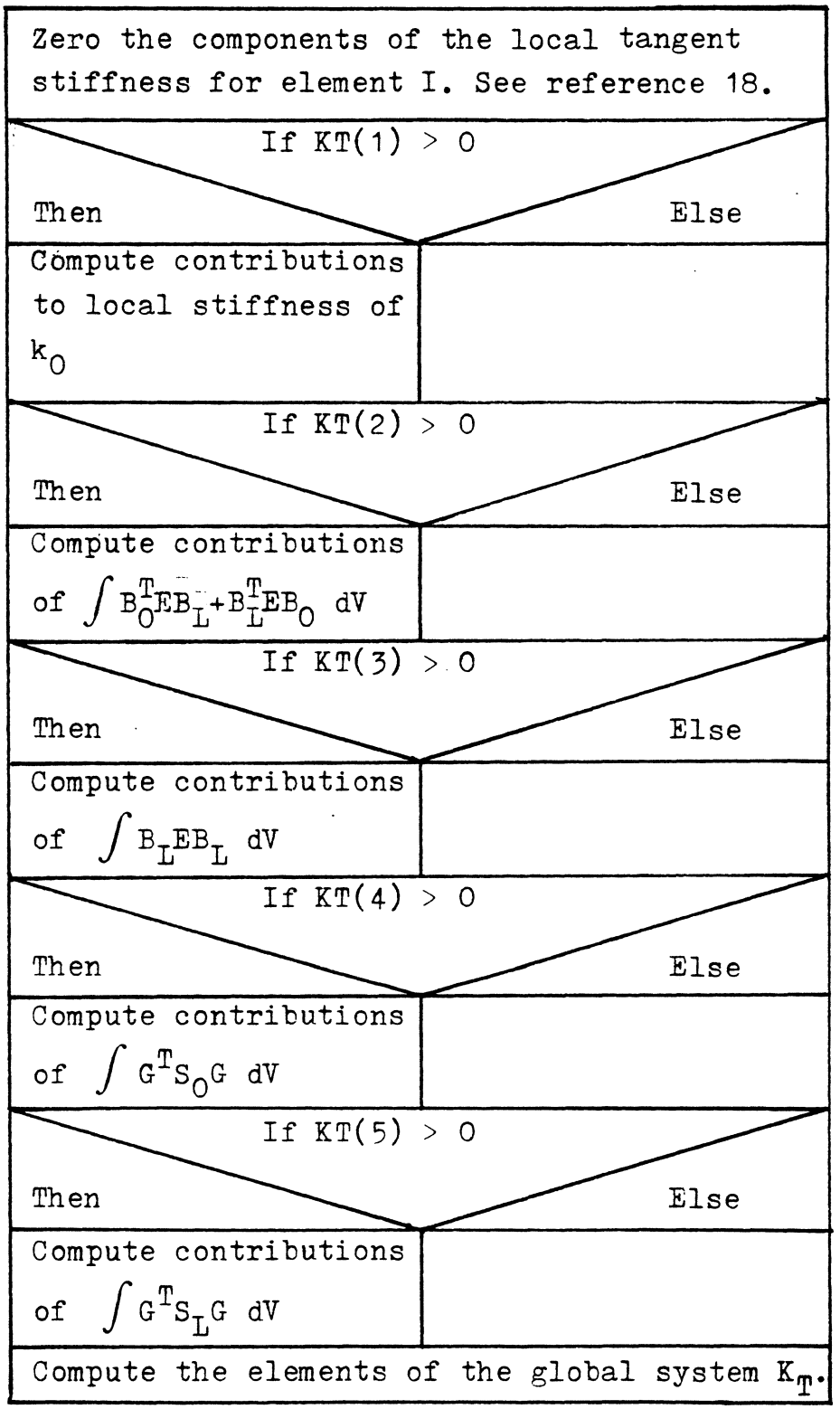


Figure 4.15 : ELEMS2

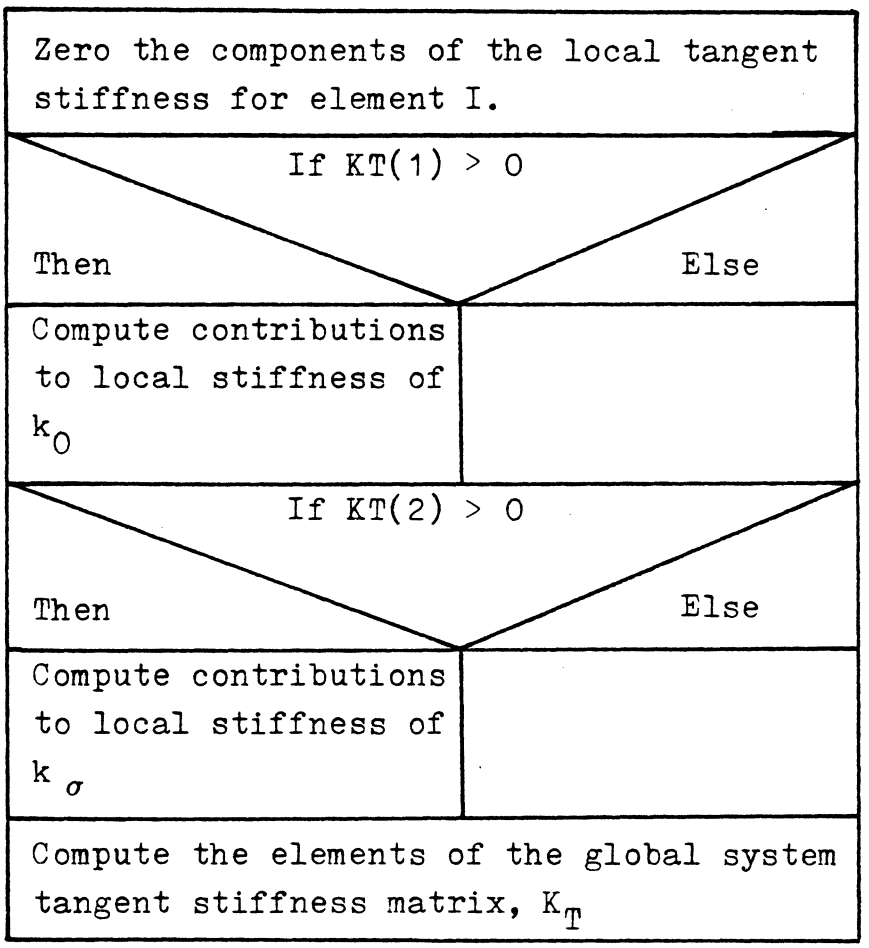


Figure 4.16 : ELEMS3

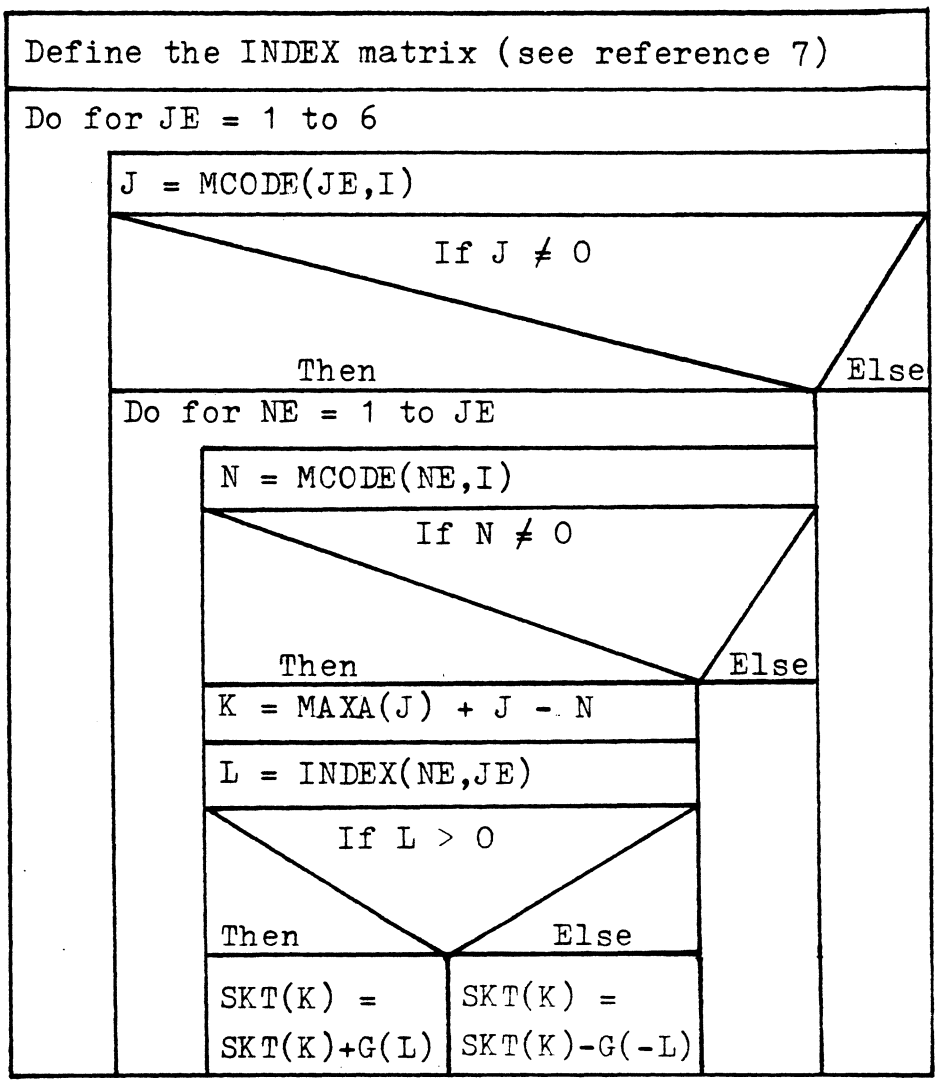


Figure 4.17 : ASSEMS

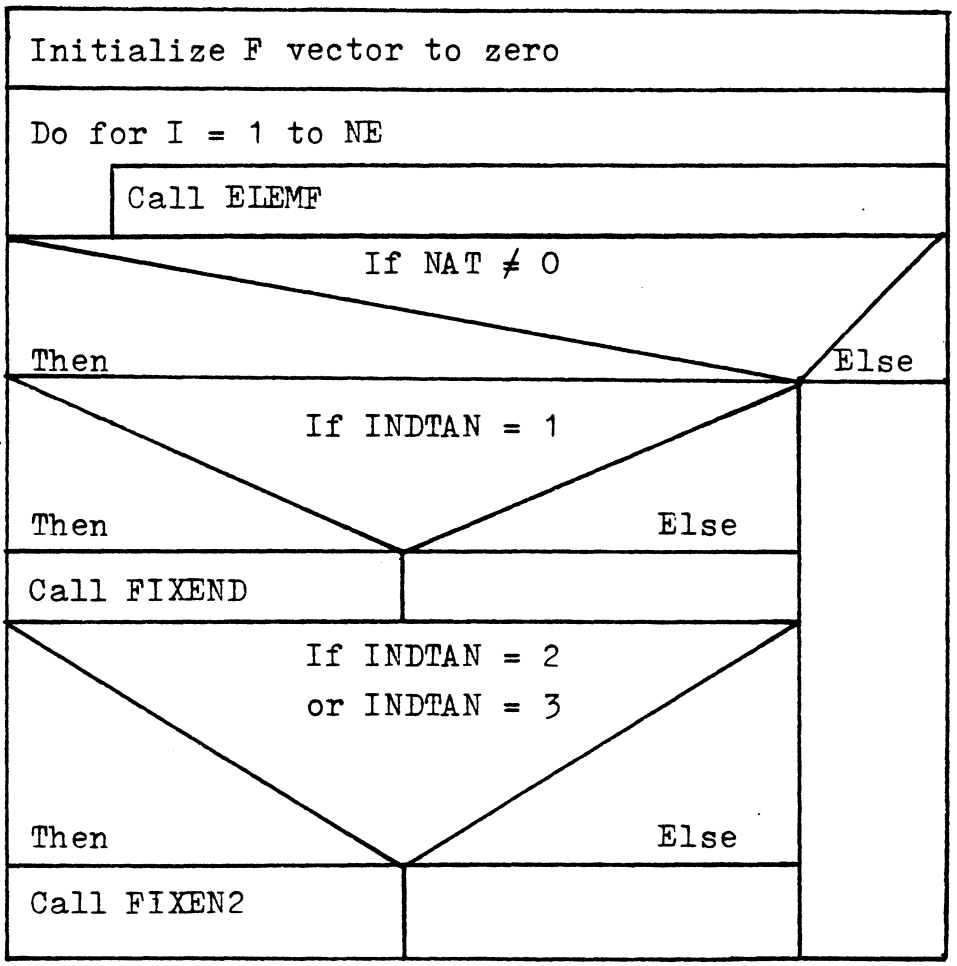


Figure 4.18 : FORCES

Output: BP1, BP2, C1, C2, F, FL, FG, Q, R1, R2, SP1, SP2, ST1, ST2, FF, RM12, QL, U.

ELEMF computes the internal element forces due to the deformations and transforms them to global forces. BOWCOR is called for the beam-column model. See figure 4.19.

Input: AREA, C1, C2, CI1, CI2, D, ELENG, EMOD, MCODE, ZI, DQQR, I, IMAX, INDTAN, FQR.

Output: BP1, BP2, C1, C2, F, FL, FG, R1, R2, SP1, SP2, ST1, ST2, RM12, QL, U.

BOWCOR uses a Newton method to find QR. See figure 4.20.

Input: AREA, ELENG, EMOD, D, DQQR, FQR, IMAX, ZI, I.

Output: BP1, BP2, R1, R2, SP1, SP2, ST1, ST2, CB, UL.

FIXEND computes the fixed end forces due to member actions for the beam-column model. See figure 4.21.

Input: A1, A2, C1, C2, DIST, ELENG, EMOD, MAT, MN, MCODE, NAT, NE, NEQ, QJ, ZI, FL.

Output: FF, Q.

FIXEN2 computes the fixed end forces due to member actions for the finite element model. See figure 4.22.

Input: A1, A2, C1, C2, DIST, DIST2, ELENG, EMOD, MAT, MCODE, MN, ZI, QJ.

Output: FF, Q.

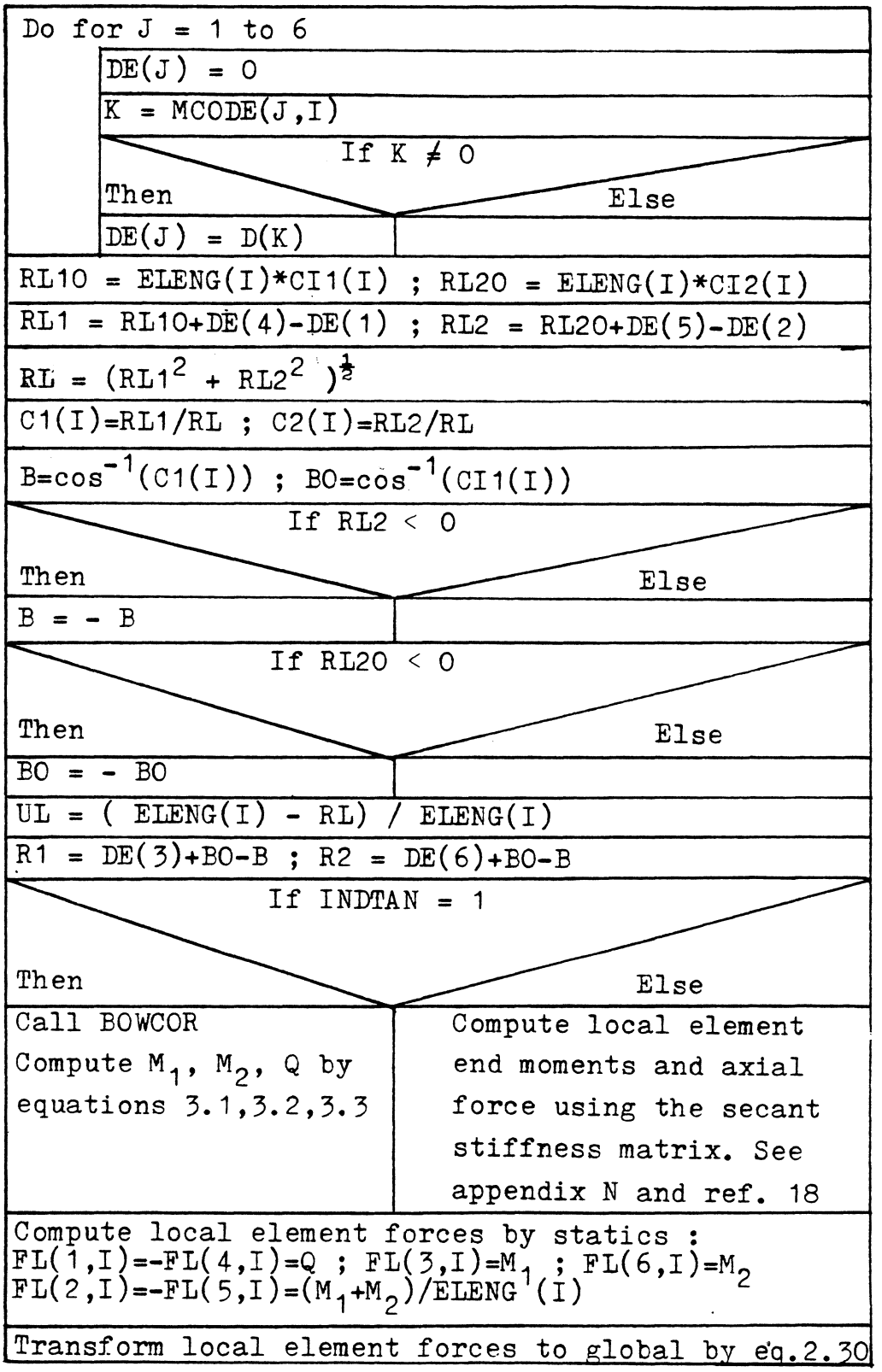


Figure 4.19 : ELEMF

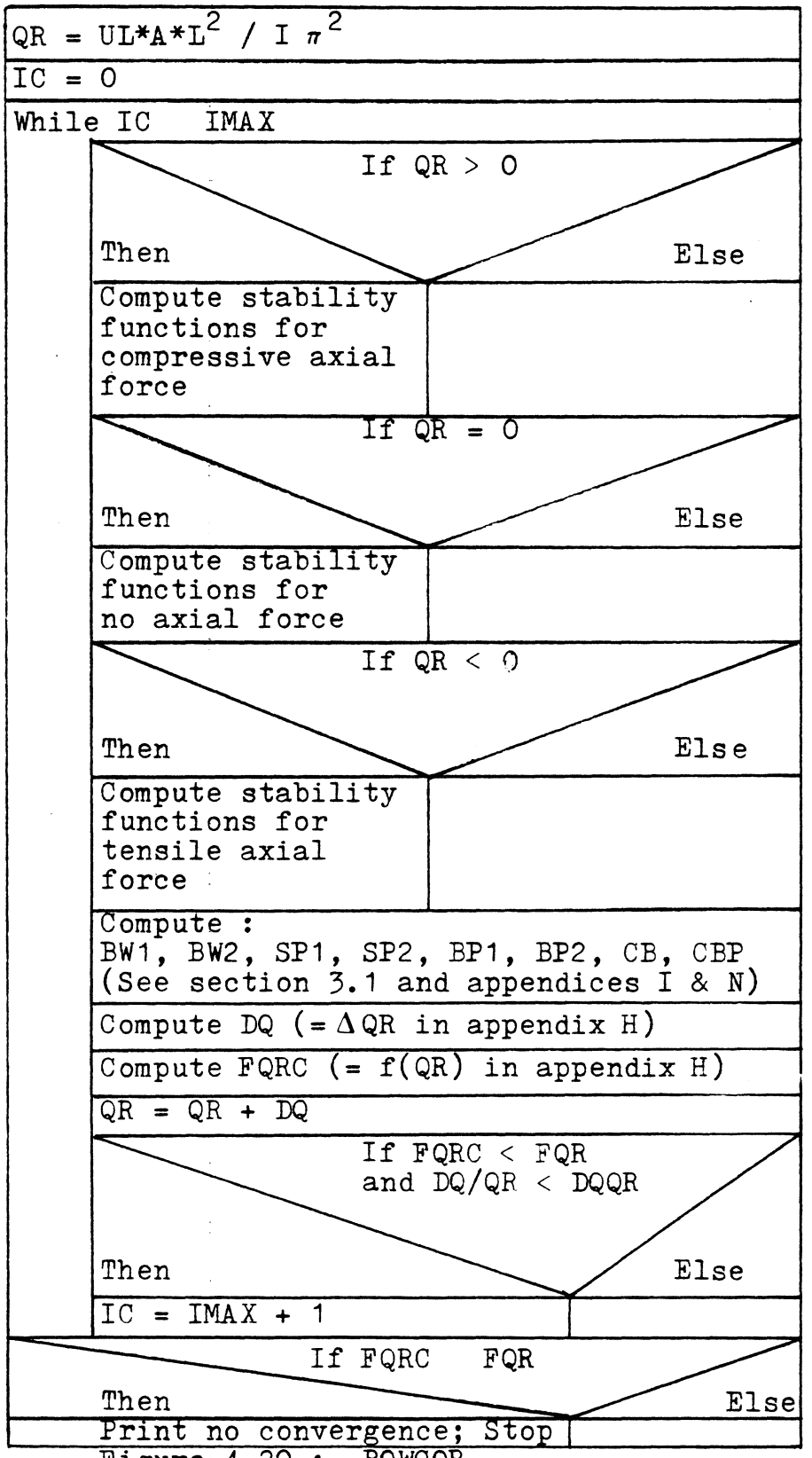


Figure 4.20 : BOWCOR

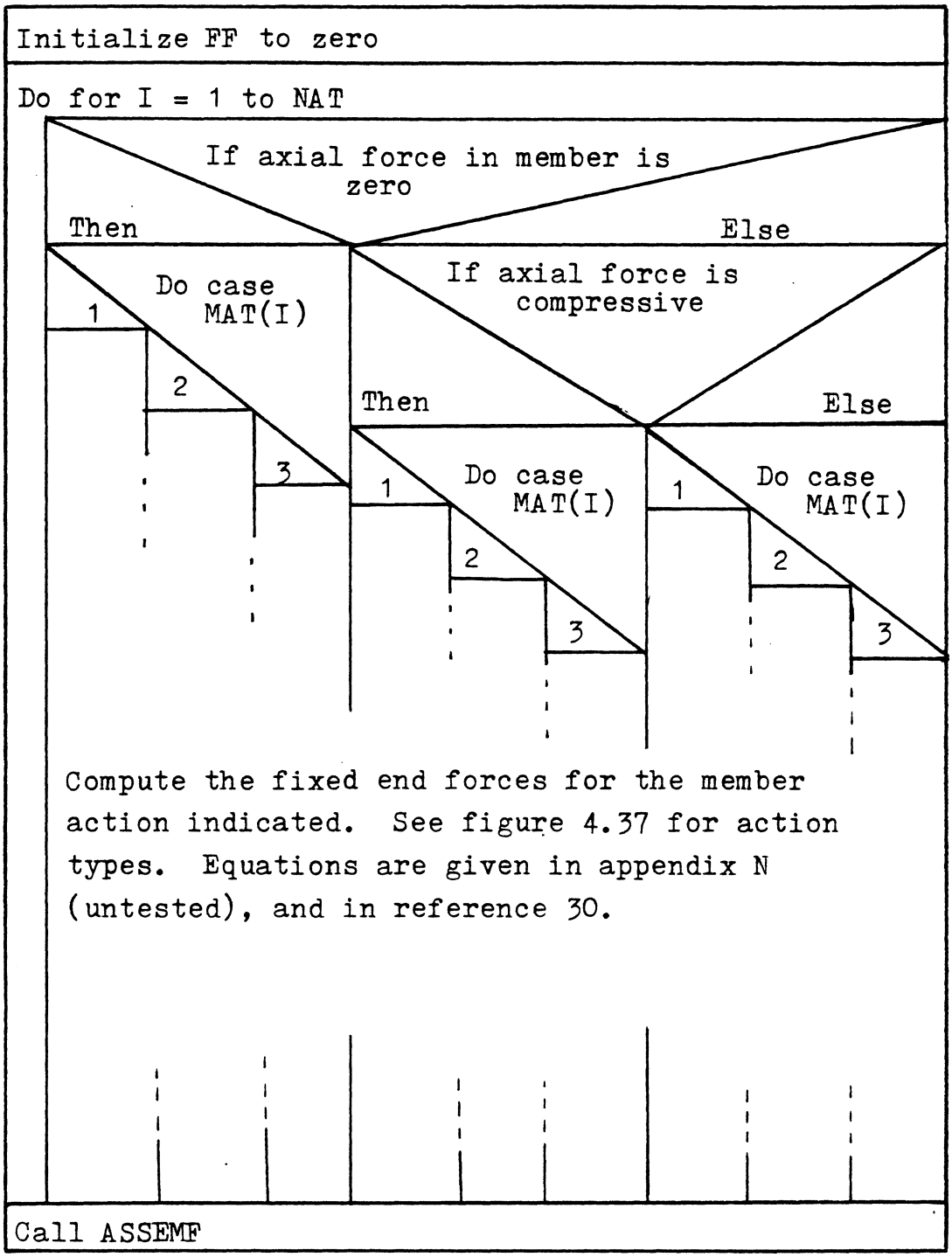


Figure 4.21 : FIXEND

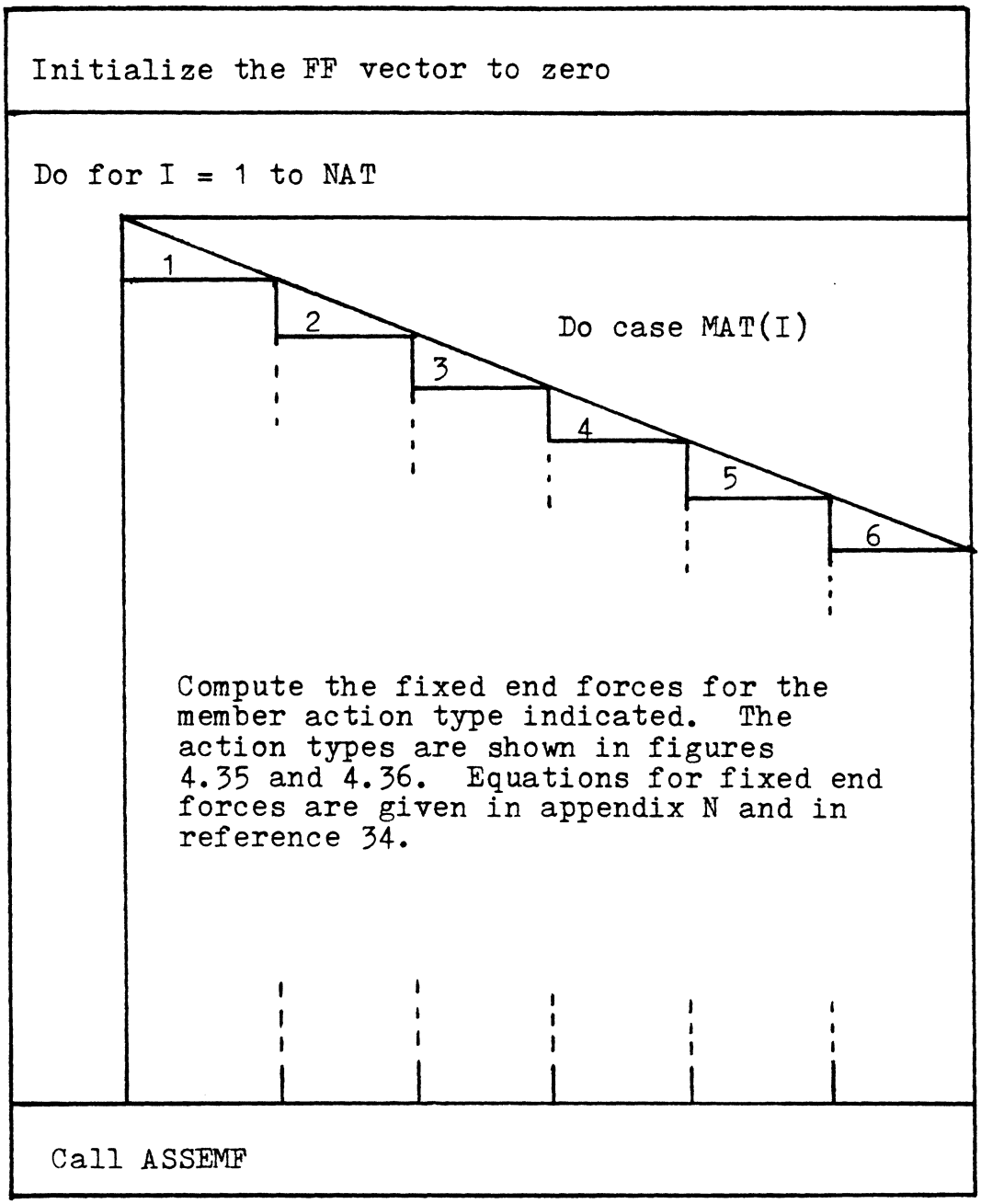


Figure 4.22 : FIXEN2

ASSEMF assembles the fixed end forces and applied joint load vector into the total applied load distribution vector, Q. See figure 4.23.

Input: C1, C2, FF, MCODE, NE, NEQ, QJ.

Output: Q.

SOLVE calls FACTOR if the stiffness matrix has been updated and calls REDUCE and BACSUB to solve a system of linear equations by Gauss elimination. See figure 4.24.

Input: SKT, MAXA, QT, F, NC, NEQ.

Output: DD.

FACTOR is adapted from Bathe (1) to perform the L D U factorization of SKT.

Input: MAXA, SKT, NEQ.

Output: SKT, NEGPIV.

REDUCE is adapted from Bathe (1) to reduce the right-side load vector.

Input: DD, MAXA, SKT, NEQ.

Output: DD.

BACSUB is adapted from Bathe (1) to perform back-substitution to obtain the solution to the system of equations.

Input: DD, MAXA, SKT, NEQ.

Output: DD.

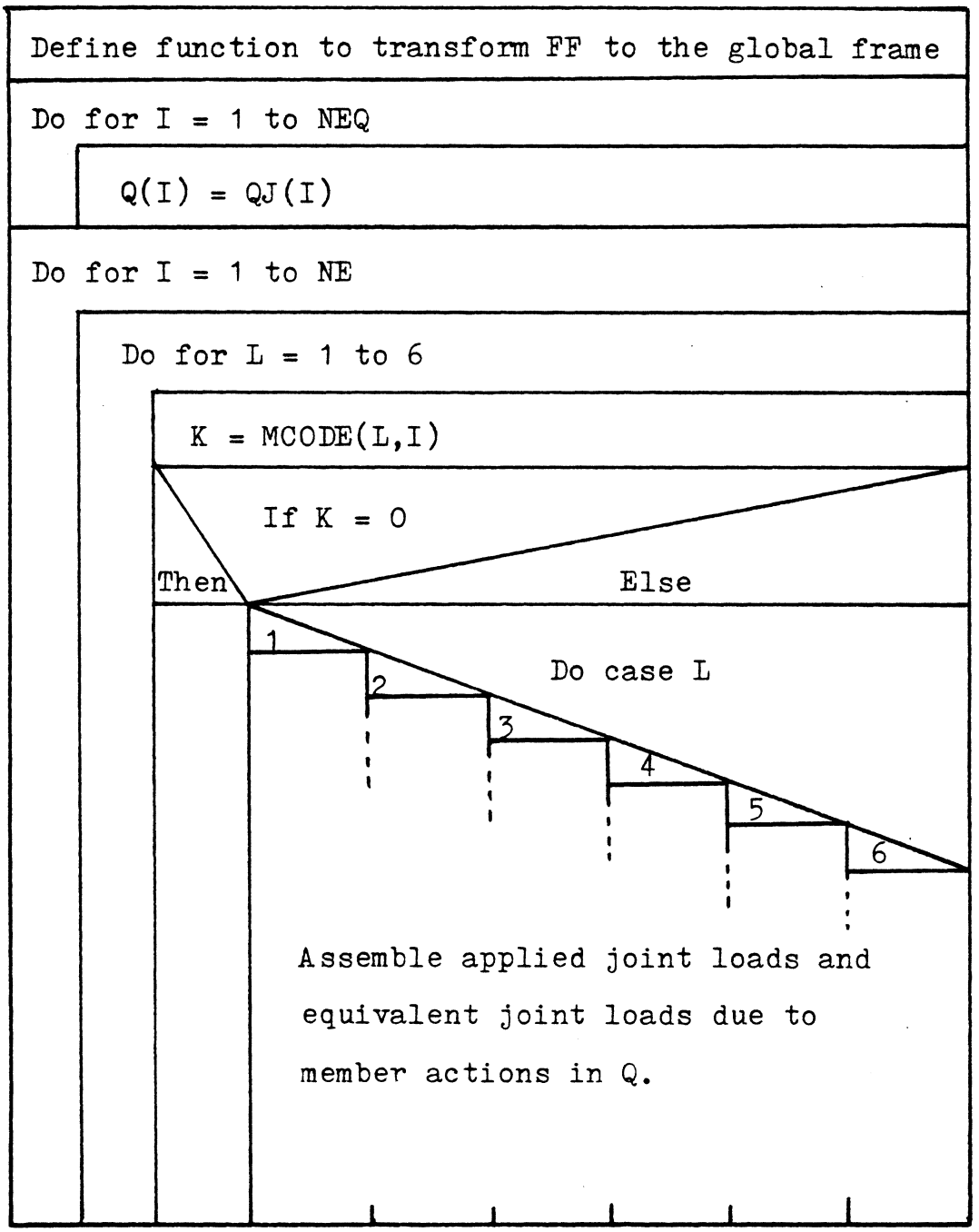


Figure 4.23 : ASSEMF

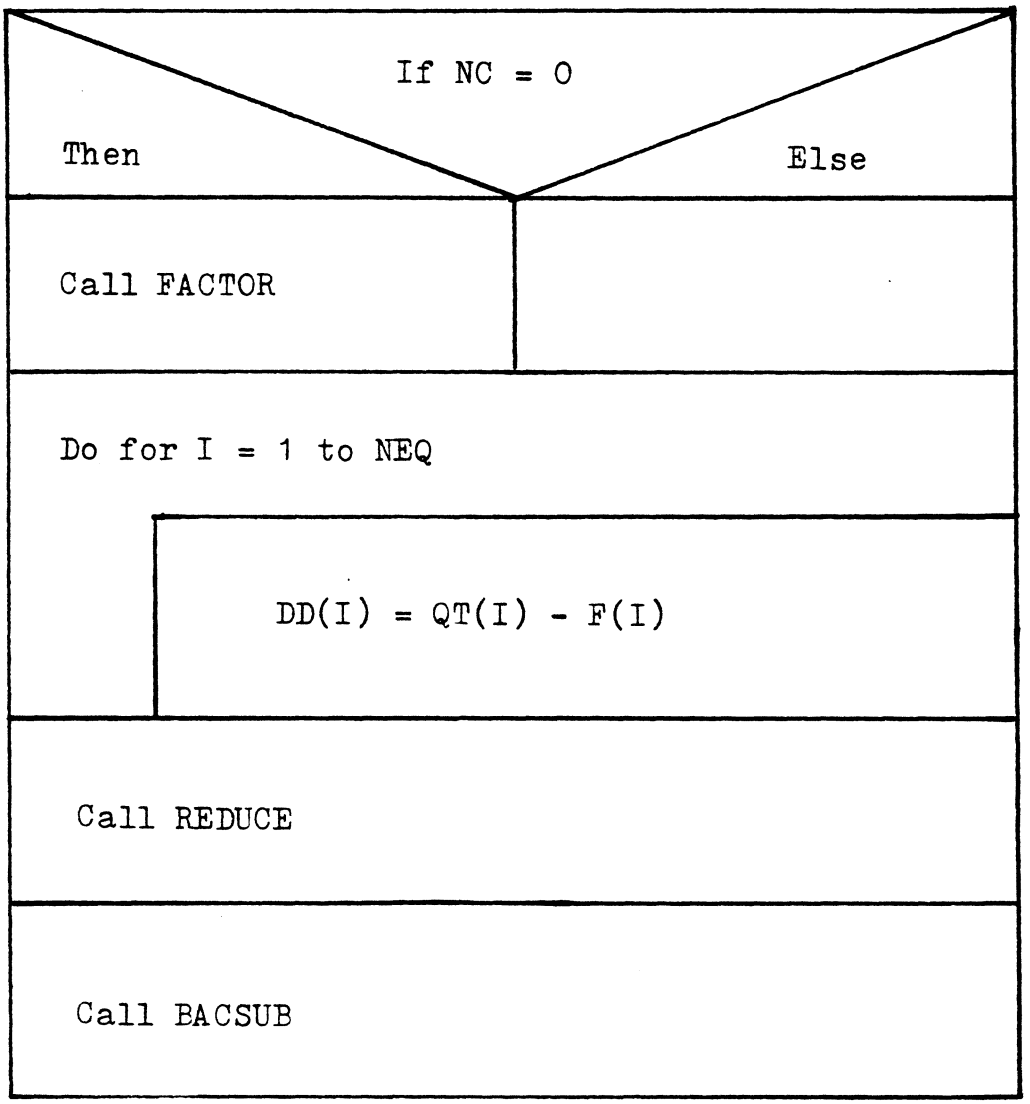


Figure 4.24 : SOLVE

TEST calls ENERGY, DISPLC, DISPLB, and UNBALF as indicated to test for convergence to an equilibrium point. See figure 4.25.

Input: D, DD, DD1, F, FP, FPI, JCODE, QT, CPDB, CPDC, CPE, CPF, DP, NJ, NEQ, ITIND.

Output: ICI.

ENERGY tests for convergence to an equilibrium configuration using internal energy criteria (1). See figure 4.26.

Input: DD, DD1, FP, FPI, QT, CPE, DP, ICI, NEQ, ITIND.

Output: ICI.

DISPLC tests for convergence to an equilibrium configuration using the infinity vector norm of displacements (5). See figure 4.27.

Input: D, DD, JCODE, CPDC, ICI, NJ.

Output: ICI.

DISPLB tests for convergence to an equilibrium configuration using the Euclidean vector norm of displacements (1). See figure 4.28.

Input: D, DD, CPDB, ICI, NEQ.

Output: ICI.

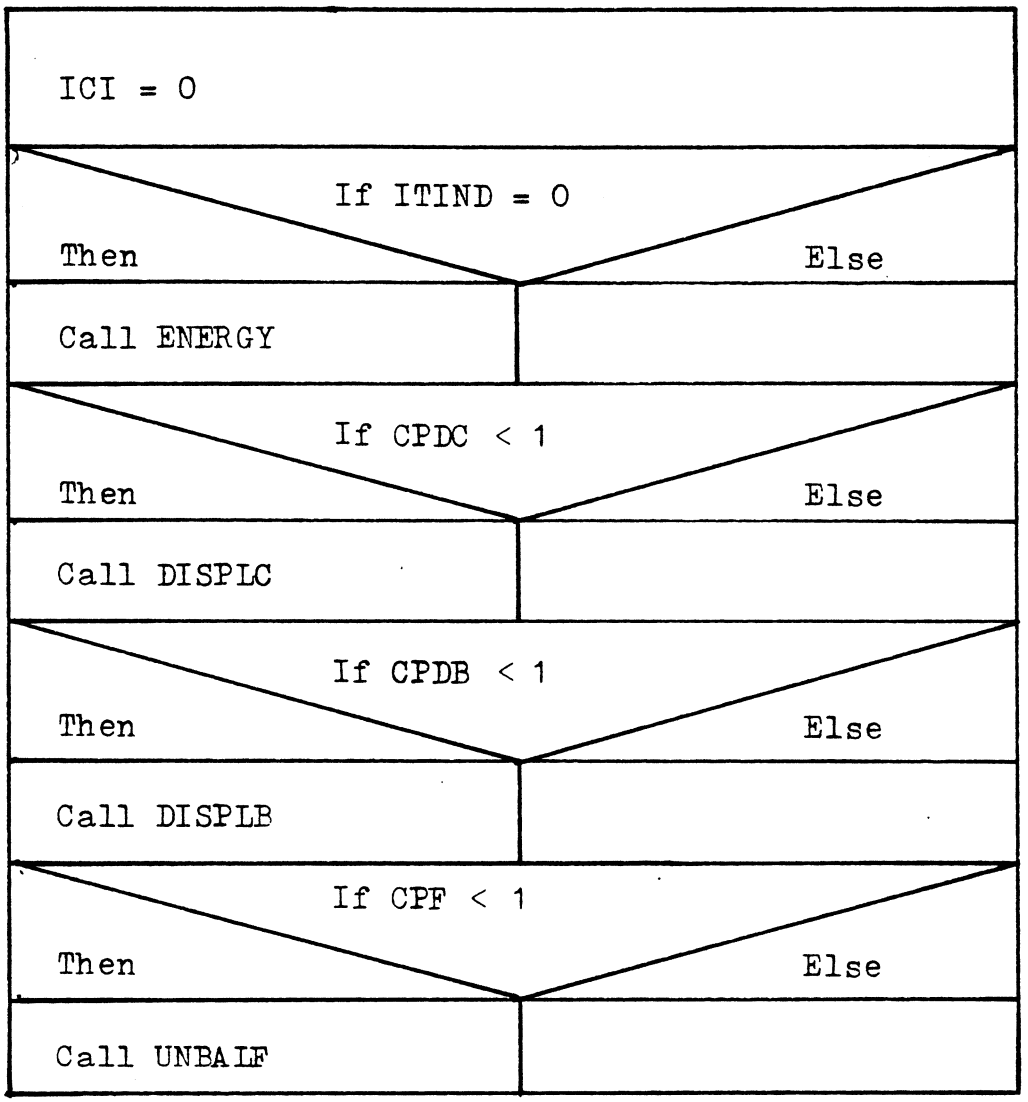


Figure 4.25 : TEST

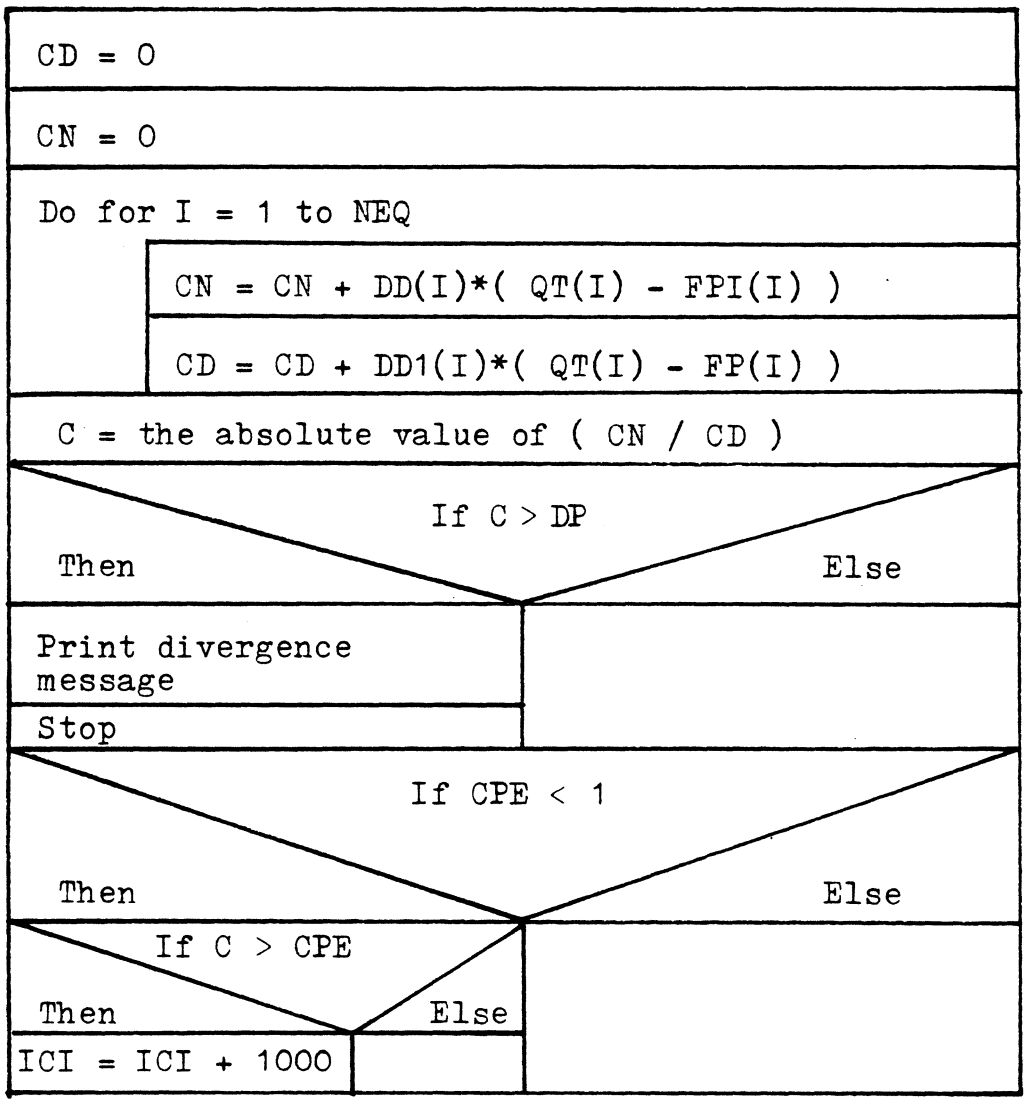


Figure 4.26 : ENERGY

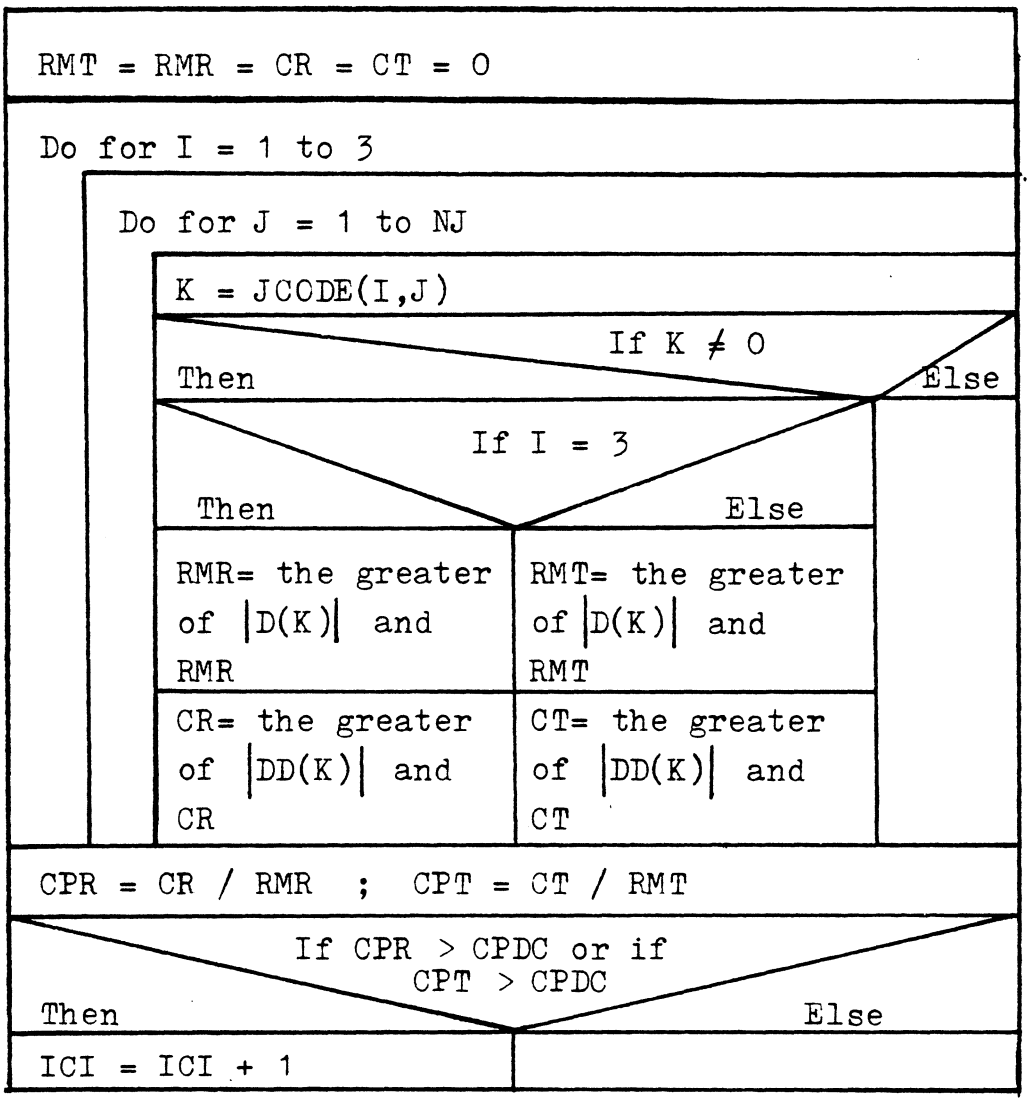


Figure 4.27 : DISPLC

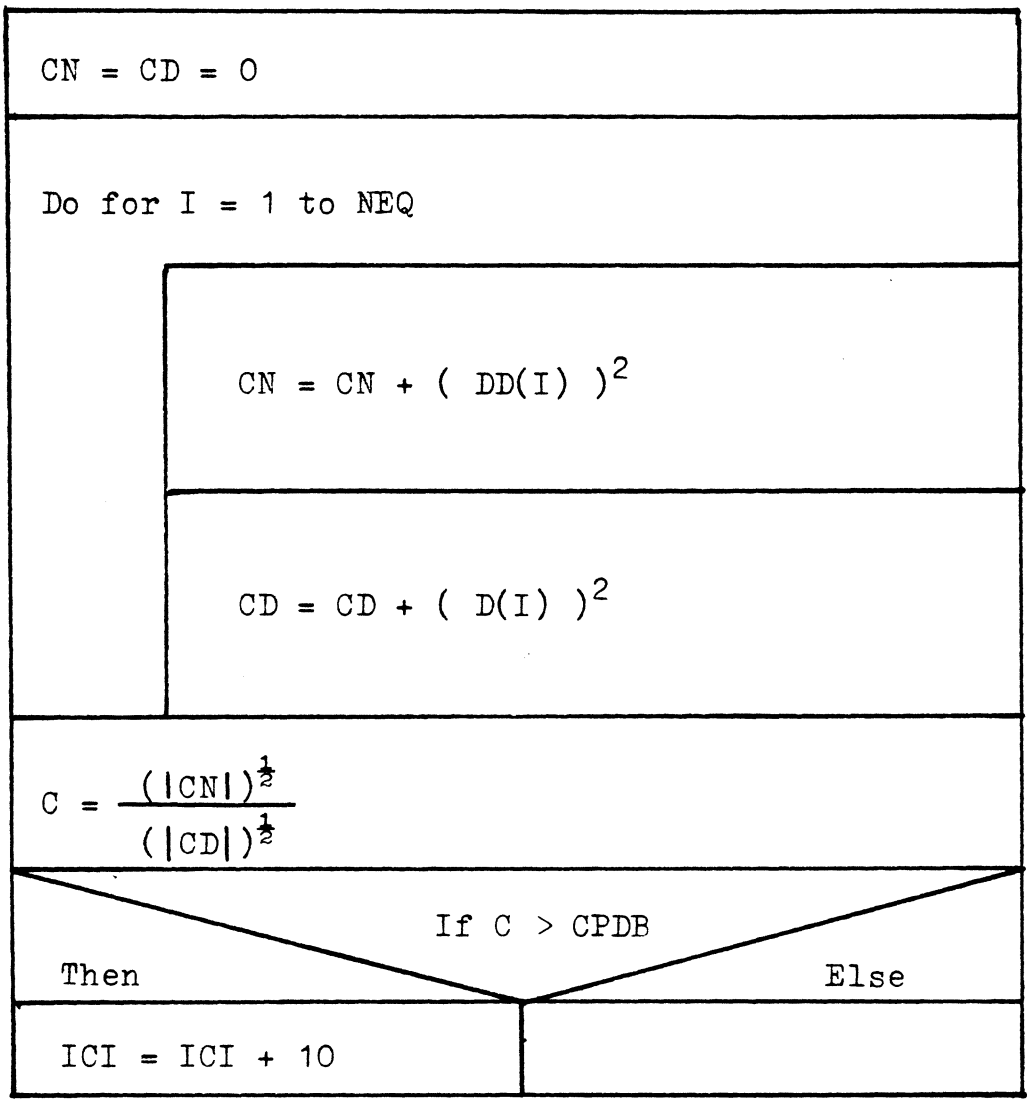


Figure 4.28 : DISPLB

UNBALF tests for convergence to an equilibrium configuration using the unbalanced force criterion. See figure 4.29.

Input: F, FP, QT, CPF, ICI, NEQ.

Output: ICI.

RESULT zeros the joint force matrix and calls JOINTF and OUTPUT. See figure 4.30.

Input: FG, MINC, NE, D, FL, JCODE, QT, NEQ, NJ, QI, IMP, NPRINT.

Output: none.

JOINTF generates the joint force matrix. See figure 4.31.

Input: FG, MINC, NE

Output: P

OUTPUT prints the desired data at each equilibrium point. See figure 4.32.

Input: D, FL, JCODE, P, QT.

Output: none.

DOTPRD computes the dot product of DOT1 and DOT2. See figure 4.33.

Input: DOT1, DOT2, N

Output: DOTPRD

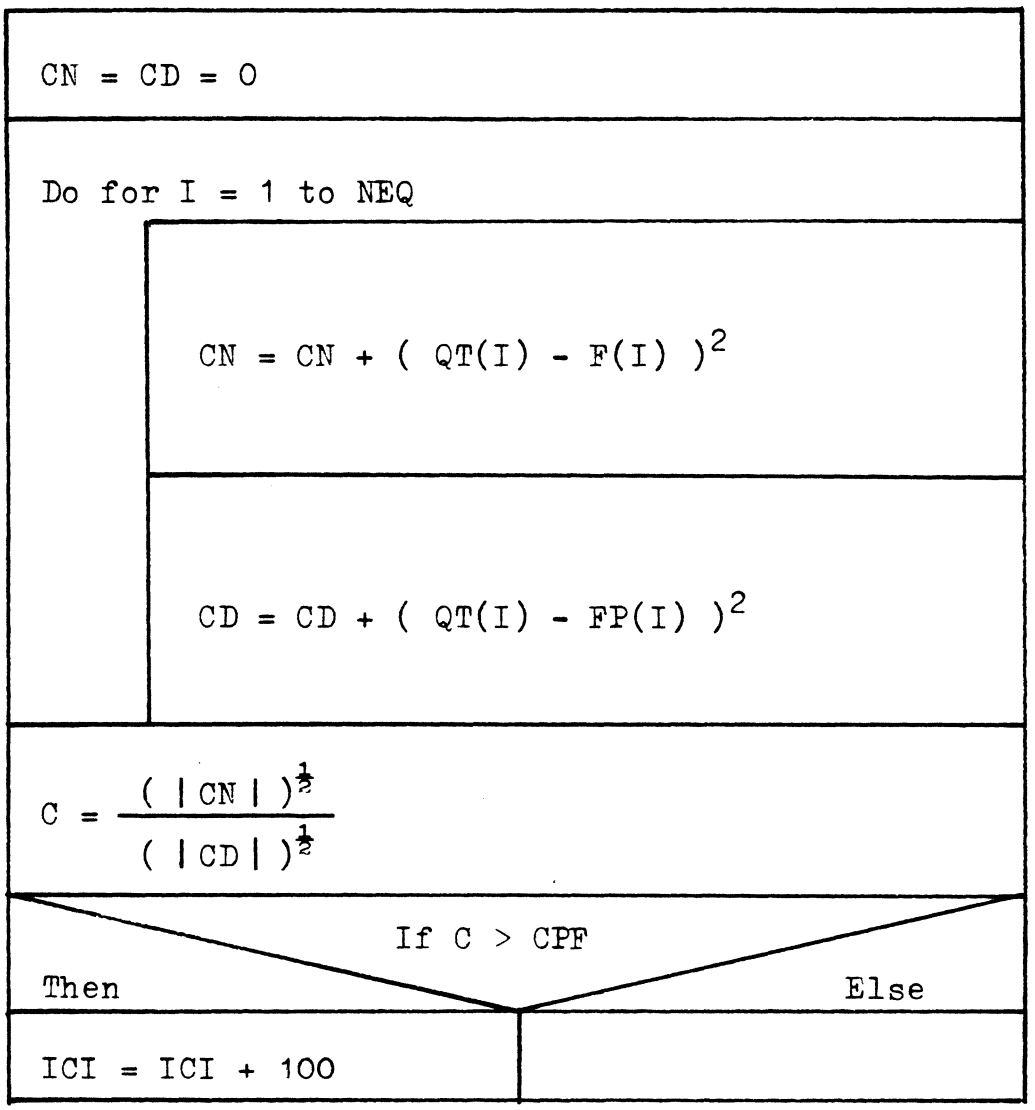


Figure 4.29 : UNBALF

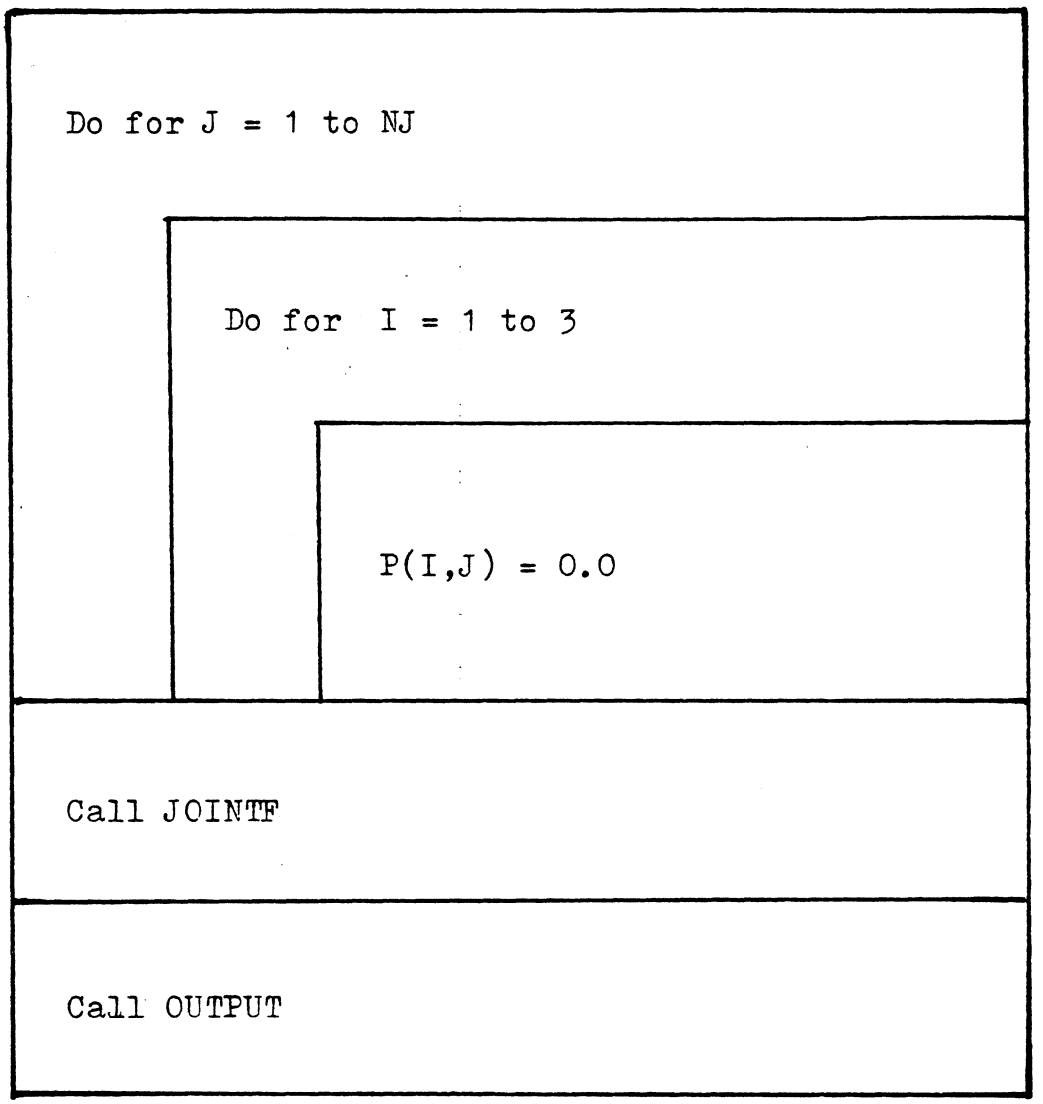


Figure 4.30 : RESULT

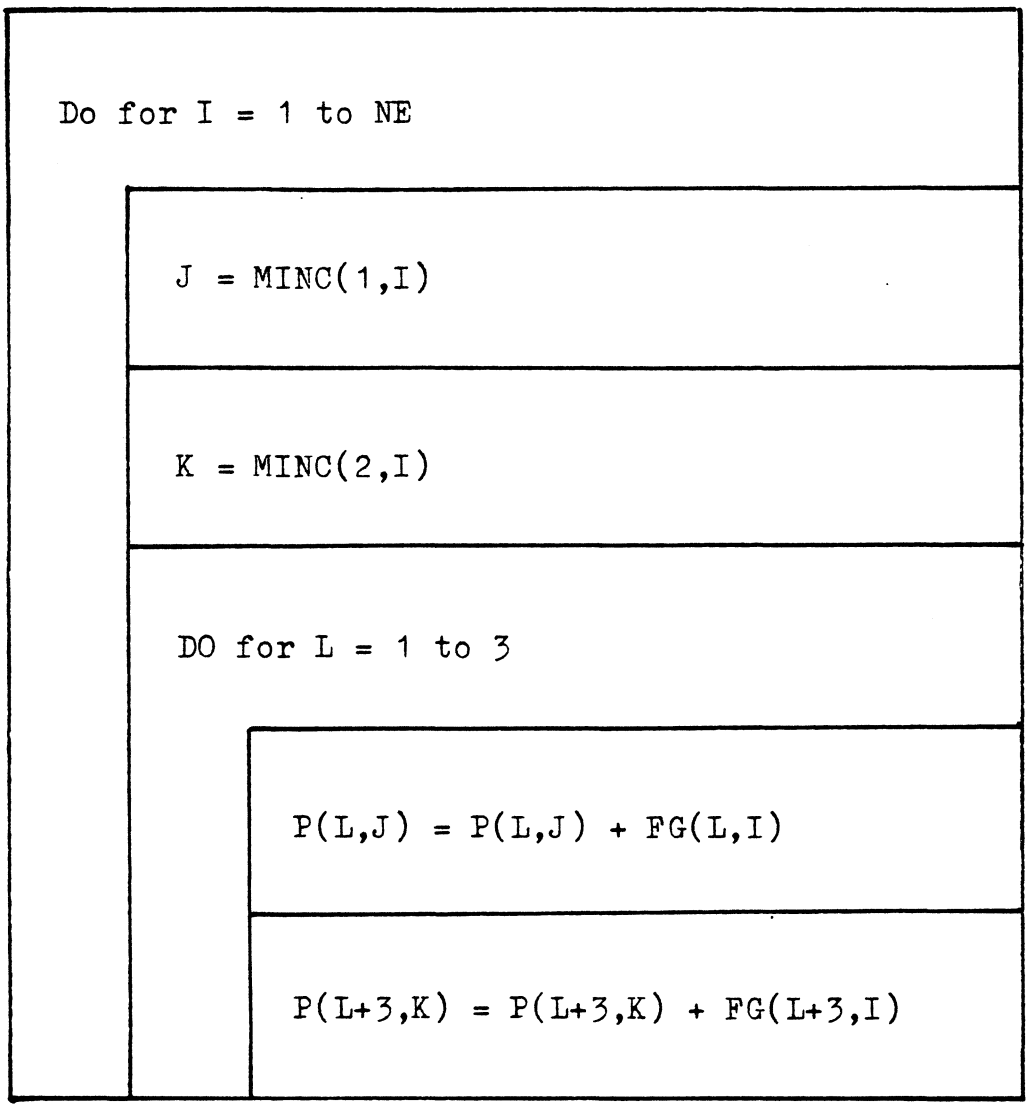


Figure 4.31 : JOINTF

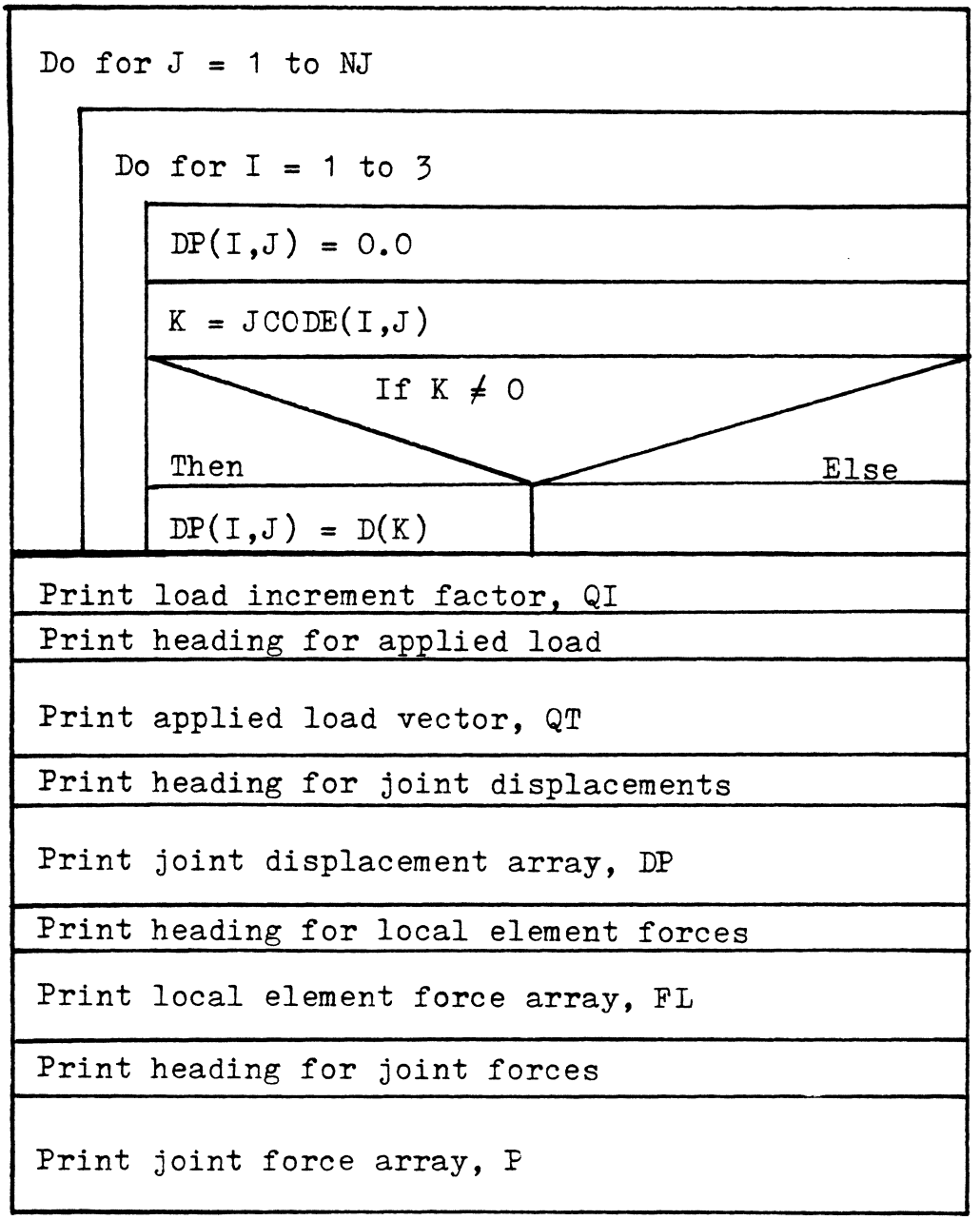


Figure 4.32 : OUTPUT

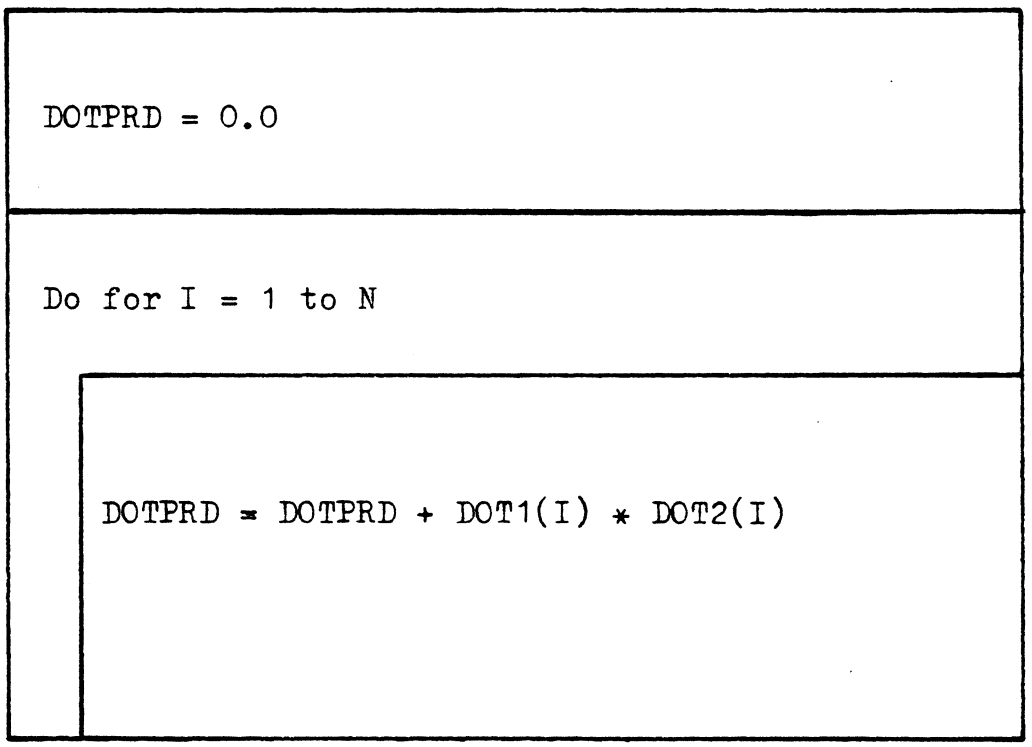


Figure 4.33 : DOTPRD

UPDATE computes the applied load vector, QT, from the load distribution vector, Q, and the load incrementing factor, QI. See figure 4.34.

Input: Q, NEQ, QI.

Output: QT.

Figures 4.35 and 4.36 show the member actions for the finite element model, and figure 4.37 shows the member actions for the beam-column model.

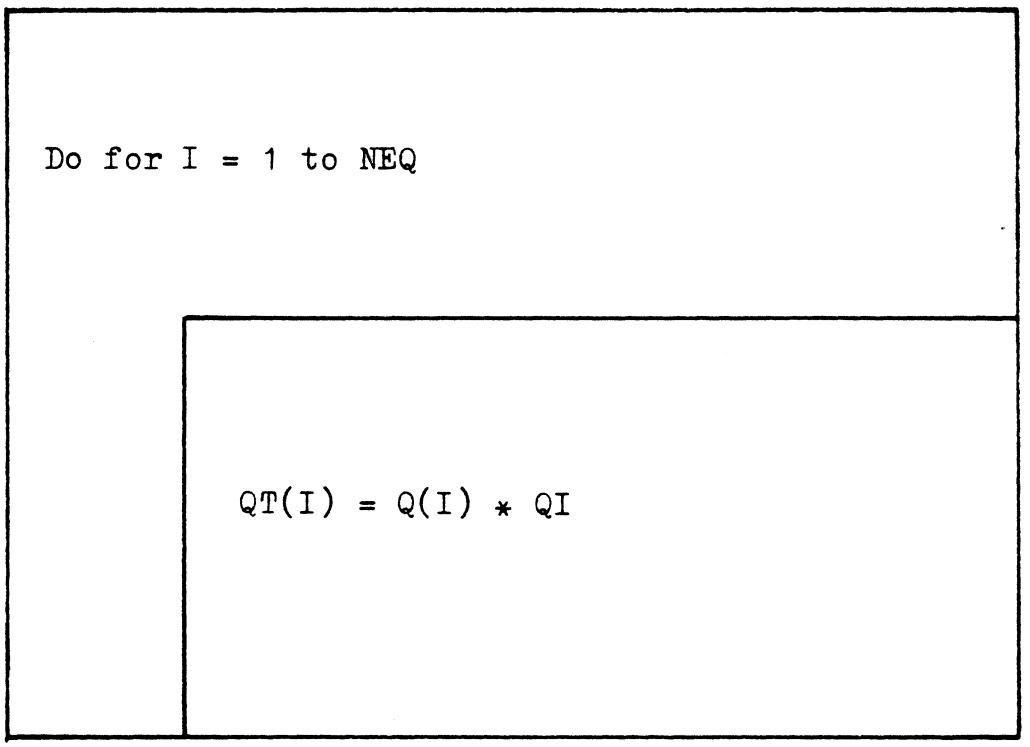
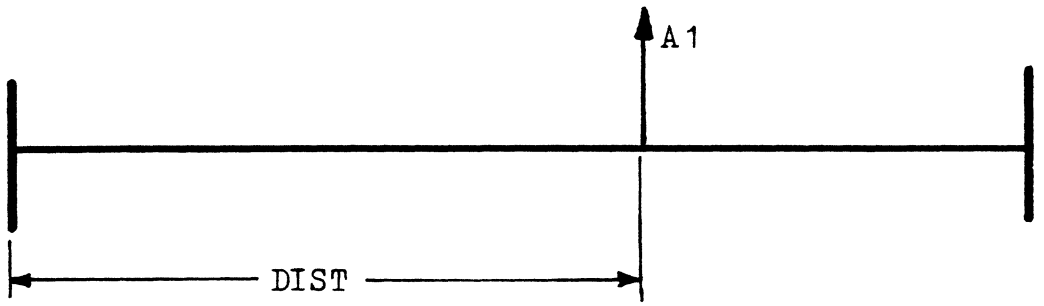
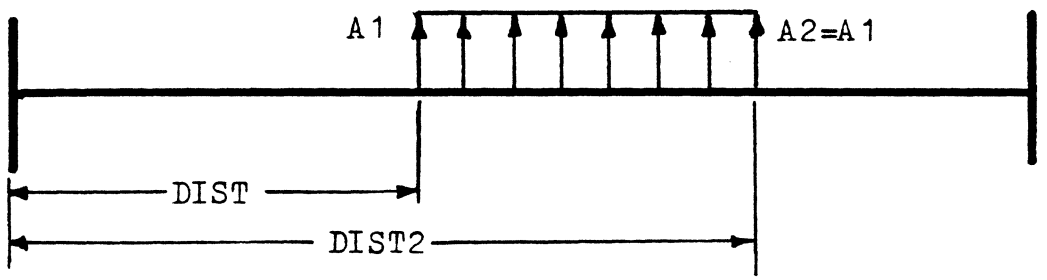


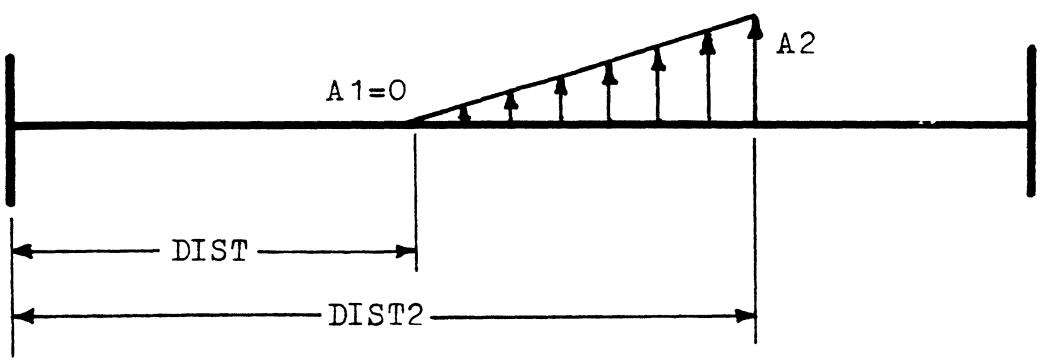
Figure 4.34 : UPDATE



(a) MAT = 1



(b) MAT = 2



(c) MAT = 3

Figure 4.35 : Member Actions for the Finite Element Model I

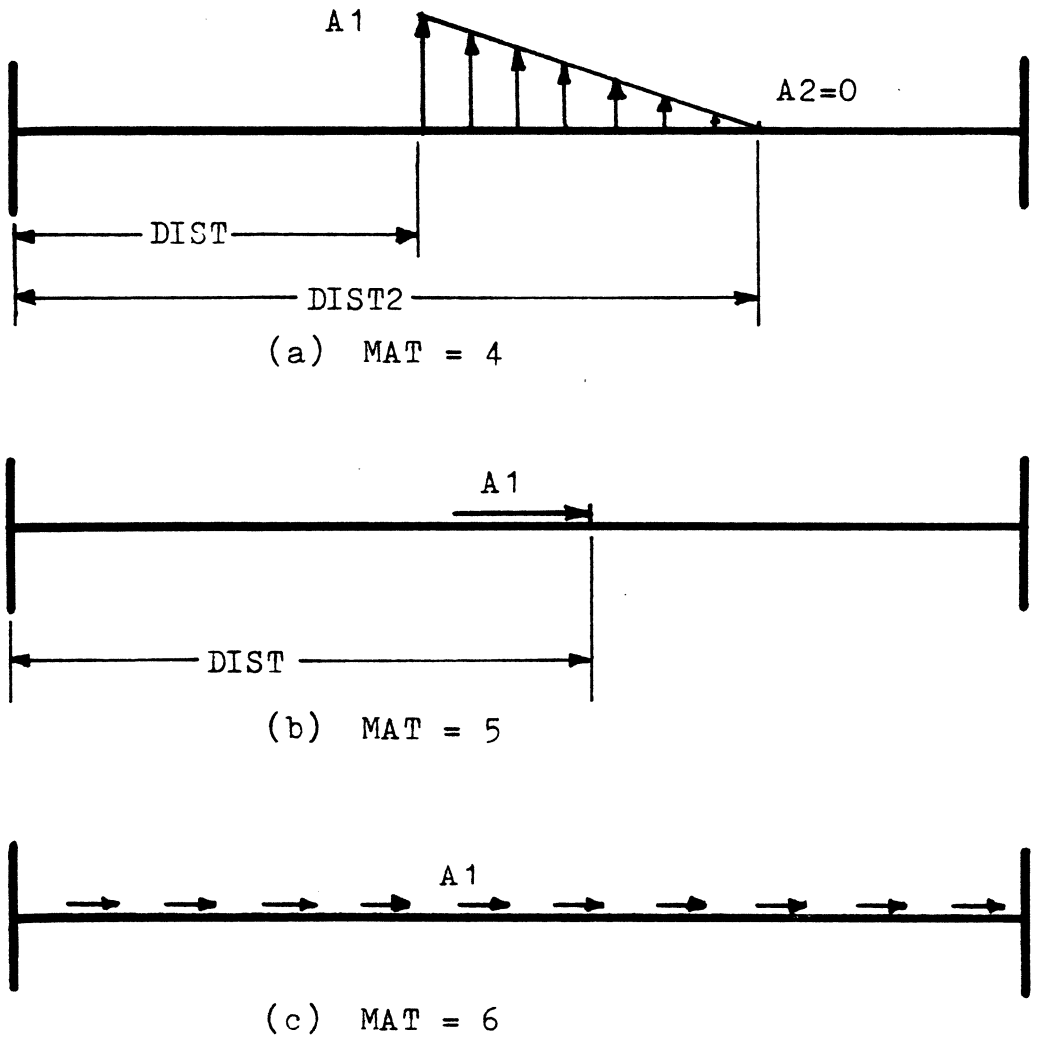


Figure 4.36 : Member Actions for the Finite Element Model II

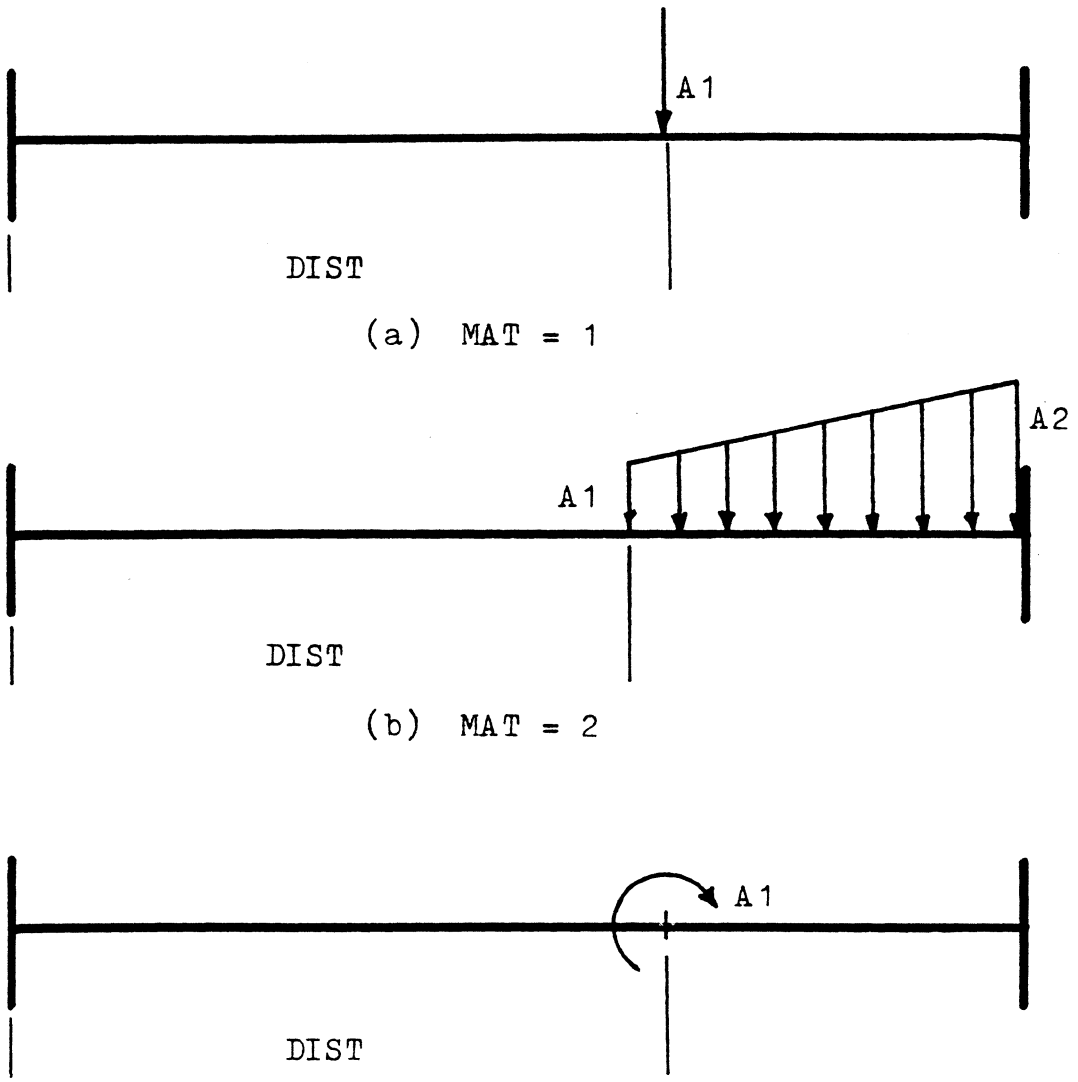


Figure 4.37 : Member Actions for the Beam-Column Model

CHAPTER 5

TEST PROBLEM RESULTS

5.1 Introduction

All of the test problems are analyzed using the modified Riks/Wempner method with the following parameters :

$$\begin{aligned} CPF &= 0.1 \\ CPDC &= 0.01 \\ CPDB &= 0.001 \\ FQR &= 1 * 10^{-9} \\ CPE &> 1 \end{aligned} \tag{5.1}$$

Excepting plots of tangents for trial structures 1 and 2 with only K_0 in effect, analysis is performed using ITDES = 1.

The results of separate studies cannot be compared accurately when the results are given only in plots. For the benefit of future studies, the values used to generate some of the plots presented in this chapter are given in appendix M.

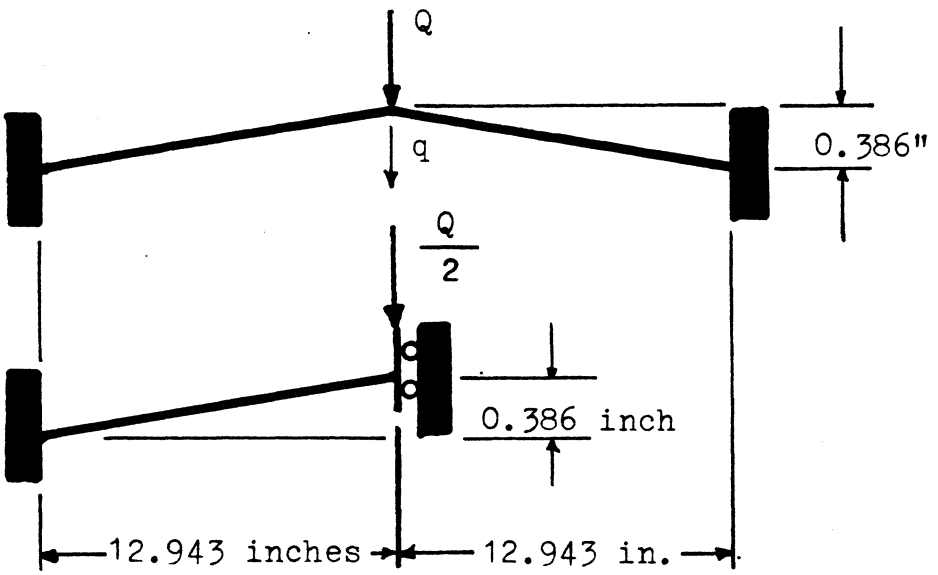
5.2 Test Problem 1

The first test problem is a two bar "toggle" frame which has been investigated by Williams (38) and by others (18, 27, 40). The configuration, initial load increment, and material and geometric properties are

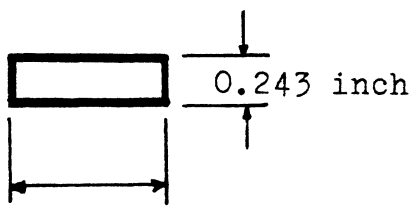
shown in figure 5.1. Since the deformation is governed by symmetric buckling modes (38), analysis of half of the structure is sufficient. Reference to the number of elements used for the analysis corresponds to the number of elements used to model the half-structure. Values plotted for the load correspond to the load applied to the whole structure.

In figure 5.2, the finite element model is used to compare the effects of mesh refinement. One, six, and twelve elements are used to model the half-structure, and the load values plotted are the loads on the whole structure. The meshes produce similar results until deformations approach the first limit point. After this point, one element is too stiff, but six elements produce a path that is a close approximation to the results for twelve elements.

Figure 5.3 presents a comparison of one beam-column element to twelve finite elements. It is obvious in this plot that the beam-column model produces a similar level of accuracy using a much coarser mesh. this result is expected because the beam-column model contains the actual solution to the differential equation for a beam-column, while the finite element model approximates the element configuration using interpolation functions.



Cross section :



Deflection of degree of freedom "q" (above) is plotted.

$A = 0.183 \text{ in.}^2$

$E = 10{,}300. \text{ kip/in.}^2$

$I = 0.00090039 \text{ in.}^4$

$\Delta \lambda^0 = 0.004$ kips

Figure 5.1 : Test Problem 1

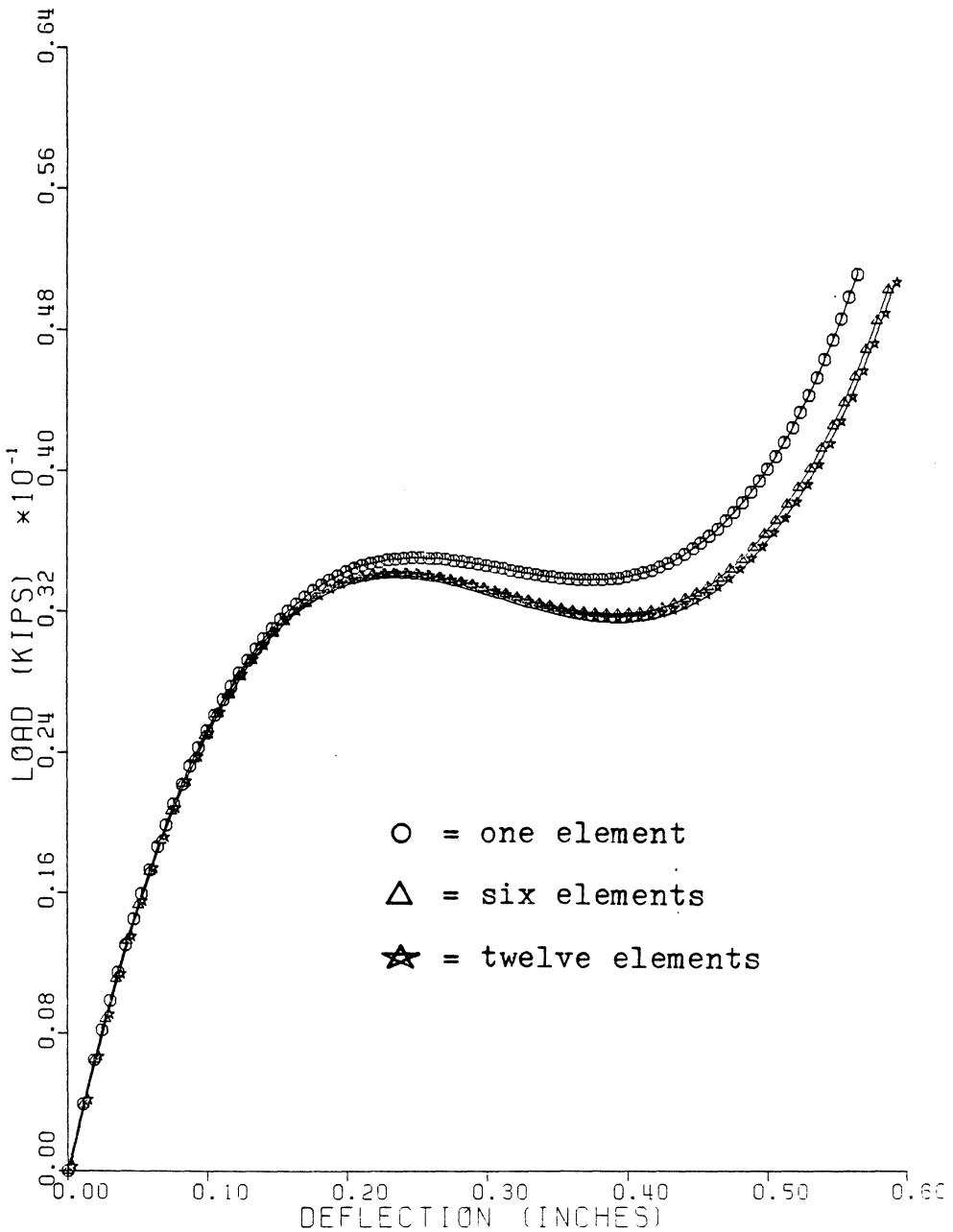


Figure 5.2 : Finite Element Model

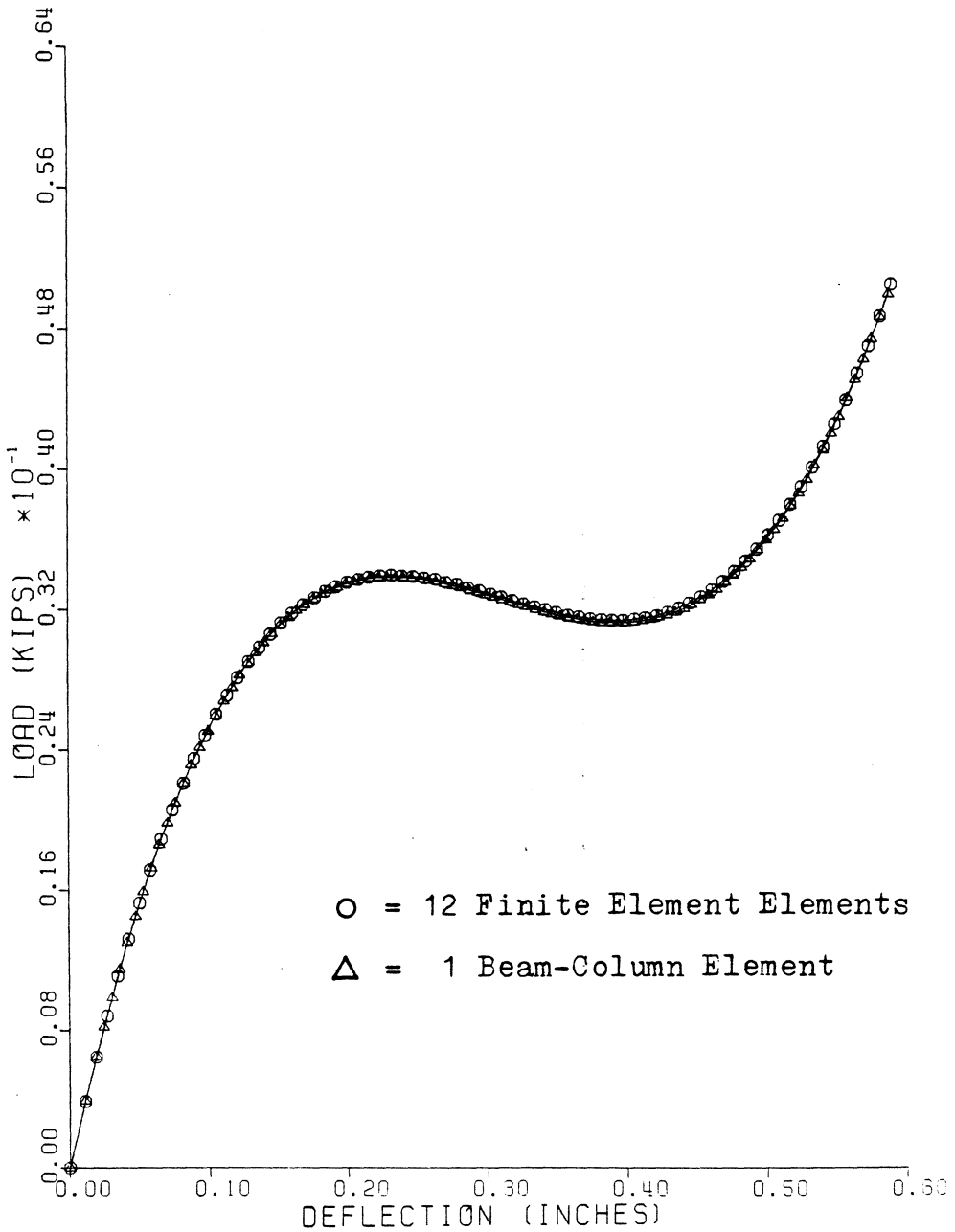


Figure 5.3 : Comparison of Models

The curves in this figure predict the same equilibrium path that is shown in the literature.

A model's representation of the tangent stiffness may be examined by plotting the "tangents" to the equilibrium path predicted by the model. A plot of tangents may be generated by plotting the initial arc length, Δs, of the modified Riks/Wempner method at each equilibrium point. The effect of varying the composition of the tangent stiffness matrix may also be studied using this technique.

Figures 5.4 and 5.5 are obtained using the beam-column model. Figure 5.4 depicts the tangents at each equilibrium point when the tangent stiffness matrix is composed of the conventional stiffness matrix used in linear analysis and the initial stress stiffness matrix. It is obvious that this composition of the stiffness matrix produces a good estimator of the equilibrium path for the search for the next equilibrium point. The tangents in figure 5.4 are so close to the equilibrium path itself that it is difficult to distinguish between the equilibrium path and the plot of tangents. Figure 5.5 shows the estimates of the tangents obtained when only the conventional linear stiffness matrix is used. Figures 5.6 and 5.7 show the equilibrium path obtained by six finite elements.

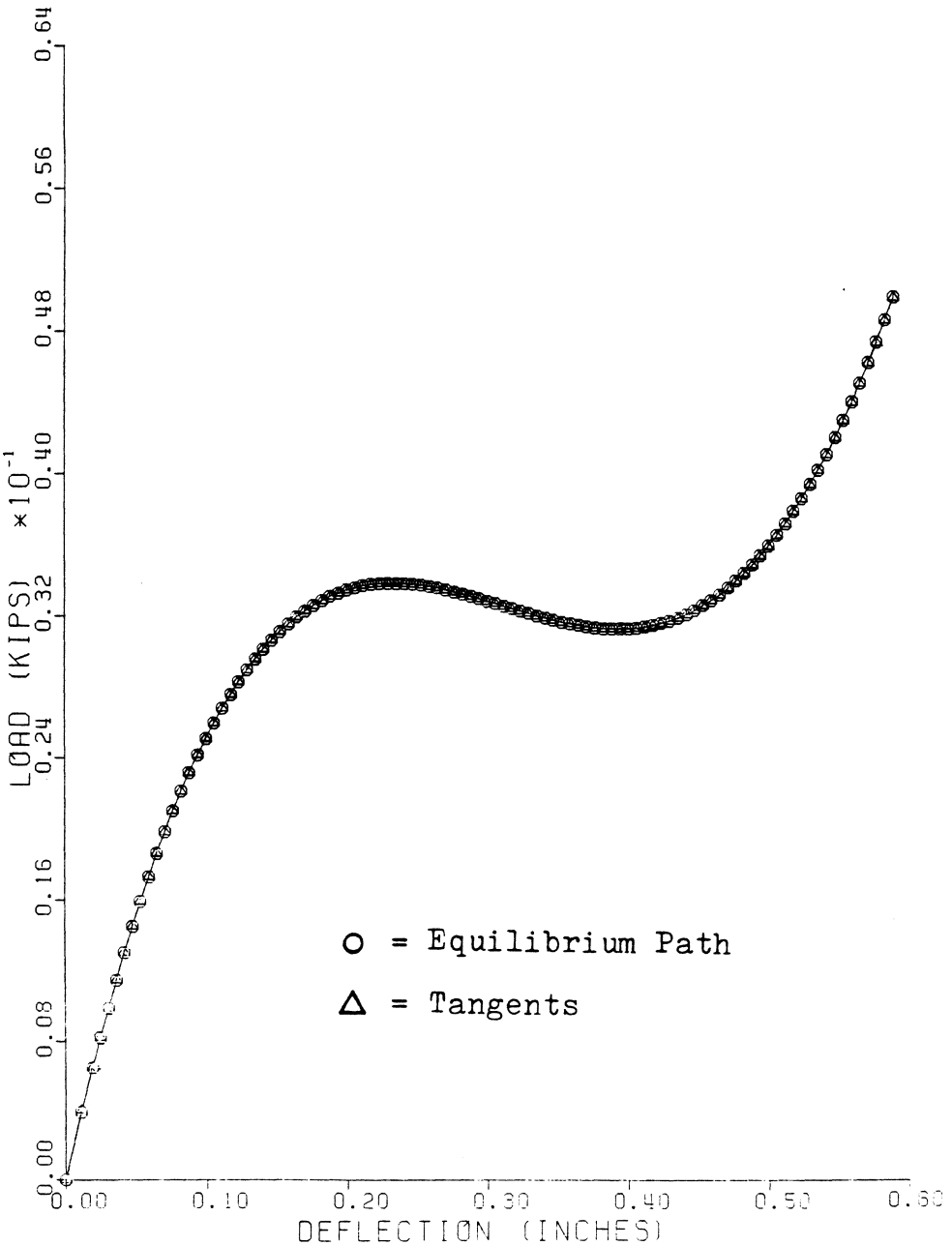


Figure 5.4 : Beam-Column Model, Full Tangent Stiffness

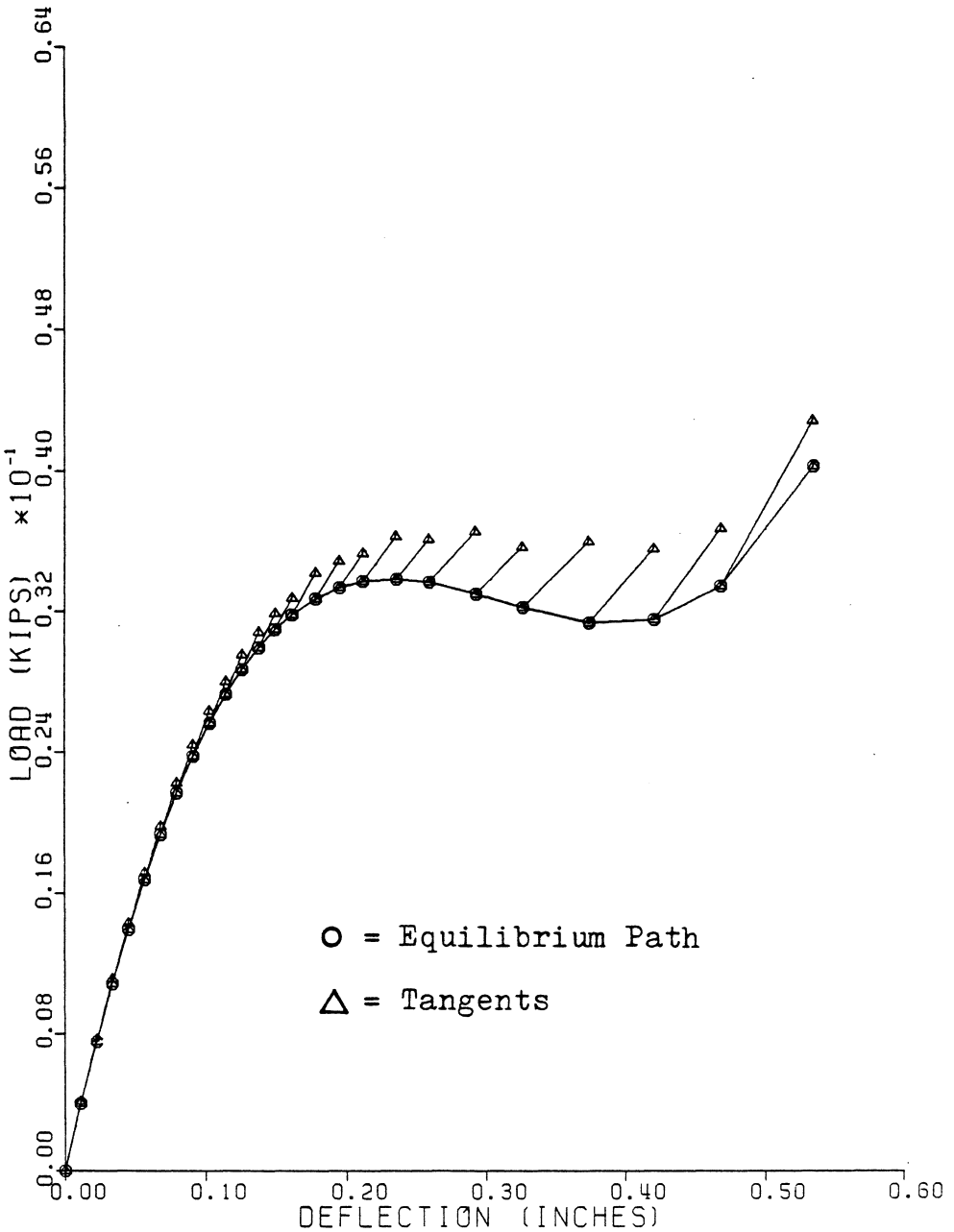


Figure 5.5 : Beam-Column Model,

K_0 Only

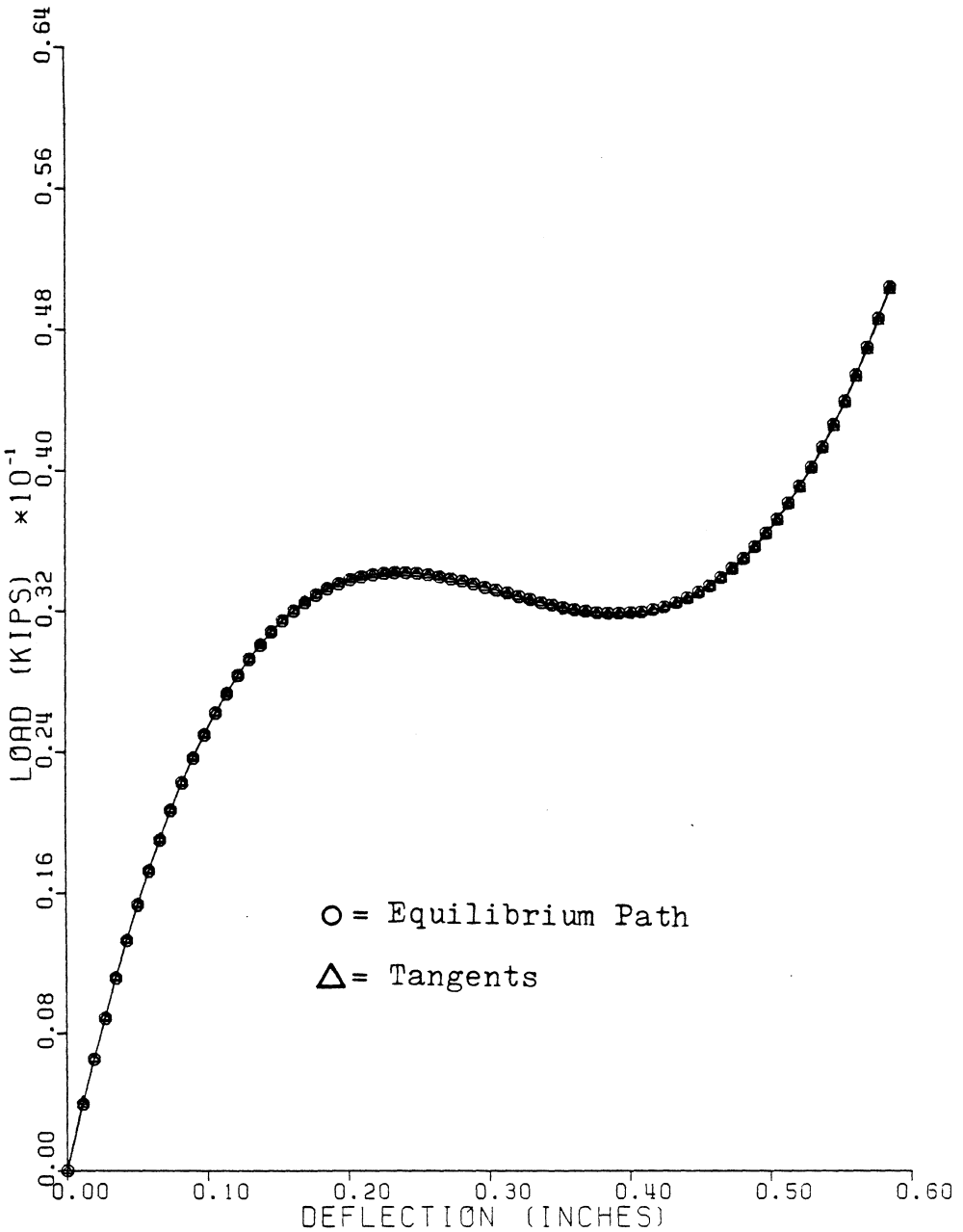


Figure 5.6 : Finite Element Model, Full Tangent Stiffness

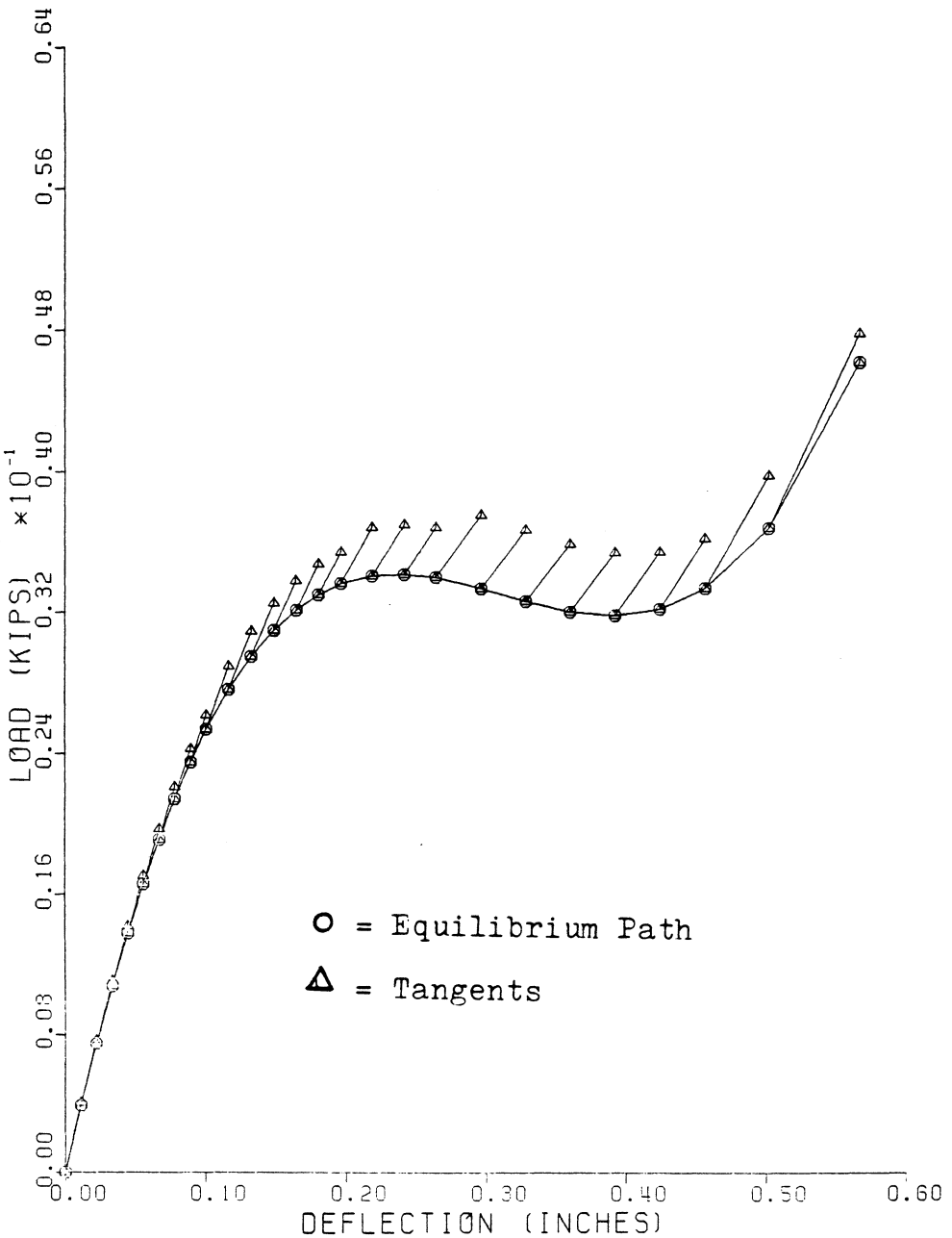


Figure 5.7 : Finite Element Model,

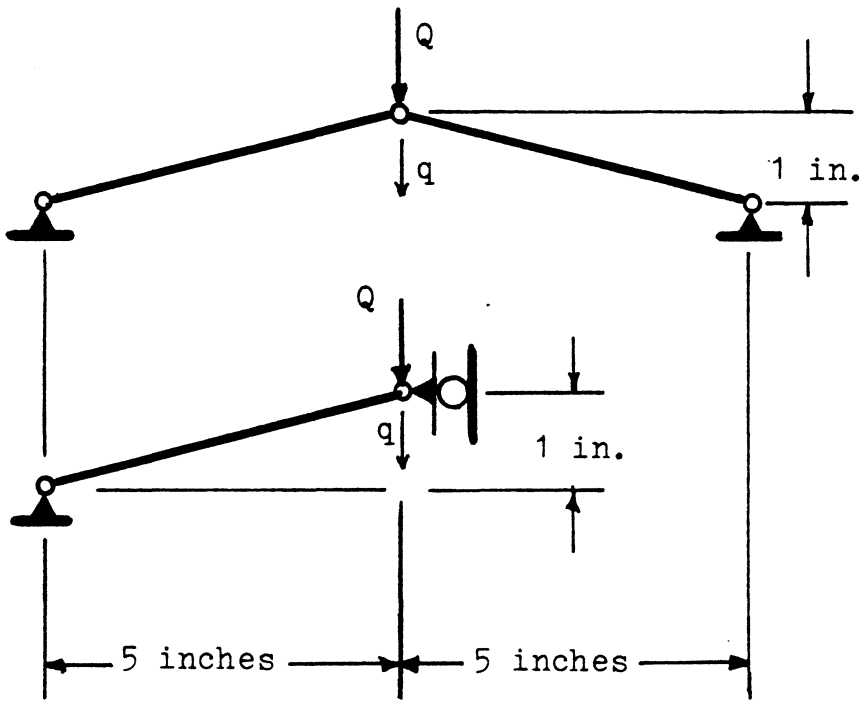
K_0 Only

The tangents to the curve at each equilibrium point are also plotted in these figures. The tangents are composed of both the conventional stiffness matrix and the initial stress stiffness matrix in figure 5.6, while those in figure 5.7 represent the conventional stiffness matrix only.

When the initial stress stiffness is neglected, both models obviously contain poor predictors of the behavior. Notice that the instability between the first and second limit points is not indicated by this representation of the tangent stiffness matrix. Also note that the true equilibrium path is obtained regardless of the fact that the tangent stiffness matrix is a poor predictor. Recall that the equilibrium path is determined by the element force equations (or the secant stiffness matrix) and convergence criteria, and the tangent stiffness matrix is used merely as a predictor of the next equilibrium point and to detect instability.

5.3 Test Problem 2

The second test problem is the two bar truss, or "vonMises' truss", studied in references 6, 8, and 37. The configuration, initial load increment, and geometric and material properties are shown in figure 5.8. Note that the moment of inertia does not contribute to the



$$A = 1.0 \text{ inch}^2$$

$$E = 1.0 \text{ lb./in.}$$

$$I = 1.0 \text{ inch}^4$$

$$\Delta \lambda^0 = 0.0003 \text{ lb.}$$

In this study, one element is used to model the half-structure for all analyses of this problem.

Figure 5.8 : Test Problem 2

behavior of the idealized truss, but the value specified for it should have the same order of magnitude as the area and modulus of elasticity to prevent overflow or underflow errors. Since the models used are frame models, the moment of inertia will be involved in computations even though the models are applied to a truss. The closed form solution for this problem is given by Hansen (6). The results obtained by this study are the same as those presented in the literature. The symmetric buckling mode governs the behavior of this structure, so analysis of the half-structure is sufficient.

The finite element models used in this study were developed for frame structures, but the boundary conditions for the half-structure can be set to eliminate bending (see figure 5.8). If this were not the case, it would be necessary to modify the member code matrix (see appendix N) to accurately model a truss using a frame analysis program (7). When the solution process converges to an equilibrium configuration, bending deformations must be eliminated in order to satisfy the boundary conditions. An interesting question arises as to whether a frame structure model can initially predict a trial configuration from which bending deformations are excluded.

Figure 5.9 presents a comparison of the beam-column model to the finite element model. Both models give the same equilibrium path and the same equilibrium points. This result is not surprising since the interpolation functions for axial behavior used in the finite element model satisfy the homogeneous differential equation for a truss element (7).

Figures 5.10 and 5.12 show that the tangent stiffness matrices composed of both the conventional stiffness and the initial stress stiffness are accurate predictors in the beam-column and the finite element models respectively. Figure 5.11 shows the effect of neglecting the initial stress stiffness matrix on the tangent stiffness matrix for the beam-column model, and figure 5.13 shows this effect for the finite element model. Notice that instability is never detected by the tangent stiffness matrix when it is composed only of the conventional stiffness matrix.

A poor representation of the tangent stiffness matrix can be very deceiving. As is shown by this and by the previous test problem, a model which uses an incorrect tangent stiffness matrix can produce the correct equilibrium path as long as the secant matrix (containing element force equations) is correct. Unless tangents are plotted, or a postbuckling analysis is performed and

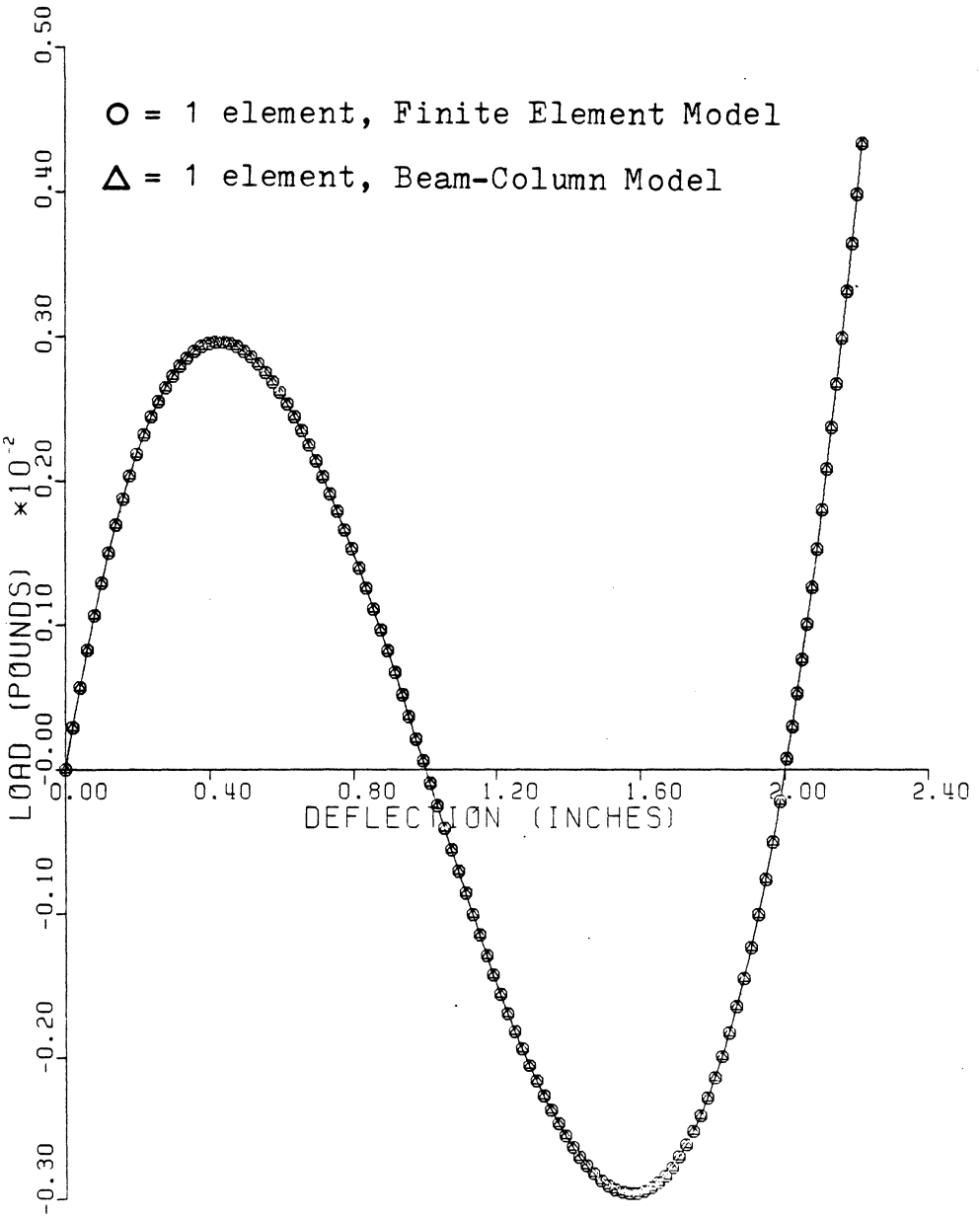


Figure 5.9 : Comparison of Models

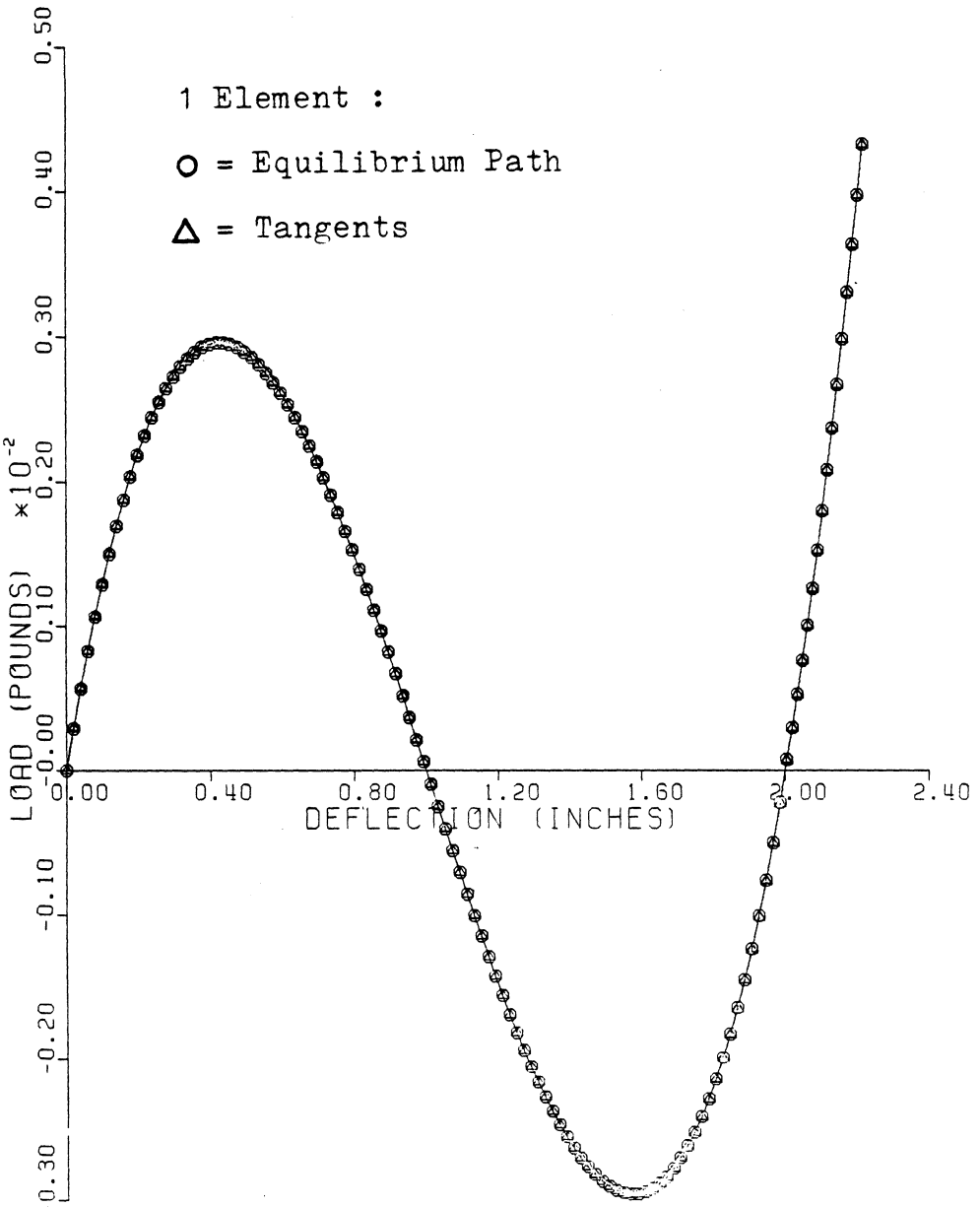


Figure 5.10 : Beam-Column Model,

$K_0 + K_\sigma$

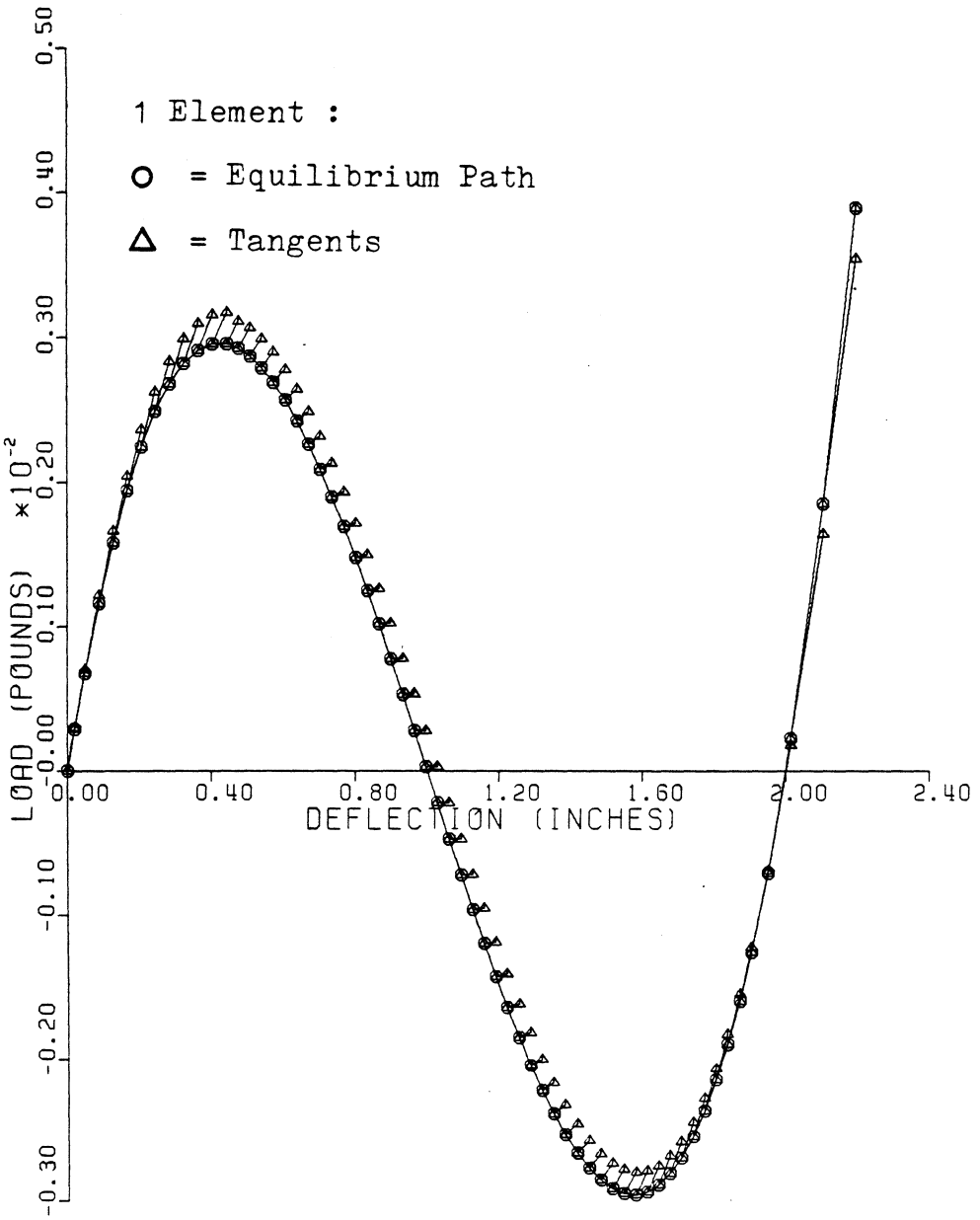


Figure 5.11 : Beam-Column Model,

K_0 Only

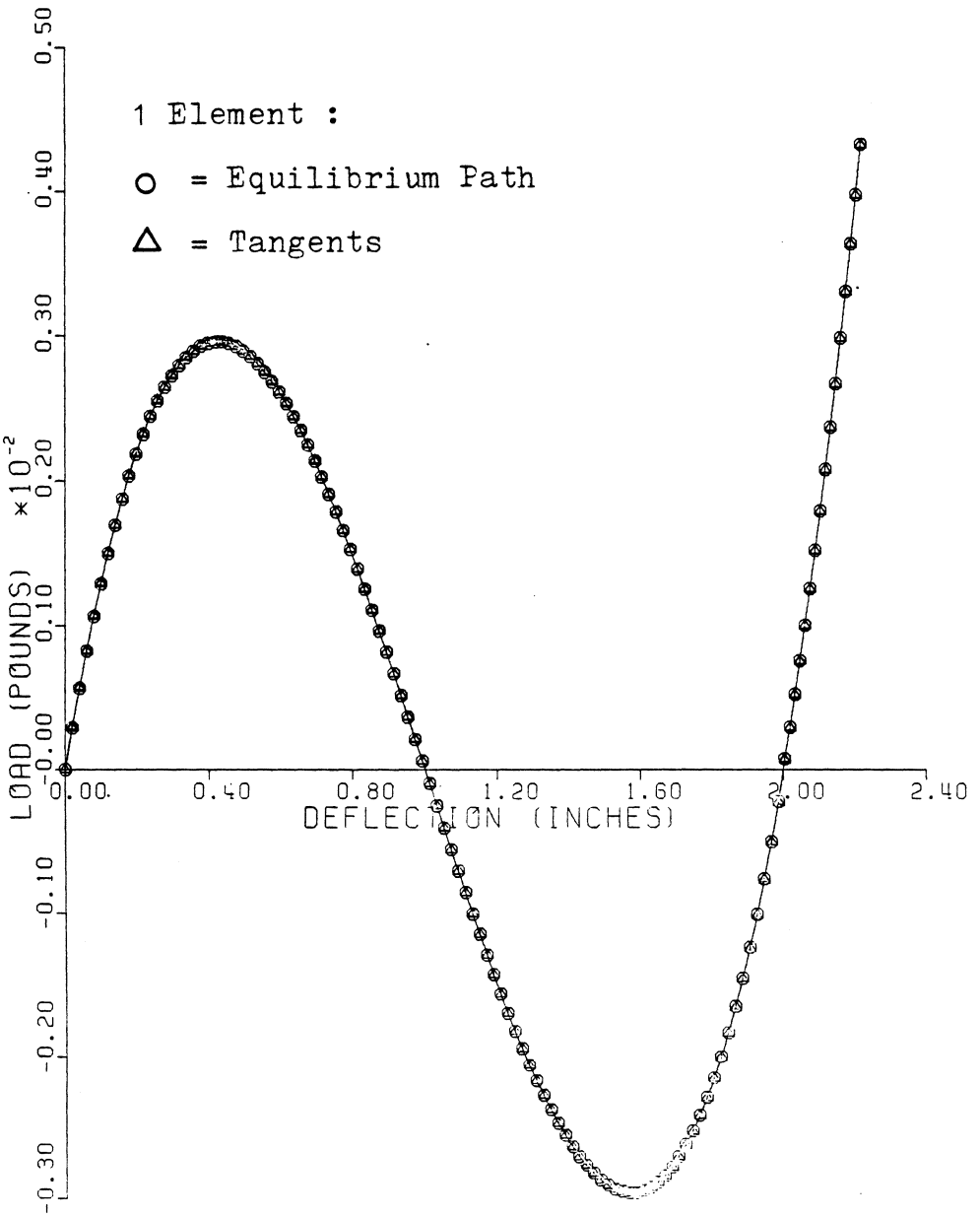


Figure 5.12 : Finite Element Model,

$K_0 + K_\sigma$

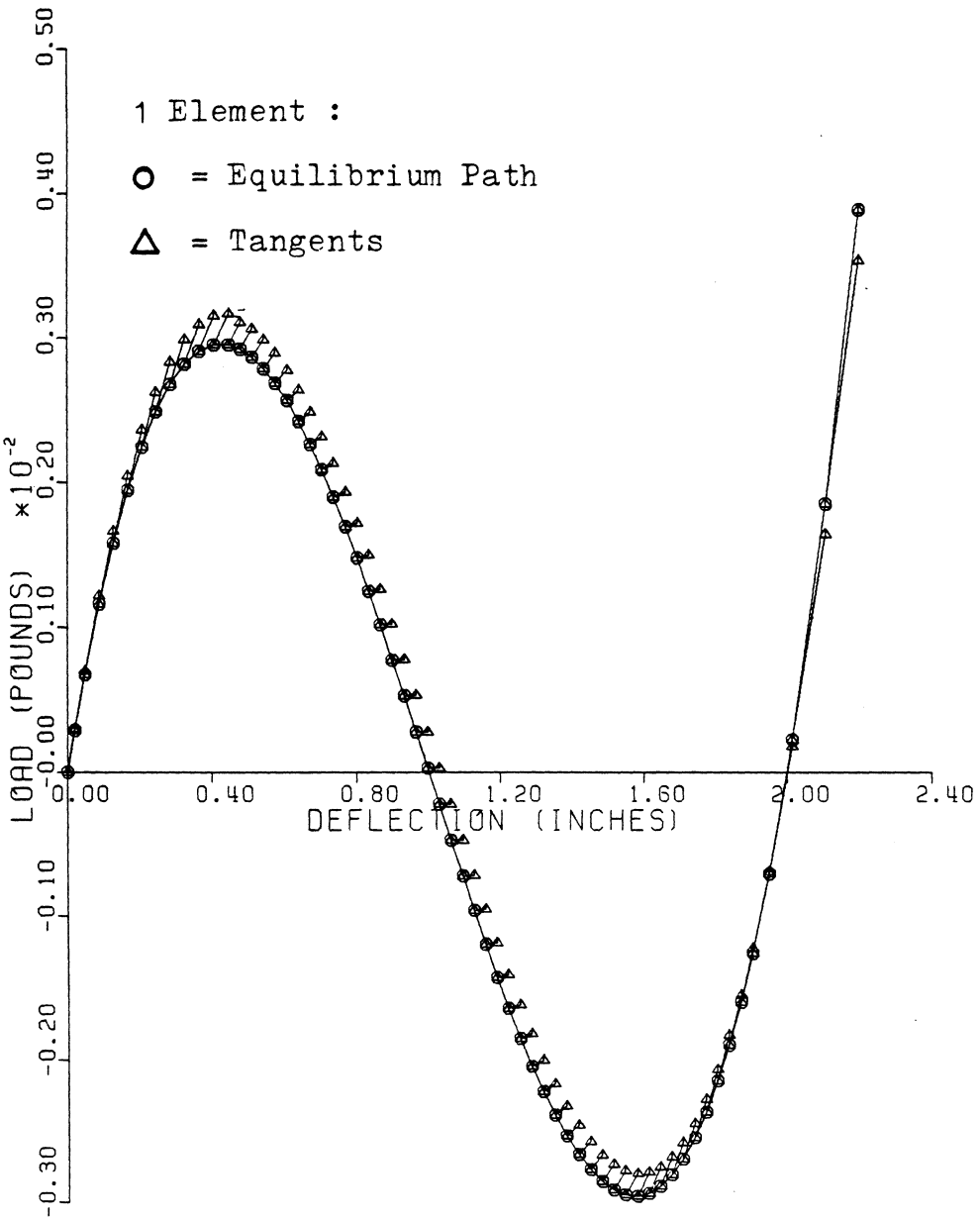


Figure 5.13 : Finite Element Model,

K_0 Only

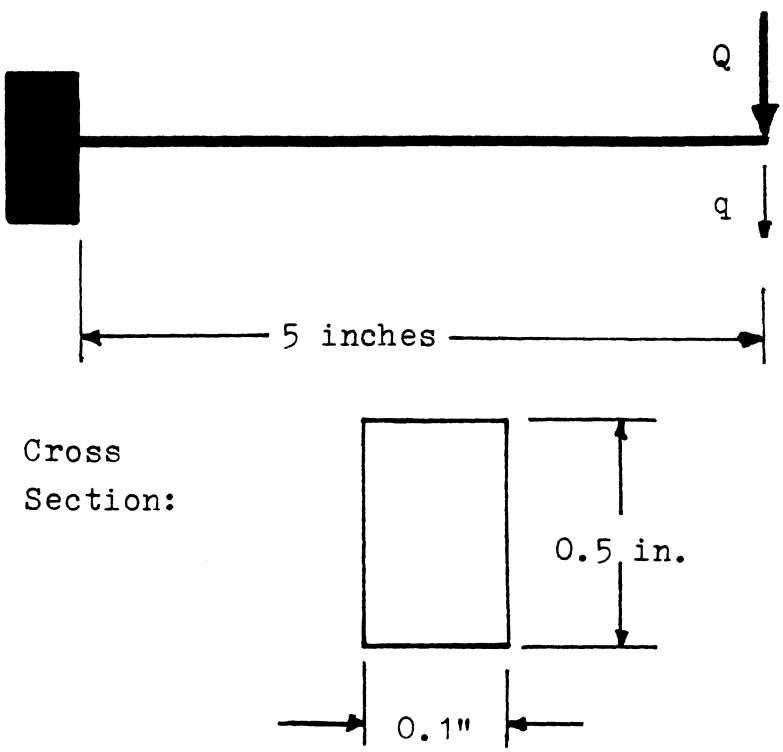
the tangent stiffness does not indicate instability when it should, the model could mistakenly be accepted as being correct. Such an error could be disastrous if a bifurcation point failed to be detected. A poor tangent stiffness may indicate instability following a limit point, but may fail to indicate instability following a bifurcation point. Note that a model derived using a convected coordinate system may contain a correct secant stiffness matrix and thus give a correct equilibrium path, but it may contain an incorrect representation of the tangent stiffness matrix. The initial stress stiffness matrix for the beam-column model arises from the incrementation of the local-to-global transformation relation, and the incremental form is used only in the development of the tangent stiffness (not in the development of the secant stiffness). The effect on the tangent stiffness of neglecting the incrementation is shown in figures 5.5 and 5.11 in which only the conventional tangent stiffness matrix, K_0, is included in the beam-column model. A finite element model derived using convected coordinates and neglecting the incrementation of the transformation would produce the same type of poor representation of the tangent stiffness as is given by the beam-column model.

5.4 Test Problem 3

The third test problem is the tip loaded cantilever investigated by Tang (35). The configuration, initial load increment, and geometric and material properties are shown in figure 5.14.

It is commonly known that as a structure is subdivided into more and more elements, the behavior of the finite element model approaches the behavior of the actual structure. The finite element model approximates a continuum using a finite number of degrees of freedom. Thus, as the number of degrees of freedom contained in the model increases, the behavior of the model approaches the behavior of the structure. A model having a coarse mesh is characteristically stiffer than one with a finer mesh (i.e. less displacement at a given load level).

In figure 5.15, every fifteenth point used to define each curve is marked on that curve. The beam-column model gives similar results for two and five elements. Although the finite element model gives satisfactory results for five elements, the response predicted is clearly stiffer than the response predicted by two beam-column elements. Thus, two beam column elements predict a response that is closer to the continuum response than that predicted by five finite elements.



$A = 0.05 \text{ inch}^2$

$E = 30 * 10^6 \text{ psi.}$

$I = 0.0010416 \text{ in.}^4$

$\Delta\lambda^0 = 1000 \text{ pounds}$

Figure 5.14 : Test Problem 3

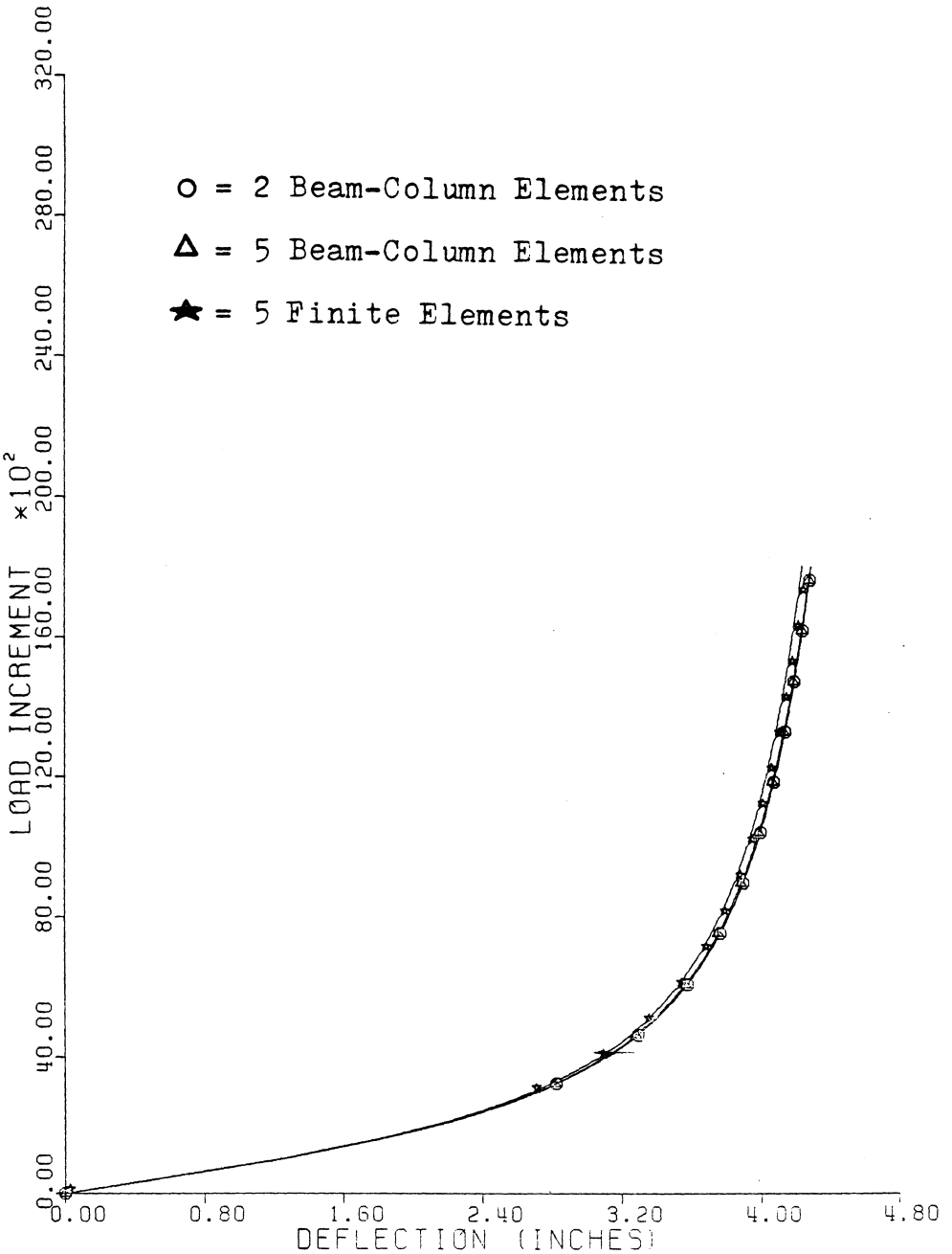


Figure 5.15 : Results for Problem 3

5.5 Test Problem 4

The fourth test problem is a simply supported beam studied in reference 33. The configuration, initial load increment, and geometric and material properties are shown in figure 5.16.

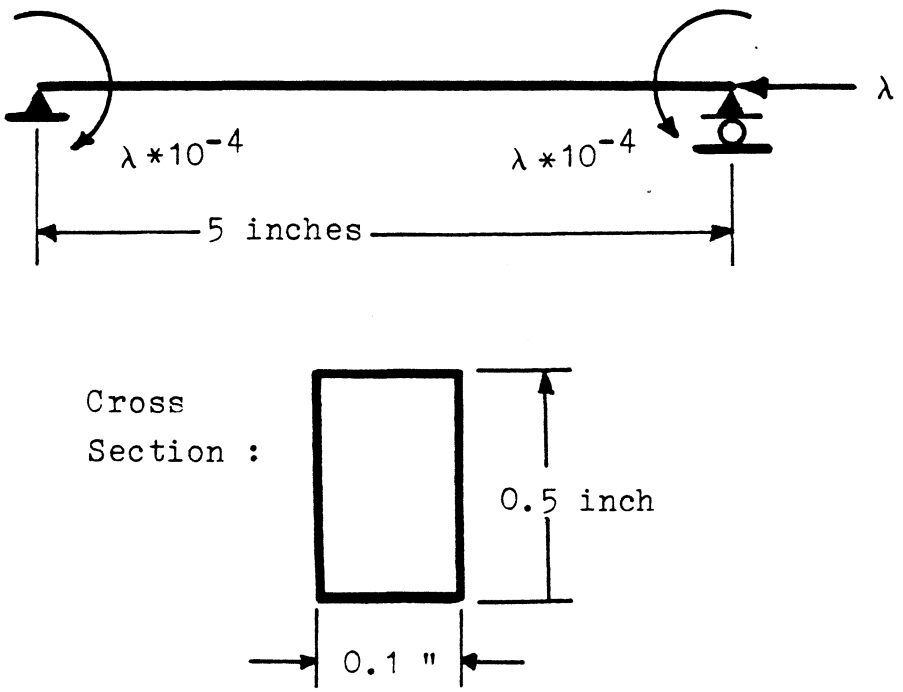
The Euler buckling load for the beam-column is given as

$$\pi^2 \, E \, I \, / \, L^2 = 12{,}337. \text{ lbs.} \qquad (5.1)$$

(33). As is shown in figure 5.17, the beam-column model buckles just below the Euler buckling load for one element. One element is much too stiff to accurately represent the behavior of the structure using the finite element model. Five elements give a closer representation of the actual behavior for the finite element model, but this representation is a stiffer and poorer representation than one beam-column element. Thus, the beam-column model gives a better prediction of the actual behavior using less elements.

5.6 Test Problem 5

The fifth test problem is the circular arch investigated by Schreyer and Masur (31), Sharifi and Popov (32), and in reference 18. The configuration, initial load increment, and material and geometric properties are shown in figure 5.18. The



$A = 0.05$ inch

$E = 30 * 10^6$ psi.

$I = 0.0010416$ in.^{4}

$\Delta\lambda^0 = 1000$ pounds

Figure 5.16 : Test Problem 4

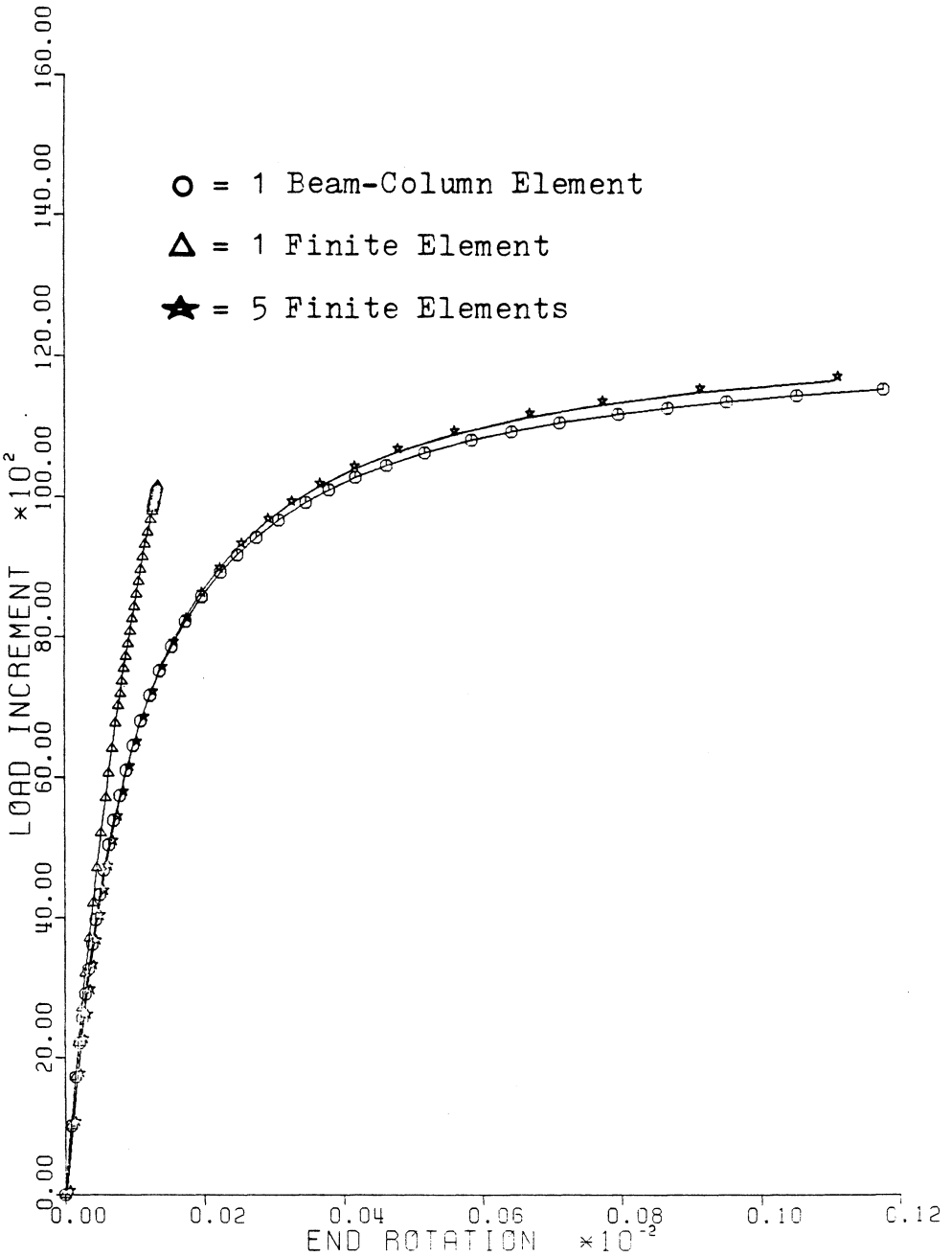
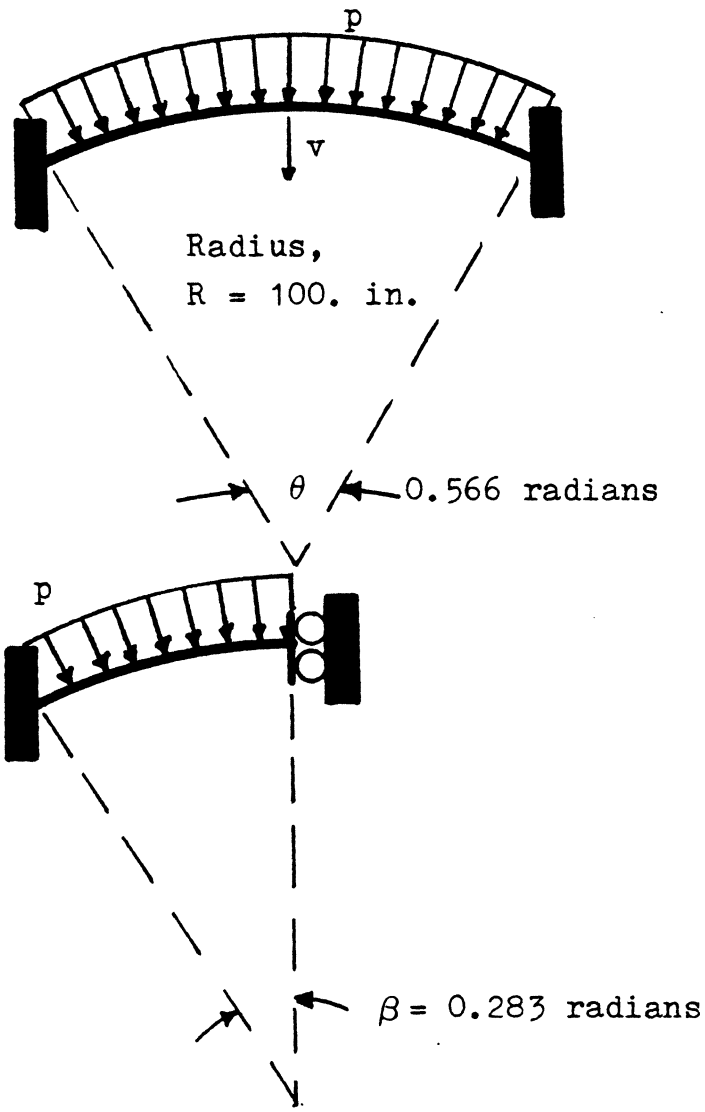


Figure 5.17 : Test Problem 4 Results

nondimensional pressure, $\bar{p}$, is plotted against the displacement ratio in figure 5.19. The displacement ratio is the vertical displacemnt at the center of the arch, v, divided by the radius of the arch, R. Apparently there is a misprint in reference 18 in the equation used to nondimensionalize the pressure since $\beta = \theta/2$. This test problem was tested using the finite element model only. The fixed end force equations for the beam-column model are presented by Roark and Young (30), but the implementation of member actions is untested for the beam-column model.

Because the fixed end forces for the beam-column model depend on the axial forces in the members, the fixed end forces must be updated whenever the configuration of the structure is updated. The computer program used for this study is designed, therefore, to update the fixed end forces each time the configuration is updated for both models. Thus, a member action that is normal to the member remains normal as the structure deforms. The element actions are implemented as follower forces (18) in this way.

The plot obtained by the finite element model for the pressure-loaded arch is similar to those given by Schreyer and Masur, and Sharifi and Popov.



h = depth of cross section = 2 inches

A = 2. inches^2

E = 10,000. Ksi.

I = $0.66\overline{6}$

$\Delta\lambda^0$ = 0.01643

$\bar{p} = (12\ \beta^2\ R^3 / \pi^2\ E\ h^3)p$

Figure 5.18 : Test Problem 5

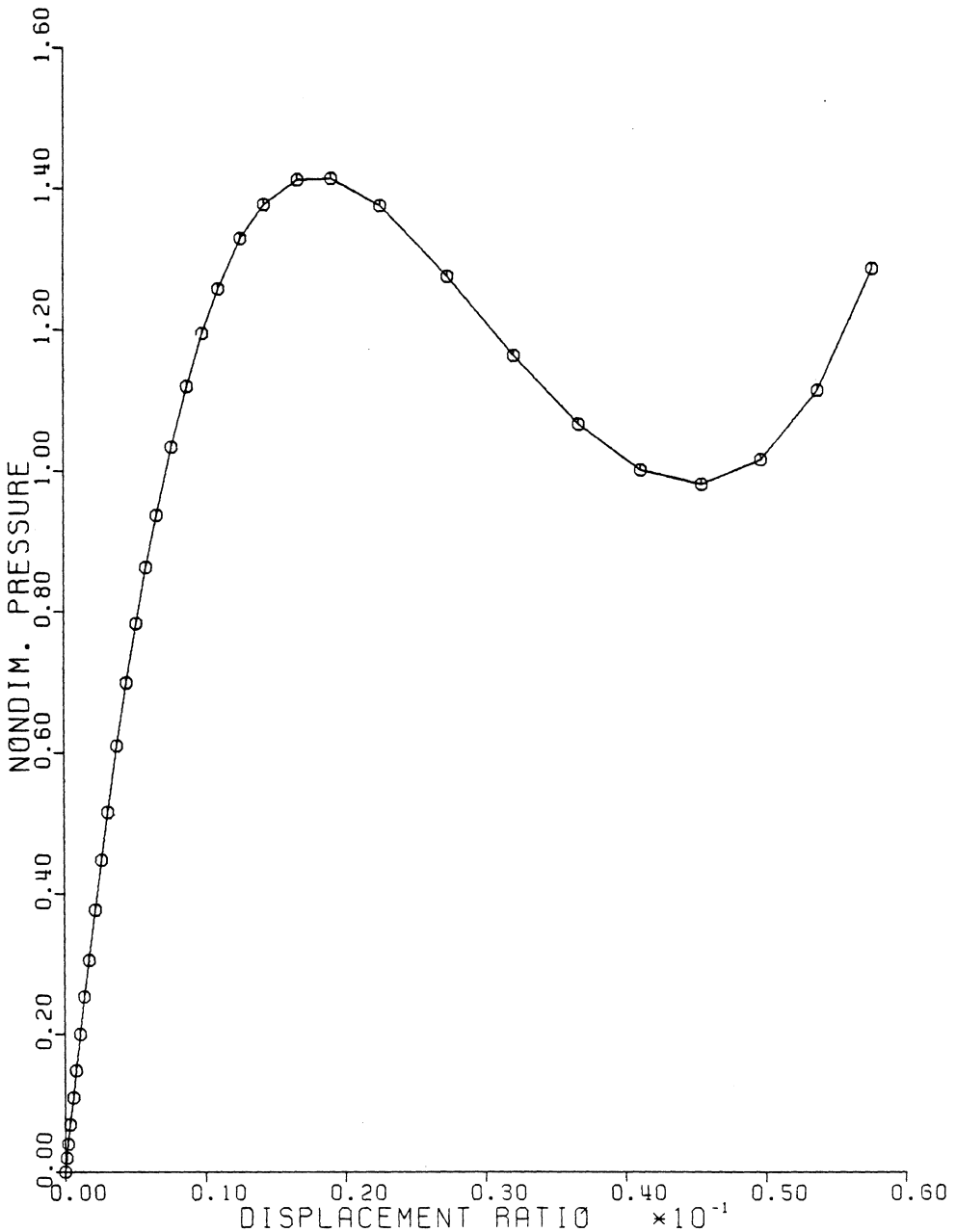


Figure 5.19 : Test Problem 5 Results

CHAPTER 6
CONCLUSION

6.1 Conclusions

Two models for geometrically nonlinear finite element analysis are compared by this study. The model derived using conventional beam-column theory consistently produces more accurate results for frame structures using a coarser mesh than that required by the model developed using standard finite element techniques. Both models give equally good results for the truss problem because both contain the solution of the homogeneous differential equation for axial behavior. The beam-column model is expected to give better results with a coarser mesh for frame problems because the model represents the solution to the differential equation for a beam-column for each element. The beam-column model is obviously more efficient in this respect.

The tangent stiffness matrices are accurate predictors for both models when both the conventional and the initial stress stiffness contributions are included. When the initial stress stiffness contribution to the tangent stiffness is neglected, both models are poor predictors of the structural behavior. Thus, the

initial stress stiffness matrix must be included in both models for accurate representation of large displacement problems.

The two models are also compared on the basis of the difficulty required to implement each one. A Newton method iteration is needed to find the axial force in the beam-column model. Also, the stability coefficients must be computed, and which equations are used to determine them depends on whether the axial force is compressive, tensile, or zero. If element actions are to be included, more difficulty is involved in the implementation of element actions in the beam-column model than in the finite element model. The fixed end force equations for lateral loading which are compatible with the beam-column theory involve the axial force in each member, and they must be updated each time the configuration is updated. Which equations are used also depends on whether the axial force is compressive, tensile, or zero. These complications are not involved in the finite element model.

A third area in which the models differ is their ability to be extended to material nonlinearity. Since the beam-column model is formulated with constant material properties for each element, it cannot be readily extended to material nonlinearity. The finite element

model, on the other hand, is formulated for each point and integrated over the volume of the element. Thus, the finite element model can be extended to include material nonlinearity.

Other useful observations that arise from this study include the plotting of tangents. The tangents to the equilibrium path as they are determined by the tangent stiffness matrix are plotted by connecting the current equilibrium point to the first estimate of the next equilibrium point by a line. The length of this line is Δs in the modified Riks/Wempner method. A plot of tangents is a useful tool in the determination of the accuracy of the representation of the tangent stiffness contained in a model. Note that care should be taken when a geometric configuration is plotted on sets of axes that have different scales.

This study demonstrates that when the full representation of the tangent stiffness is not used, structural instability may not be detected by examination of the inertia of the tangent stiffness matrix. Limit points can be obtained from the shape of the equilibrium path, but bifurcation points cannot be detected this way. Thus, in order to be assured of detecting instability, the full representation of the tangent stiffness matrix

should be used when the behavior of the structure is not known beforehand.

The importance of including the incrementation of the local-to-global coordinate transformation matrix in the formulation of the tangent stiffness matrix using convected coordinates has been demonstrated with the beam-column model. Since the initial stress stiffness matrix arises from this incrementation for the beam-column model (12), the contribution of this term is represented in the plots of the tangent stiffness which neglect it. The test problems and plots show that instability may not be detected if the incrementation is neglected.

6.2 Recommendations for Future Study

The following are possible future extensions of this study :

1. Include and study element actions for the beam-column model.
2. Use the element action portion of the program to model element imperfections. Some equations for fixed end forces due to element imperfections are given by Roark and Young (30).
3. A comparison of the two models is currently being performed for space frames by Jau (42).

4. Implement both models in an updated Lagrangian formulation. Such a formulation may exhibit different effects of neglecting contributions to the tangent stiffness matrix.
5. The finite element model can be extended to include both material and geometric nonlinearity.
6. The finite element model can also be extended to include the initial displacement stiffness matrix, K_L, and the effect of using different combinations of the contributions to the tangent stiffness can be studied for both the B- and the N-notations.
7. The computer program developed for this study contains four tests for convergence to an equilibrium point. The effects of different convergence tests and different convergence parameters on an analysis can be studied.
8. The effectiveness of an automatic node renumbering scheme to obtain a narrow band could be studied in the context of a variable band solver.
9. A scheme for automated mesh generation or refinement could be included in the program.

APPENDIX A

REFERENCES

1. Bathe, K.-J., Finite Element Procedures in Engineering Analysis, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1982.

2. Bathe, K.-J. and Cimento, A.P., "Some Practical Procedures for the Solution of Nonlinear Finite Element Equations," Journal of Computer Methods in Applied Mechanics and Engineering, Vol. 27, 1980, pp. 59-85.

3. Bazant, Z.P., "Discussion of 'Incremental Finite Element Matrices'",(reference 29), Journal of the Structural Division, ASCE, September, 1974, p. 1976.

4. Belytschko, T. and Hsieh, B.J., "Nonlinear Transient Finite Element Analysis with Convected Coordinates", International Journal for Numerical Methods in Engineering, Vol. 7, 1973, pp. 255-271.

5. Cook, R.D., Concepts and Applications of Finite Element Analysis, second edition, John Wiley and Sons, 1981.

6. Hansen, M.C., A Comparison of Two Solution Techniques for Snap-Through Instability of Trusses, M.S. Thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, March, 1981.

7. Holzer, S. M., Matrix Structural Analysis With Program Development, 1983, to be published.

8. Holzer, S.M., "Degree of Stability of Equilibrium", Journal of Structural Mechanics, Vol. 3, 1974, pp. 61 - 75.

9. Holzer, S.M., "Lecture Notes on Finite Element Analysis of Structures", Virginia Polytechnic Institute and State University, Blacksburg, Virginia, Winter, 1983.

10. Holzer, S.M., "Lecture Notes on Dynamics of Structures", Virginia Polytechnic Institute and State University, Blacksburg, Virginia, Spring, 1983.

11. Holzer, S.M., "Notes on Geometrically Nonlinear Analysis", Virginia Polytechnic Institute and State University, Blacksburg, Virginia, June, 1983.

12. Holzer, S.M., "Notes on the Tangent Stiffness Matrix", Virginia Polytechnic Institute and State University, Blacksburg, Virginia, October, 1983.

13. Holzer, S.M., Watson, L.T., and Vu, P.D., "Stability Analysis of Lamella Domes", Proceedings of the ASCE Symposium on Long Span Roof Structures, St. Louis, Missouri, October, 1981, pp. 179 - 209.

14. Huseyin, K., Nonlinear Theory of Elastic Stability, Noordhoff International Publishing, 1975.

15. Jagannathan, D.S., Epstein, H.I., and Christiano, P., "Fictitious Strains due to Rigid Body Rotation", Journal of the Structural Division, ASCE, Vol. 101, No. ST11, November, 1975.

16. Jagannathan, D.S., Epstein, H.I., and Christiano, P., "Nonlinear Analysis of Reticulated Space Trusses", Journal of the Structural Division, ASCE, Vol. 101, No. ST12, December, 1975.

17. Jagannathan, D.S., Epstein, H.I., and Christiano, P., "Snap-Through Buckling of Reticulated Shells", Proc. Instn. Civ. Eng., Part 2, December, 1975, pp. 727 - 742.

18. Katzenberger, G.S., "Geometrically Nonlinear Analysis of Pressure-Loaded Arches, Rings, and Frames Using a Follower Force Algorithm", M.S. Thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, August, 1983.

19. Lamma, E.E., "Tracing the Fundamental and Secondary Equilibrium Paths of Geometrically Nonlinear Space Trusses Using the Modified Riks/Wempner Method", M.S. Thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, August, 1982.

20. Langhaar, H.L., Energy Methods in Applied Mechanics, John Wiley and Sons, Inc., 1962.

21. Mallet, R.H., and Marcal, P.V., "Finite Element Analysis of Nonlinear Structures", Journal of the Structural Division, Proceedings of the American Society of Civil Engineers, Vol. 94, No. ST9, September, 1968, pp. 2081 - 2105.

22. Mallet, R.H., and Marcal, P.V., "Closure of 'Finite Element Analysis of Nonlinear Structures'", (reference 21), Journal of the Structural Division, Proceedings of the American Society of Civil Engineers, No. ST1, January, 1970, p.132.

23. Mikkola, M.J., "Notes on Modelling of Geometrically Nonlinear Problems of Framed Structures", Virginia Polytechnic Institute and State University, Blacksburg, Virginia, August, 1981.

24. Oran, C.O., "Tangent Stiffness in Plane Frames", Journal of the Structural Division, ASCE, Vol. 99, No. ST6, June, 1973, pp. 973 - 983.

25. Oran, C.O., and Kassimali, A., "Large Deformations of Framed Structures Under Static and Dynamic Loads", Computers and Structures, Vol. 6, pp. 539 - 547, Pergamon Press, 1976.

26. Ovunc, B., and Oden, J.T., "Discussion of 'Finite Element Analysis of Nonlinear Structures'", (reference 21), Journal of the Structural Division, ASCE, No. ST6, June, 1969, pp. 1379 - 1381.

27. Papadrakakis, M., "Post-Buckling Analysis of Spatial Structures by Vector Iteration Methods", Computers and Structures, Vol. 14, No. 5 - 6, 1981, pp. 393 - 402.

28. Przemieniecki, J.S., Theory of Matrix Structural Analysis, McGraw-Hill, 1968.

29. Rajasekaran, S., and Murray, D.W., "Incremental Finite Element Matrices", Journal of the Structural Division, ASCE, Vol. 99, No. ST12, December, 1973, pp. 2423 - 2437.

30. Roark, R.J., and Young, W.C., Formulas for Stress and Strain, fifth edition, McGraw-Hill, Inc., 1975.

31. Schreyer, H.L., and Masur, E.F., "Buckling of Shallow Arches", Journal of the Engineering Mechanics Division, ASCE, Vol. 89, No. EM3, Proc. Paper 3562, June, 1963, pp. 197 - 199.

32. Sharifi, P., and Popov, E.P., "Nonlinear Buckling Analysis of Sandwich Arches", Journal of the Engineering Mechanics Division, ASCE, Vol. 97, No. EM5, October, 1971, pp. 1397 - 1412.

33. Soltani, M., "Stability Analysis of Geometrically Nonlinear Plane Frames Via the Finite Element Technique", M.S. Thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, August, 1982.

34. Somers, A.E., Jr., "Lecture Notes on Computer Aided Structural Design", Virginia Polytechnic Institute and State University, Blacksburg, Virginia, Spring, 1983.

35. Tang, S.C., Yeung, K.S., and Chon, C.T., "On the Tangent Stiffness Matrix in a Convected Coordinate System", Computers and Structures, Vol. 12, Pergamon Press Ltd., 1980, pp. 849 - 856.

36. Timoshenko, S.P., and Gere, J.M., Theory of Elastic Stability, McGraw-Hill, Inc., 1961.

37. Vu, P.D., "Tracing Nonlinear Equilibrium Paths by the Modified Riks/Wempner Method", M.S. Thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, October, 1981.

38. Williams, F.W., "An Approach to the Non-Linear Behavior of the Members of a Rigid Jointed Plane Framework with Finite Deflections", Quarterly Journal of Mechanics and Applied Mathematics, Vol. XVII, Part 4, 1964, pp. 451 - 469.

39. Wood, R.D., and Schrefler, B., "Geometrically Nonlinear Analysis - A Correlation of Finite Element Notations", International Journal for Numerical Methods in Engineering, Vol. 12, 1978, pp. 635 - 642.

40. Wood, R.D., and Zienkiewicz, O.C., "Geometrically Nonlinear Finite Element Analysis of Beams, Frames, Arches, and Axisymmetric Shells", Computers and Structures, Vol. 7, Pergamon Press, 1977, pp. 725 - 735.

41. Zienkiewicz, O.C., The Finite Element Method, third, expanded and revised edition, McGraw-Hill Book Company Ltd., 1977.

42. Jau, Jih Jih, Dissertation to be submitted to the Faculty of Virginia Polytechnic Institute and State University, Blacksburg, Virginia.

43. Nassi, I. and Schneiderman, B., "Flow Chart Techniques for Structured Programming,", ACM SIGPLAN Notices, Vol. 8, No. 8, August, 1973 pp. 12 - 26.

APPENDIX B

NOTATION

The following is a list of notation used. Additional notation is given in the program listing (appendix N).

A	Area of the cross section of a member
B	for the beam-column model: local (convected) to global force transformation matrix for the finite element model: strain-displacement relation matrix
B_1, B_2	Bowing functions
c_b	length correction factor due to bowing
c_1, c_2	Member direction cosines (see reference 7 for a more complete explanation)
d	local element displacements (six per element)
e_1, e_2, e_3	local displacements in convected coordinate system
E	Modulus of elasticity
f	local element force (six per element)
F	generalized internal element force vector due to deformations
h	depth of member cross section
I	Moment of inertia
J	Jacobian matrix (9)

K_T system tangent stiffness matrix

K_0 conventional stiffness matrix for linear analysis

K_σ initial stress stiffness matrix

K_L initial displacement stiffness matrix

k member stiffness matrix

L length of member

M moment

M_1, M_2 moment at a- and b-end of member respectively

N Hermite interpolation functions (5, 7, 9)

p applied pressure

$\bar{p}$ nondimensional pressure

P applied force

q generalized displacement

Q axial load in member, generalized load, applied load

$\bar{Q}$ load distribution vector

Q_E Euler buckling load

R residual force vector, applied loads - element forces; radius of circular arch in test problem 5

s_1, s_2 stability functions

u axial deformation (convected coordinates); displacement of a point in the x-direction

U internal strain energy

v	displacement of a point in the y-direction
W	total work
W_e	external work
δW, δW_e, δU	first variation of W, W_e, and U respectively
$\bar{\pi}$	total potential energy
$\delta\bar{\pi}$, $\delta^2\bar{\pi}$	first and second variations of the total potential energy respectively
ϵ	strain
$\epsilon(x,y)$	strain at point (x,y)
λ	load incrementing factor; coordinate transformation matrix
$\Delta\lambda$	change in load increment
ξ	normalized cartesian coordinate (7, 9)
σ	stress
θ_1, θ_2	local element rotations
$X_{,y}$	partial derivative of X with respect to y
$X_{,yy}$	second partial derivative of X with respect to y

APPENDIX C

DERIVATION OF SQUARE TERM IN GREEN'S STRAIN

From figure C.1,

$$\Delta x^* = (\Delta x^2 + \Delta v^2)^{\frac{1}{2}} \tag{C.1}$$

The fundamental definition of strain is

$$\epsilon = \lim_{\Delta x \to 0} \frac{\Delta x^* - \Delta x}{\Delta x} \tag{C.2}$$

From equation C.1,

$$\frac{\Delta x^* - \Delta x}{\Delta x} = \left[1 + \left(\frac{\Delta v}{\Delta x}\right)^2 \right]^{\frac{1}{2}} - 1 \tag{C.3}$$

A binomial expansion gives

$$\left[1 + \left(\frac{\Delta v}{\Delta x}\right)^2 \right]^{\frac{1}{2}} = 1 + \tfrac{1}{2} \left(\frac{\Delta v}{\Delta x}\right)^2 \tag{C.4}$$

From C.2, C.3, and C.4,

$$\epsilon = \lim_{\Delta x \to 0} 1 + \tfrac{1}{2}\left(\frac{\Delta v}{\Delta x}\right)^2 - 1 = \tfrac{1}{2}\left(\frac{dv}{dx} \right)^2 \tag{C.5}$$

Equation C.5 is the square term in Green's strain.

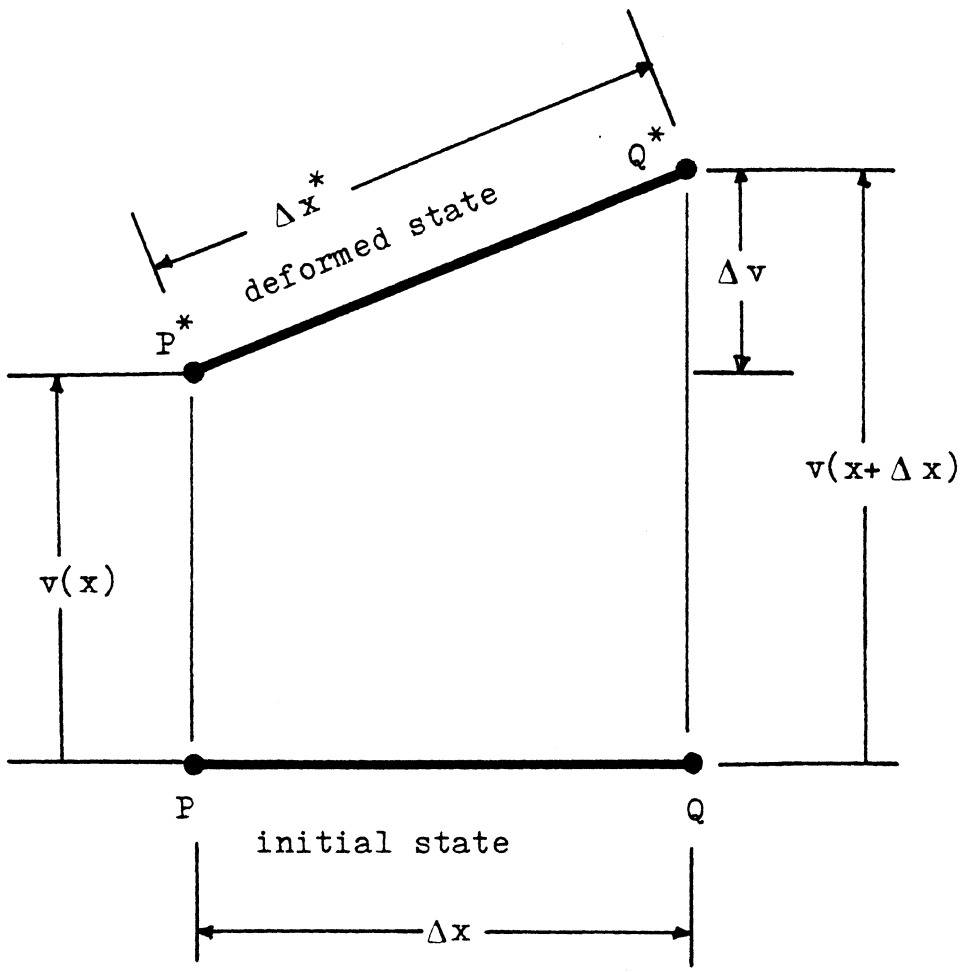


Figure C.1 : Square Term

APPENDIX D

DEVELOPMENT OF GREEN'S STRAIN AT A POINT

Green's strain parallel to the x-axis in a plane is

$$\epsilon_x = \frac{du}{dx} + \frac{1}{2}\left(\frac{du}{dx}\right)^2 + \left(\frac{dv}{dx}\right)^2 \qquad (D.1)$$

Neglecting the first square term because it is small in relation to du/dx for small rotations (see section 2.2 and reference 16) gives :

$$\epsilon_x = \frac{du}{dx} + \frac{1}{2}\left(\frac{dv}{dx}\right)^2 \qquad (D.2)$$

From figure D.1,

$$\epsilon(x,y) = \frac{d\hat{u}}{dx} + \frac{1}{2}\left(\frac{d\hat{v}}{dx}\right)^2 \qquad (D.3)$$

If dv/dx in figure D.1 is a small angle (i.e. $(dv/dx)^2 \cong 0$ relative to unity), plane sections normal to the neutral axis remain normal, and the length of $\overline{PQ}$ is the same as the length of $\overline{P^*Q^*}$ in figure D.1,

$$\hat{v} = v \qquad (D.4)$$

From figure D.1,

$$\hat{u} = u - y\frac{dv}{dx} \qquad (D.5)$$

From equations D.3, D.4, and D.5,

$$\epsilon(x,y) = \frac{du}{dx} - y\frac{d^2v}{dx^2} + \frac{1}{2}\left(\frac{dv}{dx}\right)^2 \qquad (D.6)$$

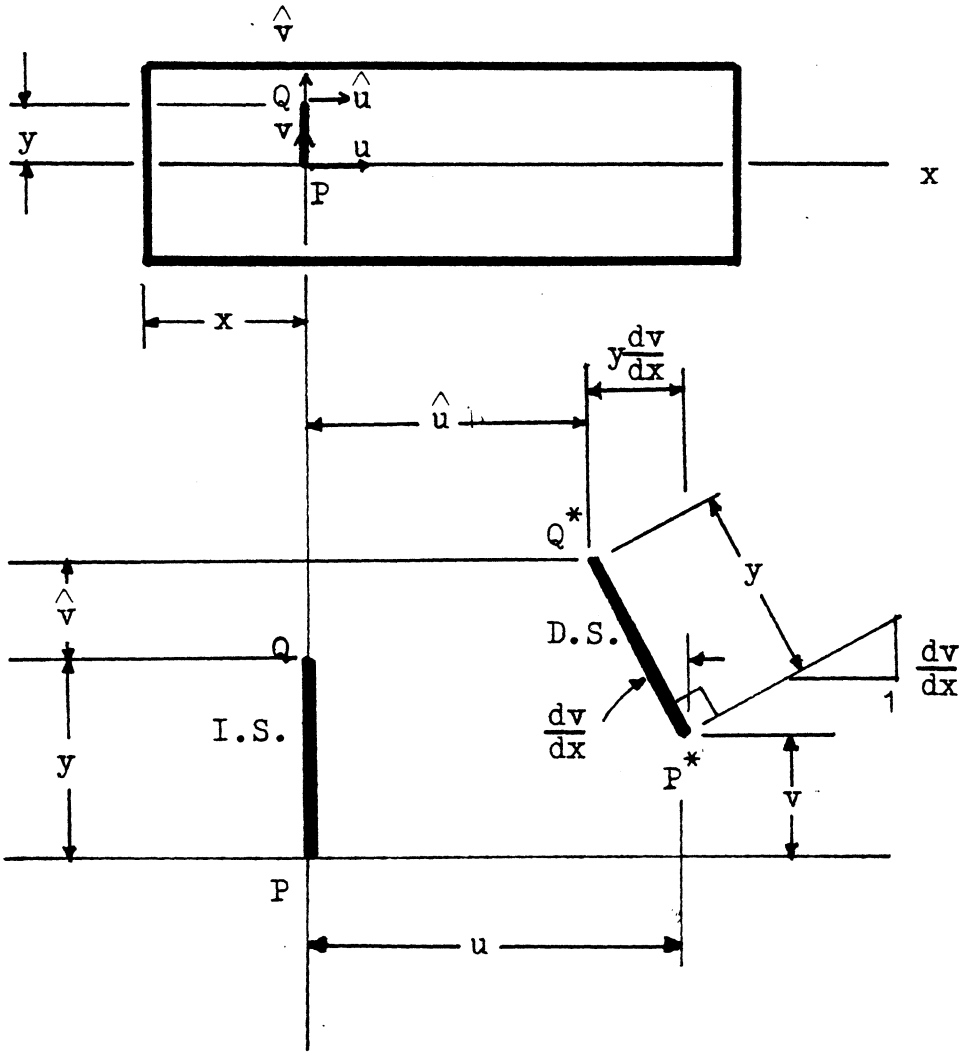
which is Green's strain at a point, where

$$\frac{d\hat{u}}{dx} = \frac{du}{dx} - y\frac{d^2v}{dx^2} \tag{D.7}$$

and

$$\frac{1}{2}\left(\frac{dv}{dx}\right)^2$$

is a nonlinear coupling term.



I.S. = initial state

D.S. = deformed state

Figure D.1 : Green's Strain at a Point

APPENDIX E

ORTHOGONALITY CONDITION FOR THE MODIFIED RIKS/WEMPNER METHOD

The orthogonality condition in the modified Riks/Wempner method assures that iteration proceeds along a line that is normal to the tangent to the equilibrium at the last equilibrium point. The notation used here (and in reference 13) corresponds to the notation of the computer program in appendix N as follows :

$$
\begin{aligned}
QI &= \lambda \\
DQI &= \Delta\lambda \\
DDO1 &= \Delta q^{OI} \\
DD1 &= \Delta q^{I} \\
DD2 &= \Delta q^{II} \\
DD &= \Delta q \\
D &= q
\end{aligned}
\qquad \text{(E.1)}
$$

Representing vectors in the (λ , q) plane by Δr, see figure E.1 for the modified Riks/Wempner iteration from point k to point k+1 along the normal to Δr^{O} moving from equilibrium point "O" to point "N".

From figure E.1:

$$
\Delta r^{O} = \begin{bmatrix} \Delta q^{O} \\ \Delta \lambda^{O} \end{bmatrix} \qquad \text{(E.2)}
$$

and

$$\Delta r^k = \begin{bmatrix} \Delta q^k \\ \Delta \lambda^k \end{bmatrix} = \Delta \lambda^k \begin{bmatrix} \Delta q^{kI} \\ 1 \end{bmatrix} + \begin{bmatrix} \Delta q^{kII} \\ 0 \end{bmatrix} \quad (E.3)$$

Also,

$$\Delta r^k = \Delta \lambda^k \Delta r^{kI} + \Delta r^{kII} \quad (E.4)$$

For orthogonality,

$$\Delta r^0 \cdot \Delta r^k = 0 \quad (E.5)$$

or

$$\begin{bmatrix} \Delta q^0 \\ \Delta \lambda^0 \end{bmatrix} \cdot \Delta \lambda^k \begin{bmatrix} \Delta q^{kI} \\ 1 \end{bmatrix} + \begin{bmatrix} \Delta q^0 \\ \Delta \lambda^0 \end{bmatrix} \cdot \begin{bmatrix} \Delta q^{kII} \\ 0 \end{bmatrix} = 0 \quad (E.6)$$

which gives

$$\Delta \lambda^k (\Delta q^0 \cdot \Delta q^{kI} + \Delta \lambda^0) + (\Delta q^0 \cdot \Delta q^{kII}) = 0 \quad (E.7)$$

Rearranging equation E.7 gives

$$\Delta \lambda^k = - \frac{(\Delta q^0 \cdot \Delta q^{kII})}{(\Delta q^0 \cdot \Delta q^{kI} + \Delta \lambda^0)} \quad (E.8)$$

which is the orthogonality condition corresponding to equation 33 in reference 13.

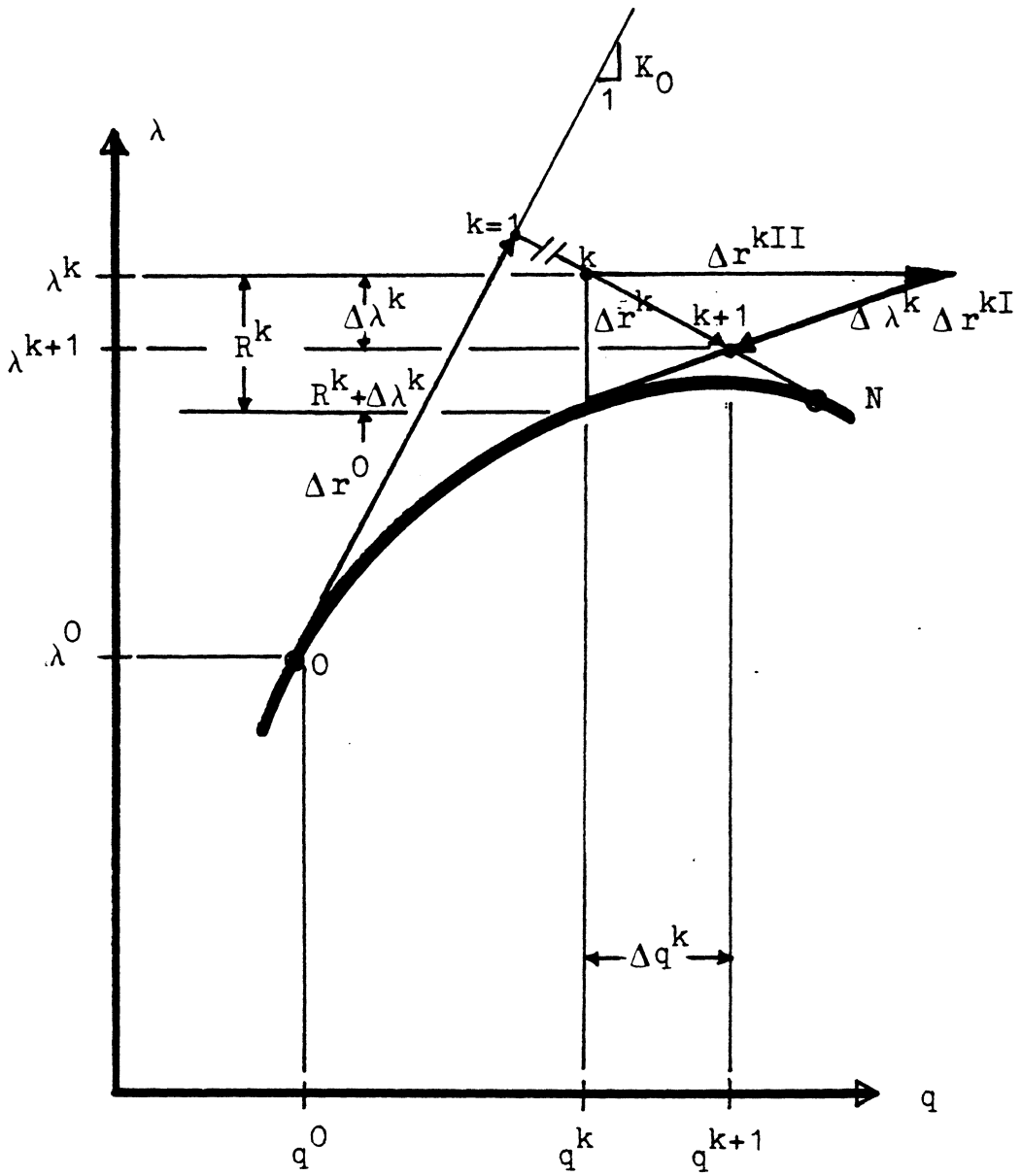


Figure E.1 : Iteration Along the Normal

APPENDIX F

STIFFNESS MATRIX AND STABILITY

The Law of Minimum Potential Energy states that the potential energy of a system must be at a relative minimum for stability (20, 41). Zienkiewicz (41) gives the variation of the total potential energy for a system:

$$\delta \overline{\pi} = \delta D^T F \tag{F.1}$$

For equation F.1 to be at a relative minimum, the following must be true

$$\delta \overline{\pi} = 0 \tag{F.2}$$

$$\delta^2 \overline{\pi} > 0 \tag{F.3}$$

By elementary variational principles and equation F.1,

$$\delta^2 \overline{\pi} = \delta D^T \delta F \tag{F.4}$$

Recalling the definition of the tangent stiffness,

$$\Delta F = K_T \Delta D \tag{F.5}$$

Thus, since variational principles follow the principles of elementary calculus,

$$\delta D^T K_T \delta D > 0 \tag{F.6}$$

for stability. It follows directly from equation F.6 that K_T must be positive definite for stability.

APPENDIX G

DEVELOPMENT OF STABILITY FUNCTIONS AND INTERNAL FORCE EQUATIONS FOR THE BEAM-COLUMN MODEL

The differential equation for the behavior of a beam-column (figure G.1) is developed as follows (36). Summing vertical forces in figure G.2:

$$- V + q\,dx + V+dV = 0 \qquad (G.1)$$

or

$$q = - \frac{dV}{dx} \qquad (G.2)$$

Summing moments at point n gives :

$$M + q\,dx\,\frac{dx}{2} + (V+dV)dx - (M+dM) + P\,\frac{dy}{dx}\,dx = 0 \qquad (G.3)$$

Neglecting second order terms,

$$V\,dx - dM + P\,dy = 0 \qquad (G.4)$$

or

$$V = \frac{dM}{dx} - P\,\frac{dy}{dx} \qquad (G.5)$$

From beam theory, neglecting shear deformations and shortening of the beam axis, the expression for the curvature of the beam axis gives

$$M = - E\,I\,\frac{d^2y}{dx^2} \qquad (G.6)$$

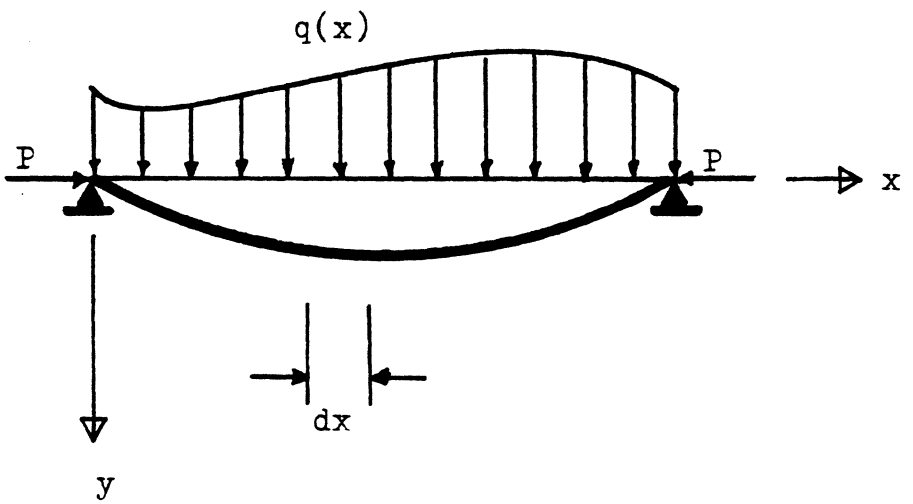


Figure G.1 : Beam-Column

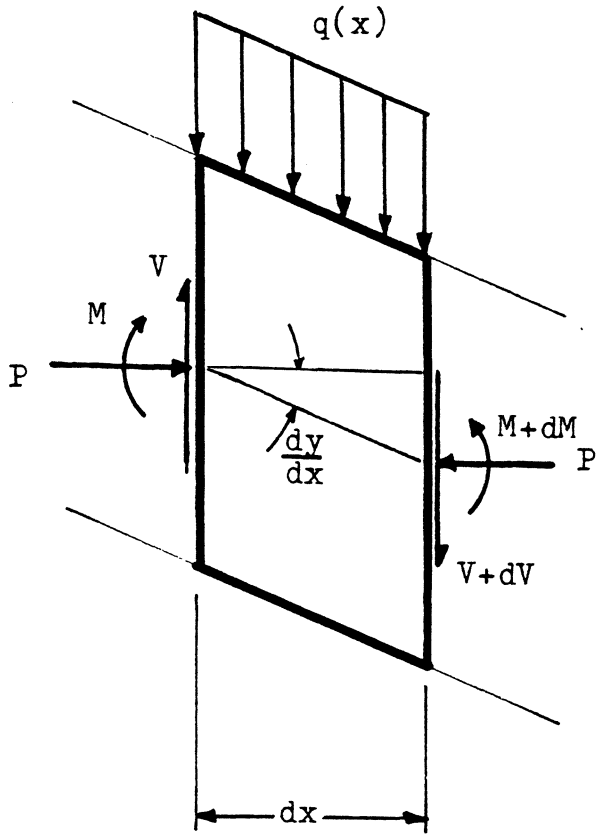


Figure G.2 : Differential Element

Combining equations G.5 and G.6 for a beam with constant cross section,

$$E\,I\,\frac{d^3y}{dx^3} + P\,\frac{dy}{dx} = -\,V \qquad (G.7)$$

Using G.2,

$$E\,I\,\frac{d^4y}{dx^4} + P\,\frac{d^2y}{dx^2} = q \qquad (G.8)$$

which is the differential equation for a beam-column.

For a compressive axial force, the solution of the homogeneous equation is

$$y = A\,\sin kx + B\,\cos kx + Cx + D \qquad (G.9)$$

where

$$k = (\,P\,/\,EI\,)^{\frac{1}{2}} \qquad (G.10)$$

Applying the boundary conditions for the beam in figure G.3,

$$\begin{aligned} y(0) &= 0 \\ y(L) &= 0 \\ \frac{dy}{dx}(0) &= \theta_1 \\ \frac{dy}{dx}(L) &= \theta_2 \end{aligned} \qquad (G.11)$$

gives

$$A = \frac{-(\,\theta_1 + \theta_2)(s_1 + s_2) + \theta_1\,\phi^2}{\phi^2\,k} \qquad (G.12a)$$

$$B = \frac{\theta_1 s_1 + \theta_2 s_2}{k^2 L} \qquad (G.12b)$$

$$C = \theta_1 - A k \qquad (G.12c)$$

$$D = - B \qquad (G.12d)$$

where

$$\phi = k L \qquad (G.13)$$

$$s_1 = \frac{\phi(\sin\phi - \phi\cos\phi)}{(2-2\cos\phi - \phi\sin\phi)} \qquad (G.14)$$

$$s_2 = \frac{\phi(\phi - \sin\phi)}{(2-2\cos\phi - \phi\sin\phi)} \qquad (G.15)$$

and from equations G.6, G.9, and G.12,

$$M_1 = \frac{E I}{L} (s_1 \theta_1 + s_2 \theta_2) \qquad (G.16)$$

and

$$M_2 = \frac{E I}{L} (s_2 \theta_1 + s_1 \theta_2) \qquad (G.17)$$

To find the axial force, note that

$$\int_0^L \epsilon \, dx = \int_0^L \left(\frac{du}{dx} + \frac{1}{2}\left(\frac{dy}{dx}\right)^2 \right) dx \qquad (G.18)$$

and

$$\frac{dy}{dx} = A k \cos kx - B k \sin kx + C \qquad (G.20)$$

Combining equations G.18, G.19, and G.20, substituting integration constants, performing integration, and applying the limits gives

$$Q = E A \left(\frac{u}{L} - c_b \right) \qquad (G.21)$$

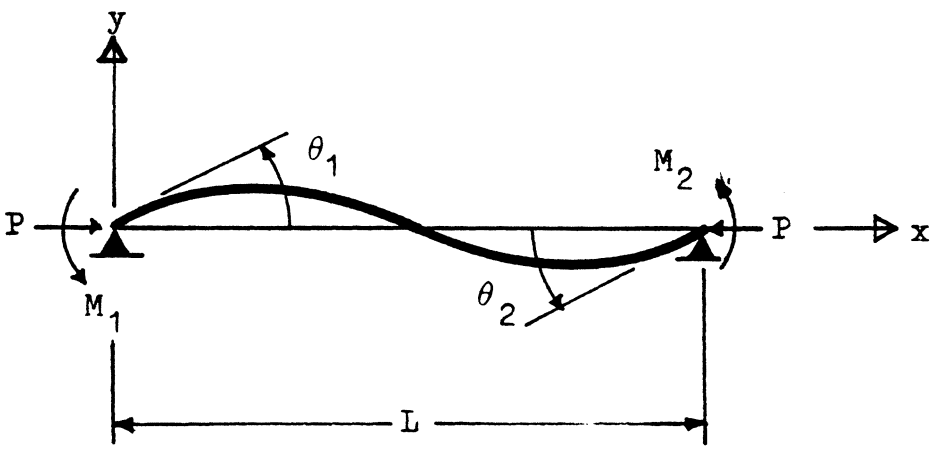


Figure G.3 : Boundary Conditions

where,

$$c_b = B_1 (\theta_1 + \theta_2)^2 + B_2 (\theta_1 + \theta_2)^2 \qquad (G.22)$$

$$B_1 = \frac{(s_1 + s_2)(s_2 - 2)}{8 \pi^2 q} \qquad (G.23)$$

$$B_2 = \frac{s_2}{8 (s_1 + s_2)} \qquad (G.24)$$

$$q = P / Q_E = Q / Q_E = QR \qquad (G.25)$$

$$Q_E = \pi^2 E I / L^2 \qquad (G.26)$$

For a tensile axial force,

$$y = A \sinh kx + B \cosh kx + Cx + D \qquad (G.27)$$

for

$$k = (-Q / EI)^{\frac{1}{2}} \qquad (G.28)$$

$$\psi = -QL^2/EI = kL = - \pi^2 q \qquad (G.29)$$

Applying the boundary conditions gives (23):

$$A = \frac{(s_1 + s_2)(\psi_1 + \psi_2) + (kL)^2 \psi}{k^3 L^2}$$

$$B = - (s_1 \psi_1 + s_2 \psi_2) / k^2 L \qquad (G.30)$$

$$C = - (s_1 + s_2)(\psi_1 + \psi_2) / (kL)^2$$

$$D = - B$$

where

$$s_1 = \frac{\psi(\psi \cosh\psi - \sinh\psi)}{(2-2\cosh\psi + \psi\sinh\psi)} \qquad (G.31)$$

$$s_2 = \frac{\psi(\sinh\psi - \psi)}{(2-2\cosh\psi + \psi\sinh\psi)} \qquad (G.32)$$

and equations G.16, G.17, and G.21 apply.

APPENDIX H

NEWTON METHOD TO FIND QR

Recalling that c_b is a function of QR, QR is found by obtaining the solution to

$$f(QR) = QR - \frac{L^2 A}{\pi^2 I} \left(\frac{u}{L} - c_b \right) = 0 \qquad (H.1)$$

Expanding f(QR) in a Taylor series,

$$f(QR + \Delta QR) = f(QR) + f'(QR)\,\Delta QR \qquad (H.2)$$

or

$$\Delta QR = - \frac{f(QR)}{f'(QR)} \qquad (H.3)$$

where the prime superscript denotes one differentiation with respect to QR. So,

$$f'(QR) = 1 + \frac{L^2 A}{\pi^2 I} c_b' \qquad (H.4)$$

where

$$c_b' = B_1'(\theta_1 + \theta_2)^2 + B_2'(\theta_1 - \theta_2)^2 \qquad (H.5)$$

$$s_1' = - 2 \pi^2 (B_1 + B_2) \qquad (H.6)$$

$$s_2' = - 2 \pi^2 (B_1 - B_2) \qquad (H.7)$$

B_1' and B_2' are determined from equations G.23 and G.24.

and

$$QR_{k+1} = QR_k + \Delta QR_k \tag{H.8}$$

for the $(k+1)^{st}$ iteration. Iteration proceeds until f(QR) is sufficiently close to zero.

APPENDIX I

LOCAL TANGENT STIFFNESS FOR THE BEAM-COLUMN MODEL

Combining equations H.1, G.22, G.23, and G.24,

$$QR = \frac{L^2 A}{\pi^2 I} \left(\frac{u}{L} - B_1(\theta_1 + \theta_2)^2 - B_2(\theta_1 - \theta_2)^2 \right) \quad (I.1)$$

From I.1 the following may be obtained.

$$QR_{,\theta_1} = G_1 / \pi^2 H \quad (I.2)$$

$$QR_{,\theta_2} = G_2 / \pi^2 H \quad (I.3)$$

$$QR_{,(u/L)} = 1 / H \quad (I.4)$$

where

$$G_1 = s_1' \theta_1 + s_2' \theta_2 \quad (I.5)$$

$$G_2 = s_1' \theta_2 + s_2' \theta_1 \quad (I.6)$$

$$H = \frac{\pi^2 I}{L^2 A} + B_1'(\theta_1 + \theta_2)^2 + B_2'(\theta_1 - \theta_2)^2 \quad (I.7)$$

and the comma subscript represents one partial differentiation with respect to each subscript following the comma.

Recalling that

$$\Delta S = k_T \Delta e \quad (I.8)$$

where

$$S = \begin{bmatrix} M_1 \\ M_2 \\ QL \end{bmatrix} \tag{I.9}$$

$$e = \begin{bmatrix} \theta_1 \\ \theta_2 \\ u/L \end{bmatrix} \tag{I.10}$$

and the component of k_T in the i^{th} row and j^{th} column is

$$k_{T_{ij}} = S_{i,e_j} + S_{i,QR}(QR_{,e_j}) \tag{I.11}$$

Application of equation I.11 gives equation 3.13 in which

$$k_T = t \quad . \tag{I.12}$$

APPENDIX J

GLOBAL TANGENT STIFFNESS FOR THE BEAM-COLUMN MODEL

The transformation from local forces for convected coordinates, p, to full local forces, f, is:

$$f = C^T p \tag{J.1}$$

or

$$\begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1/L \\ 1/L' & 1/L' & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1/L \\ -1/L' & -1/L' & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} M_1 \\ M_2 \\ QL \end{bmatrix} \tag{J.2}$$

where $L'=L(1+\quad)$ is the chord length of the deformed member (24). δ is neglected (set = 0 ; i.e. L' is approximated by L), but $\Delta\,\delta$ must be included for equation J.11. Contragredience (7) applies, so

$$e = C\ d \tag{J.3}$$

or

$$\begin{bmatrix} \theta_1 \\ \theta_2 \\ u/L \end{bmatrix} = \begin{bmatrix} 0 & 1/L' & 1 & 0 & -1/L' & 0 \\ 0 & 1/L' & 0 & 0 & -1/L' & 1 \\ 1/L & 0 & 0 & -1/L & 0 & 0 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \end{bmatrix} \quad (J.4)$$

The transformations from local, f, to global, F, foces and from global, D, to local, d, displacements for standard coordinates (7) are

$$d = \overline{\lambda}\, D \quad (J.5)$$

$$F = \overline{\lambda}^T f \quad (J.6)$$

where

$$\overline{\lambda} = \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} \quad (J.7)$$

and

$$\lambda = \begin{bmatrix} c_1 & c_2 & 0 \\ -c_2 & c_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (J.8)$$

From equations J.1, J.6, and 3.17 :

$$B = \overline{\lambda}^T\, C^T \quad (J.9)$$

(for F = B p)

(Note that c_1 and c_2 are direction cosines (7).)

$$B = \begin{bmatrix} -c_2/L' & -c_2/L' & c_1/L \\ c_1/L' & c_1/L' & c_2/L \\ 1 & 0 & 0 \\ c_2/L' & c_2/L' & -c_1/L \\ c_1/L' & -c_1/L' & -c_2/L \\ 0 & 1 & 0 \end{bmatrix} \qquad (J.10)$$

The components of K_0 (equation 3.24) are computed from equation 3.13 (for t) and equation J.10, by

$K_0 = B\ t\ B^T$.

Neglecting the difference between L and L' (or setting $\delta = 0$),

$$\Delta B = \frac{1}{L} \begin{bmatrix} -\Delta c_2 + c_2 \Delta\delta & -\Delta c_2 + c_2 \Delta\delta & \Delta c_1 \\ \Delta c_1 - c_1 \Delta\delta & \Delta c_1 - c_1 \Delta\delta & \Delta c_2 \\ 0 & 0 & 0 \\ \Delta c_2 - c_2 \Delta\delta & \Delta c_2 - c_2 \Delta\delta & -\Delta c_1 \\ -\Delta c_1 + c_1 \Delta\delta & -\Delta c_1 + c_1 \Delta\delta & -\Delta c_2 \\ 0 & 0 & 0 \end{bmatrix} \qquad (J.11)$$

From figure J.1 for $\delta = 0$,

$$\Delta\delta = [c_1(\Delta D_4 - \Delta D_1) + c_2(\Delta D_5 - \Delta D_2)]\ /L \qquad (J.12)$$

$$\Delta c_1 = [c_2^2(\Delta D_4 - \Delta D_1) - c_1 c_2(\Delta D_5 - \Delta D_2)]\ /L \qquad (J.13)$$

$$\Delta c_2 = [c_1 c_2(\Delta D_1 - \Delta D_4) + c_1^2(\Delta D_5 - \Delta D_2)]\ /L \qquad (J.14)$$

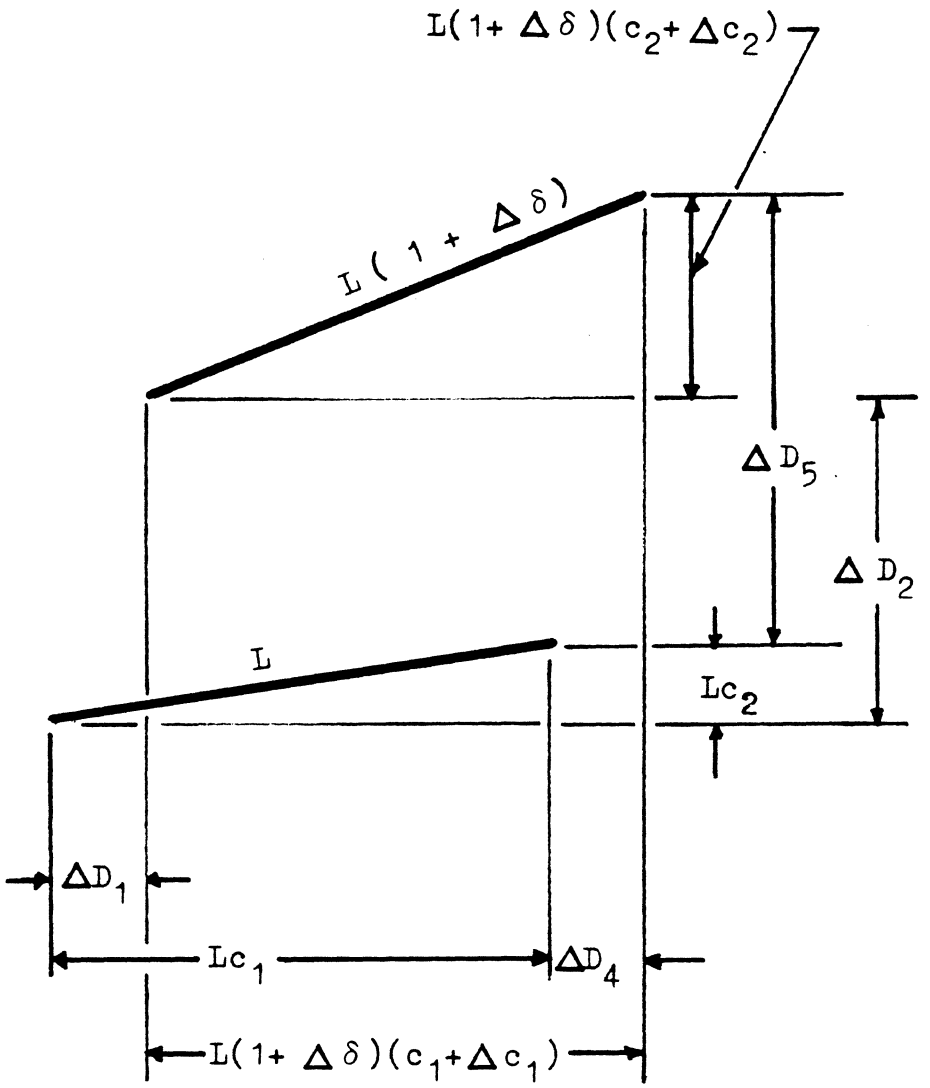


Figure J.1 : Variational State

or,

$$\Delta\delta = \frac{1}{L}\begin{bmatrix} -c_1 \\ -c_2 \\ 0 \\ c_1 \\ c_2 \\ 0 \end{bmatrix} \Delta D \qquad (J.15)$$

$$\Delta c_1 = \frac{1}{L}\begin{bmatrix} -c_2^2 \\ c_1c_2 \\ 0 \\ c_2^2 \\ -c_1c_2 \\ 0 \end{bmatrix} \Delta D \qquad (J.16)$$

$$\Delta c_2 = \frac{1}{L}\begin{bmatrix} c_1c_2 \\ -c_1^2 \\ 0 \\ -c_1c_2 \\ c_1^2 \\ 0 \end{bmatrix} \Delta D \qquad (J.17)$$

Recall equation 3.20 :

$$\Delta B\ p = \sum_{i=1}^{3} \left[p_i \left(g^{(i)} \right) \right] \Delta D \qquad (J.18)$$

Combining equations J.11, J.15, J.16, and J.17 gives

$$g^{(1)}=g^{(2)}=\frac{1}{L^2}\begin{bmatrix} t_1 & t_2 & 0 & -t_1 & -t_2 & 0 \\ t_2 & -t_1 & 0 & -t_2 & t_1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -t_1 & -t_2 & 0 & t_1 & t_2 & 0 \\ -t_2 & t_1 & 0 & t_2 & -t_1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad (J.19)$$

$$g^{(3)}=\frac{1}{L^2}\begin{bmatrix} t_3 & t_4 & 0 & -t_3 & -t_4 & 0 \\ t_4 & t_5 & 0 & -t_4 & -t_5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -t_3 & -t_4 & 0 & t_3 & t_4 & 0 \\ -t_4 & -t_5 & 0 & t_4 & t_5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad (J.20)$$

Where,

$$\begin{aligned} t_1 &= -2c_1c_2 \\ t_2 &= c_1^2 - c_2^2 \\ t_3 &= -c_2^2 \\ t_4 &= c_1\,c_2 \\ t_5 &= -c_1^2 \end{aligned} \qquad (J.21)$$

Combining equations J.18, J.19, J.20, and J.21 gives the components of k_σ in equation 3.25.

APPENDIX K

INITIAL STRESS STIFFNESS FOR THE FINITE ELEMENT METHOD

From figures K.1 a and c, observe that

$$x = \frac{L}{2} (1 + \xi) \tag{K.1}$$

so,

$$\frac{dx}{d\xi} = \frac{L}{2} \tag{K.2}$$

Since

$$v' = \frac{dv}{du} = \frac{dv}{d\xi} \tag{K.3}$$

and

$$\frac{dv}{d\xi} = \frac{dv}{dx} \frac{dx}{d\xi} = \frac{L}{2} \frac{dv}{dx} \tag{K.4}$$

so,

$$v_1' = \frac{L}{2} d_3 \tag{K.5}$$

$$v_2' = \frac{L}{2} d_6 \tag{K.6}$$

Define a vector d^* such that

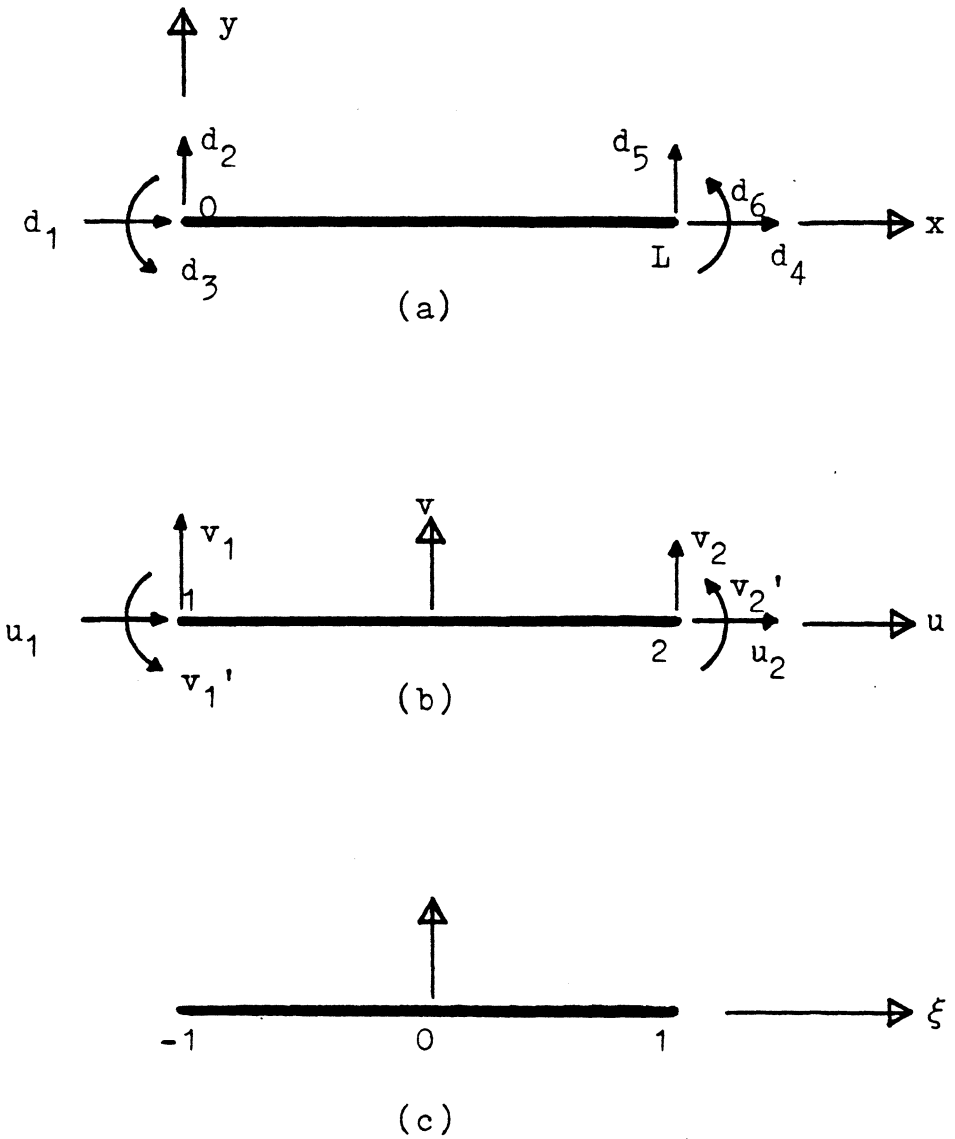


Figure K.1 : Displacements and Coordinates

$$\begin{bmatrix} u_1 \\ v_1 \\ v_1' \\ u_2 \\ v_2 \\ v_2' \end{bmatrix} = \begin{bmatrix} d_1^* \\ d_2^* \\ d_3^* \\ d_4^* \\ d_5^* \\ d_6^* \end{bmatrix} \qquad (K.7)$$

and, from figures K.1 a and b

$$d^* = T\ d \qquad (K.8)$$

where

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ & 1 & 0 & 0 & 0 & 0 \\ & & L/2 & 0 & 0 & 0 \\ & & & 1 & 0 & 0 \\ & & & & 1 & 0 \\ \text{symm.} & & & & & L/2 \end{bmatrix} \qquad (K.9)$$

as in reference 9.

The discretization process gives

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} N_1 & 0 & 0 & N_4 & 0 & 0 \\ 0 & N_2 & N_3 & 0 & N_5 & N_6 \end{bmatrix} \begin{bmatrix} d_1^* \\ d_2^* \\ d_3^* \\ d_4^* \\ d_5^* \\ d_6^* \end{bmatrix} \qquad (K.10)$$

where

$$
\begin{aligned}
N_1 &= \tfrac{1}{2}(1 - \xi) \\
N_2 &= \tfrac{1}{4}(2 + \xi)(1 - \xi)^2 \\
N_3 &= \tfrac{1}{4}(\xi + 1)(\xi - 1)^2 \\
N_4 &= \tfrac{1}{2}(1 + \xi) \\
N_5 &= \tfrac{1}{4}(2 - \xi)(1 + \xi)^2 \\
N_6 &= \tfrac{1}{4}(\xi - 1)(\xi + 1)^2
\end{aligned}
\qquad \text{(K.11)}
$$

are Hermite interpolation functions (7).

From equation 3.39,

$$\theta = G^* \, d^* \qquad \text{(K.12)}$$

Noting that

$$\theta = \begin{bmatrix} u_{,x} \\ v_{,x} \\ v_{,xx} \end{bmatrix} \qquad \text{(K.13)}$$

where the subscript ",x" denotes one partial differentiation with respect to x, and ",xx" denotes two partial differentiations with respect to x.

$$G^* = \begin{bmatrix} N_{1,x} & 0 & 0 & N_{4,x} & 0 & 0 \\ 0 & N_{2,x} & N_{3,x} & 0 & N_{5,x} & N_{6,x} \\ 0 & N_{2,xx} & N_{3,xx} & 0 & N_{5,xx} & N_{6,xx} \end{bmatrix} \qquad \text{(K.14)}$$

From equations 3.37, 3.50, 3.51, and 3.59,

$$k_\sigma = \sigma \int_V G^T H \, G \, dV \qquad \text{(K.15)}$$

From equations 3.37, K.14, and K.15,

$$k_\sigma^* = \sigma \int \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & N_{2,x}^2 & N_{2,x}N_{3,x} & 0 & N_{2,x}N_{5,x} & N_{2,x}N_{6,x} \\ 0 & N_{2,x}N_{3,x} & N_{3,x}^2 & 0 & N_{3,x}N_{5,x} & N_{3,x}N_{6,x} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & N_{2,x}N_{5,x} & N_{3,x}N_{5,x} & 0 & N_{5,x}^2 & N_{5,x}N_{6,x} \\ 0 & N_{2,x}N_{6,x} & N_{3,x}N_{6,x} & 0 & N_{5,x}N_{6,x} & N_{6,x}^2 \end{bmatrix} dV \qquad \text{(K.16)}$$

Integrating gives,

$$k_\sigma^* = \frac{\sigma A}{L} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ & 6/5 & 1/5 & 0 & -6/5 & 1/5 \\ & & 8/15 & 0 & -1/5 & -2/15 \\ & & & 0 & 0 & 0 \\ & & & & 6/5 & -1/5 \\ \text{symm.} & & & & & 8/15 \end{bmatrix} \qquad \text{(K.17)}$$

Noting that

$$k_\sigma = T^T k_\sigma^* T \qquad \text{(K.18)}$$

gives,

$$k_\sigma = \frac{\sigma A}{30L}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ & 36 & 3L & 0 & -36 & 3L \\ & & 4L^2 & 0 & -3L & -L^2 \\ & & & 0 & 0 & 0 \\ & & & & 36 & -3L \\ & \text{symm.} & & & & 4L^2 \end{bmatrix} \qquad (K.19)$$

For uniform axial stress,

$$\sigma = Q \,/\, A \qquad (K.20)$$

equations 3.68 are obtained as the contributions of k_σ in equation 3.66 and in references 5 and 28.

APPENDIX L

SIGN VECTOR

The value of SIGN(NEGPIV+1), where NEGPIV represents the number of negative eigenvalues, is used to determine loading or unloading for the equation that determines DQI from DS for the modified Riks/Wempner method. Consider the example of figure L.1. In region A, there are no negative eigenvalues (NEGPIV=0), and the structure is being loaded. The value SIGN(1) applies here and should be set equal to +1.DO (note that the value should be specified in double precision). In region B, NEGPIV=1 and SIGN(2)= +1.DO because the structure is still being loaded. In region C, NEGPIV=2 and SIGN(3)= -1.DO. In region D, NEGPIV=1, and SIGN(2) has already been determined to be +1.DO. Thus,

$$\text{SIGN} = \begin{bmatrix} +1.\text{DO} \\ +1.\text{DO} \\ -1.\text{DO} \end{bmatrix} \qquad \text{(L.1)}$$

for the example of figure L.1 (with analysis proceeding from left to right). A value of 0.DO follows the values of the SIGN vector to terminate the reading of the vector. For the Newton-Raphson method, only the final 0.DO need be specified.

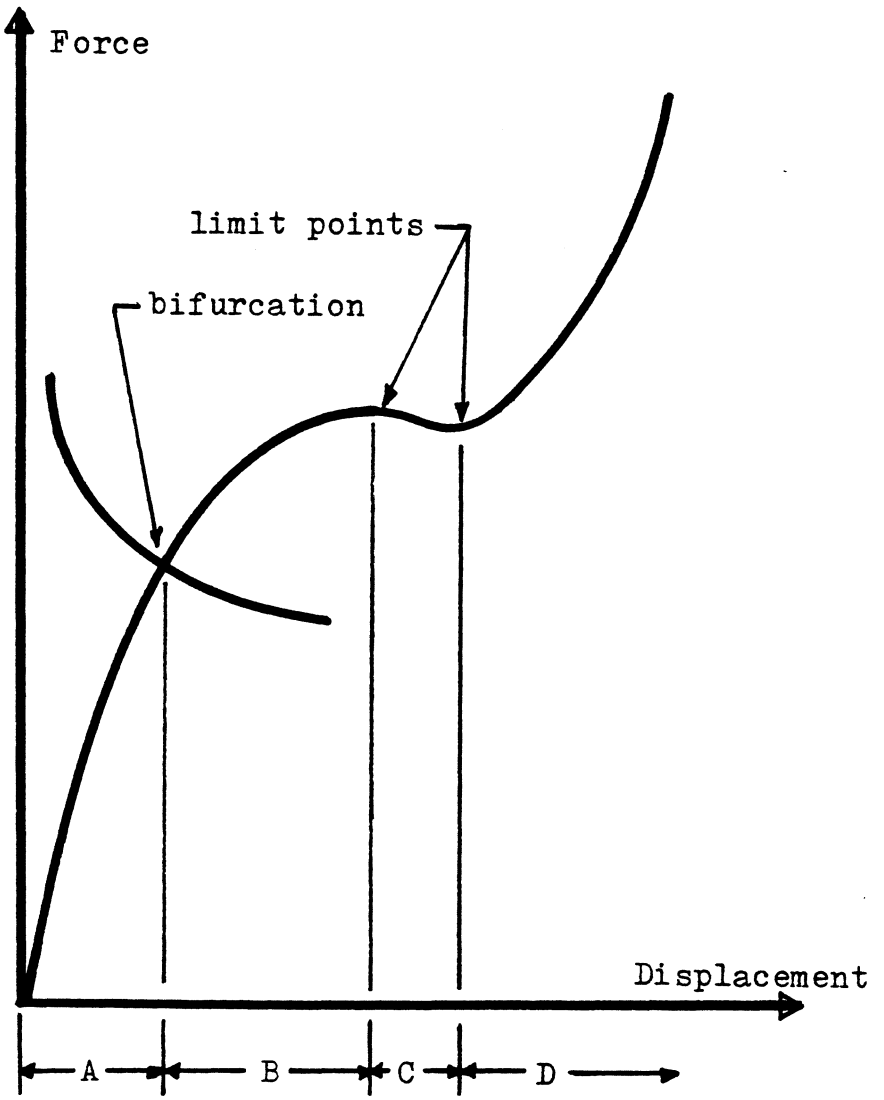


Figure L.1 : Equilibrium Path

APPENDIX M

VALUES FOR EQUILIBRIUM PLOTS

Tables of the values used to define some of the plots in chapter 5 are presented in this appendix.

Table M.1 : Test Problem 1, One Beam-Column Element

Force	Displacement	Force	Displacement
0.0038	0.0111	0.0334	0.2755
0.0064	0.0190	0.0333	0.2814
0.0081	0.0246	0.0331	0.2873
0.0098	0.0303	0.0330	0.2932
0.0114	0.0359	0.0329	0.2990
0.0130	0.0416	0.0327	0.3049
0.0145	0.0473	0.0326	0.3108
0.0159	0.0530	0.0324	0.3167
0.0172	0.0588	0.0323	0.3226
0.0185	0.0645	0.0321	0.3285
0.0198	0.0703	0.0320	0.3344
0.0209	0.0760	0.0319	0.3403
0.0221	0.0818	0.0318	0.3461
0.0231	0.0876	0.0317	0.3520
0.0241	0.0934	0.0316	0.3579
0.0251	0.0993	0.0315	0.3638
0.0260	0.1051	0.0314	0.3697
0.0268	0.1109	0.0313	0.3756
0.0276	0.1168	0.0313	0.3815
0.0283	0.1226	0.0313	0.3874
0.0289	0.1284	0.0313	0.3932
0.0296	0.1343	0.0313	0.3991
0.0301	0.1402	0.0313	0.4050
0.0306	0.1460	0.0313	0.4109
0.0311	0.1519	0.0315	0.4168
0.0315	0.1578	0.0316	0.4227
0.0319	0.1636	0.0317	0.4286
0.0323	0.1695	0.0319	0.4345
0.0326	0.1754	0.0320	0.4403
0.0329	0.1812	0.0323	0.4462
0.0331	0.1872	0.0326	0.4521
0.0333	0.1931	0.0329	0.4580
0.0335	0.1989	0.0332	0.4639
0.0336	0.2048	0.0336	0.4698
0.0337	0.2107	0.0340	0.4756
0.0338	0.2166	0.0344	0.4815
0.0338	0.2225	0.0349	0.4874
0.0339	0.2284	0.0354	0.4932
0.0339	0.2342	0.0360	0.4991
0.0338	0.2402	0.0366	0.5049
0.0337	0.2519	0.0372	0.5108
0.0337	0.2578	0.0379	0.5167
0.0336	0.2637	0.0387	0.5225
0.0335	0.2696	0.0395	0.5283

Table M.2 : Test Problem 1, Twelve Finite Elements

Force	Displacement
0.0038	0.0110
0.0064	0.0189
0.0088	0.0267
0.0111	0.0346
0.0132	0.0424
0.0152	0.0503
0.0171	0.0582
0.0189	0.0660
0.0205	0.0737
0.0221	0.0818
0.0235	0.0896
0.0248	0.0975
0.0260	0.1054
0.0271	0.1132
0.0281	0.1211
0.0290	0.1290
0.0298	0.1369
0.0305	0.1447
0.0312	0.1526
0.0317	0.1605
0.0322	0.1684
0.0326	0.1762
0.0330	0.1841
0.0333	0.1920
0.0335	0.1998
0.0337	0.2077
0.0338	0.2156
0.0339	0.2235
0.0339	0.2313
0.0339	0.2392
0.0338	0.2471
0.0338	0.2549
0.0337	0.2628
0.0335	0.2707
0.0334	0.2786
0.0332	0.2864
0.0330	0.2943
0.0328	0.3022
0.0327	0.3103
0.0325	0.3180
0.0323	0.3259
0.0321	0.3338
0.0319	0.3417
0.0318	0.3496
0.0316	0.3575
0.0315	0.3654
0.0314	0.3733
0.0314	0.3812
0.0314	0.3891
0.0314	0.3970
0.0314	0.4050
0.0315	0.4129
0.0316	0.4208
0.0318	0.4288
0.0321	0.4367
0.0324	0.4447
0.0327	0.4526
0.0331	0.4606
0.0336	0.4685
0.0341	0.4765
0.0348	0.4845
0.0354	0.4924
0.0362	0.5004
0.0371	0.5084
0.0380	0.5164
0.0390	0.5244
0.0401	0.5324
0.0413	0.5403
0.0426	0.5483
0.0440	0.5563
0.0455	0.5643
0.0470	0.5723
0.0487	0.5803
0.0505	0.5883

Table M.3 : Test Problem 2, One Element (Both Models Produce the Same Results)

Force	Displacement	Force	Displacement
0.0002	0.0198	0.0009	0.8735
0.0005	0.0397	0.0008	0.8934
0.0008	0.0596	0.0006	0.9132
0.0010	0.0795	0.0005	0.9330
0.0012	0.0994	0.0003	0.9529
0.0015	0.1192	0.0002	0.9727
0.0016	0.1391	0.0001	0.9925
0.0018	0.1590	-0.0001	1.0124
0.0020	0.1789	-0.0002	1.0322
0.0021	0.1987	-0.0004	1.0520
0.0023	0.2186	-0.0005	1.0719
0.0024	0.2385	-0.0007	1.0917
0.0025	0.2583	-0.0008	1.1115
0.0026	0.2782	-0.0010	1.1313
0.0027	0.2980	-0.0011	1.1512
0.0028	0.3378	-0.0012	1.1710
0.0028	0.3576	-0.0014	1.1909
0.0029	0.3775	-0.0015	1.2107
0.0029	0.3973	-0.0016	1.2305
0.0029	0.4172	-0.0018	1.2504
0.0029	0.4370	-0.0019	1.2702
0.0029	0.4569	-0.0020	1.2900
0.0029	0.4767	-0.0021	1.3099
0.0028	0.4966	-0.0022	1.3297
0.0028	0.5164	-0.0023	1.3495
0.0028	0.5363	-0.0024	1.3694
0.0027	0.5561	-0.0025	1.3892
0.0026	0.5760	-0.0026	1.4091
0.0026	0.5958	-0.0027	1.4289
0.0025	0.6157	-0.0027	1.4488
0.0024	0.6355	-0.0028	1.4686
0.0023	0.6553	-0.0028	1.4884
0.0022	0.6752	-0.0029	1.5083
0.0021	0.6950	-0.0029	1.5281
0.0020	0.7148	-0.0029	1.5480
0.0019	0.7347	-0.0029	1.5678
0.0017	0.7545	-0.0029	1.5877
0.0016	0.7744	-0.0029	1.6075
0.0015	0.7942	-0.0029	1.6274
0.0013	0.8140	-0.0028	1.6473
0.0012	0.8339	-0.0028	1.6671
0.0011	0.8537	-0.0027	1.6870

Table M.3 (continued) : Test Problem 2, One Element (Both Models Produce the Same Results)

Force	Displacement
-0.0027	1.7068
-0.0026	1.7267
-0.0025	1.7466
-0.0024	1.7664
-0.0022	1.7863
-0.0021	1.8062
-0.0019	1.8260
-0.0018	1.8459
-0.0016	1.8658
-0.0014	1.8857
-0.0012	1.9055
-0.0010	1.9254
-0.0007	1.9453
-0.0004	1.9652
-0.0002	1.9851
-0.0001	2.0050
0.0002	2.0190
0.0005	2.0331
0.0007	2.0471
0.0010	2.0612
0.0012	2.0753
0.0015	2.0893
0.0017	2.1034
0.0020	2.1175
0.0023	2.1315
0.0026	2.1456
0.0030	2.1597
0.0033	2.1737
0.0036	2.1879
0.0040	2.2019
0.0043	2.2160

APPENDIX N

PROGRAM LISTING

```
C
C    GEOMETRICALLY NONLINEAR STATIC ANALYSIS OF PLANE FRAMES
C    THE DISPLACEMENT-BASED FINITE ELEMENT ANALYSIS IS IMPLEMENTED
C    IN THE TOTAL LAGRANGIAN FORMULATION. THE ANALYSIS MAY BE PERFORMED
C    USING EITHER THE NEWTON-RAPHSON OR THE MODIFIED RIKS/WEMPNER
C    ITERATION SCHEMES. POST-BUCKLING ANALYSIS MAY BE PERFORMED USING
C    THE LATTER. THE FREQUENCY OF UPDATING THE STIFFNESS MATRIX
C    MAY ALSO BE CONTROLLED BY THE USER. NOTE: MANY PARAMETERS MAY BE
C    SET AT THE BEGINNING OF THE PROGRAM BUT NEED NOT BE READ IN THE
C    DATA SETS (THIS ENABLES THE USER TO PERFORM A SIMILAR ANALYSIS
C    OF SEVERAL STRUCTURES WITHOUT REPEATING PARAMETER SPECIFICATIONS
C    IN DATA SETS).
C    THE MEMBERS MUST BE PRISMATIC, BUT THEY MAY HAVE DISTINCT GEOMETRIC
C    AND MATERIAL PROPERTIES. TWO MODELS FOR ANALYSIS ARE PROVIDED:
C    THE FIRST HAS BEEN DEVELOPED USING CLASSICAL BEAM-COLUMN THEORY
C    AND THE SECOND BY STANDARD FINITE ELEMENT PROCEDURE USING
C    HERMITE INTERPOLATION FUNCTIONS TO APPROXIMATE STRAIN-DISPLACEMENT
C    RELATIONS AND WITH EQUILIBRIUM FORMULATED BY THE
C    PRINCIPLE OF VIRTUAL WORK. PRINTOUT OPTIONS ARE PROVIDED AND
C    SELECTION IS ALLOWED BETWEEN PRINTING THE FORCES AND DISPLACEMENTS
C    AT ONE DEGREE OF FREEDOM OR PRINTING THE FOLLOWING AT EACH
C    EQUILIBRIUM POINT:JOINT DISPLACEMENTS, LOCAL ELEMENT FORCES, AND
C    JOINT FORCES.
C
C
C
C
C                    PARAMETERS
C                    **********
C
C    *               TOLERANCES MAY BE DEFINED BY USER AT THE BEGINNING OF
C    *               THE PROGRAM. SET A CONV TOL GREATER THAN 1 TO
C    *               NOT PERFORM THE CORRESPONDING TEST.
C
C    CPDB            CONVERGENCE TOLERANCE FOR 2-NORM DISPLACEMENT TEST
C    CPDC            CONVERGENCE TOLERANCE FOR INFINITY-NORM DISPL TEST
C    CPE             CONVERGENCE TOLERANCE FOR ENERGY TEST
C    CPF             CONVERGENCE TOLERANCE FOR UNBALANCED FORCE TEST
C    DP              DIVERGENCE TOLERANCE FOR ENERGY TEST
C    DQQR            CONVERGENCE TOLERANCE FOR ITERATION IN BOWCOR
C    IMAX            MAX NO. OF ITERATIONS PERMITTED IN BOWCOR
C    IMP             DEGREE OF FREEDOM FOR WHICH RESULTS ARE PRINTED
C                       FOR NPRINT = 0, 1, 2
C    ITMAX           MAX NO. OF ITERATIONS PERMITTED FOR NEWTON-RAPHSON
C                       OR MODIFIED RIKS/WEMPNER
C    MX              MAX NUMBER OF ELEMENTS OR JOINTS
C    MXNEQ           MAX NUMBER OF EQUATIONS (D.O.F.)
C    NPRINT          PRINT INDICATOR  =  0 FOR EQUILIBRIUM PATH PLOT DATA
C                                           (FOR ONE D.O.F. ) (NO WORDS ARE
C                                           PRINTED)
C                                    =  1 FOR EQUILIBRIUM PATH PLOT DATA
```

```
C                                        WITH TANGENTS
C                                    =  2 FOR EQUILIBRIUM PATH RESULTS
C                                       INCLUDING EXPLANATIONS OF RES.
C                                    =  3 FOR FULL DEBUGGING OUTPUT
C                                    =  4 FOR FULL EQUILIBRIUM COFIG.
C                                       OUTPUT
C  NUPD              NUMBER OF ITERATIONS BETWEEN UPDATING TANGENT
C                       STIFFNESS MATRIX.  = 1 FOR UPDATING AT EVERY
C                       ITERATION. (MUST BE .GE. 1 )
C
C
C                    VARIABLES
C                    *********
C
C
C  A1(NAT),A2(NAT)  APPLIED MEMBER ACTIONS
C  AREA(MX)         CROSS SECTIONAL AREA OF ELEMENT I
C  BP1(MX),BP2(MX)  FIRST DERIVATIVE OF BW1, BW2 RESPECTIVELY WITH
C                      RESPECT TO QR
C  BW1(MX),BW2(MX)  BOWING FUNCTIONS
C  C                CONVERGENCE RATIO TO BE COMPARED TO CONVERGENCE
C                      PARAMETERS
C  C1(MX),C2(MX)    DIRECTION COSINES OF ELEMENT I AT ANY TIME
C  CI1(MX),CI2(MX)  DIRECTION COSINES OF INITIAL CONFIGURATION
C  CB               LENGTH CORRECTION FACTOR FOR BOWING ACTION
C  CBP              DERIVATIVE OF CB WITH RESPECT TO QR
C  CD,CN            DENOMINATOR, NUMERATOR OF C
C  CPR,CPT          CONVERGENCE RATIOS FOR ROTATION,TRANSLATION TO BE
C                      COMPARED TO CONV. PARAMETER
C  CR,CT            MAXIMUM CHANGE IN ROTATION, TRANSLATION
C  D(MXNEQ)         TOTAL GLOBAL DISPLACEMENT VECTOR
C  DD(MXNEQ)        CHANGE IN D ("DELTA-D")
C  DD0(MXNEQ)       DD FOR FIRST STEP OF MOD. R/W ITERATION TO A
C                      NEW EQUILIBRIUM POINT
C  DD01(MXNEQ)      FOR MOD. R/W METHOD: DD0 CONTRIBUTION (DD0=DQI*DD01)
C  DD1(MXNEQ)       FOR N-R METHOD: DD FOR FIRST ITERATION
C                   FOR MOD. R/W METHOD: DD CONTRIBUTION (DD=DQI*DD1+DD2)
C  DD2(MXNEQ)       FOR MOD. R/W METHOD: DD CONTRIBUTION (DD=DQI*DD1+DD2)
C  CDE(6)           GLOBAL DISPLACEMENTS OF ELEMENT BEING CONSIDERED
C  DIST(NAT)        DISTANCE TO MEMBER ACTION
C  DIST2(NAT)       DISTANCE TO END OF MEMBER ACTION
C  DJ(3,MX)         GLOBAL JOINT DISPLACEMENT MATRIX
C  DOT1(N),DOT2(N)  VECTORS FOR WHICH DOT PRODUCT IS OBTAINED IN DOTPRD
C  DQ               CHANGE IN QR USED TO CONVERGE TO QR IN BOWCOR
C  DQI              CHANGE IN LOAD INCREMENT
C  ELENG(MX)        LENGTH OF ELEMENT I
C  EMOD(MX)         MODULUS OF ELASTICITY OF ELEMENT I
C  F(MXNEQ)         ELEMENT FORCE VECTOR CAUSED BY DEFORMATIONS
C  FF(6,MX)         FIXED END FORCES DUE TO MEMBER ACTIONS
C  FG(6,MX)         GLOBAL ELEMENT FOCE MATRIX
C                      F(L,I) = LTH GLOBAL FORCE OF MEMBER I
C  FL(6,MX)         LOCAL ELEMENT FORCE MATRIX
C                      F(L,I)= LTH LOCAL FORCE OF ELEMENT I
C  FP(MXNE)         F FROM PREVIOUS LOAD INCREMENT
C  FPI(MXNE)        F FROM PREVIOUS N-R ITERATION
```

```
C  G(10)            GLOBAL ELEMENT STIFFNESS COEFFICIENTS
C  G1,G2             INTERMEDIATE FUNCTIONS USED TO COMPUTE SKT FOR
C                      BEAM-COLUMN MODEL
C  H                 INTERMEDIATE FUNCTION USED TO COMPUTE SKT
C  GP1-GP7           INTERMEDIATE FUNCTION USED TO COMPUTE SKT
C  IC                ITERATION COUNTER IN BOWCOR
C  ICI               CONVERGENCE INDICATOR
C                      = 0  AFTER TEST FOR CONVERGED
C                      .NE.0 AFTER TEST FOR NOT CONVERGED
C                      N-R OR MOD. R/W ITERATION PROCEDES AS LONG AS
C                      ICI DOES NOT = 0 (UNTIL MAX NO. ITERATIONS)
C                      ICI IS SET = 0 AT THE BEGINNING OF TEST SO THAT
C                      SOLUTION IS ASSUMED TO BE CONVERGED UNTIL IT IS
C                      PROVEN OTHERWISE. ICI IS ONLY CHANGED FROM ITS
C                      ZERO VALUE IF A CONVERGENCE TEST IS FAILED.
C                      AFTER TEST THE VALUE OF ICI MAY BE PRINTED TO
C                      INDICATE WHICH TESTS WERE FAILED:
C                        ICI=ICI+1      IF DISPLC WAS FAILED
C                        ICI=ICI+10     IF DISPLB WAS FAILED
C                        ICI=ICI+100    IF UNBALF WAS FAILED
C                        ICI=ICI+1000   IF ENERGY WAS FAILED
C  INDEX(6,6)        MATRIX DEFINING LOCATION OF G IN SKT
C                      SKT(I,J)=G(INDEX(I,J))
C                      THE SIGN OF INDEX(I,J) IS THE SIGN OF SKT(I,J)
C  ITCT              ITERATION COUNTER FOR N-R AND MOD. R/W ITERATIONS
C  ITDES             NUMBER OF ITERATIONS DESIRED FOR MOD. R/W METHOD
C                      TO OBTAIN NEXT EQUILIB. CONFIGURATION
C  JCODE(3,MXNJ)     JOINT CODE MATRIX: JCODE(I,J)= THE D.O.F. NUMBER
C                      AT JOINT J IN THE GLOBAL I-DIRECTION
C  JDIR              JOINT DIRECTION OF APPLIED JOINT FORCE
C  JNUM              JOINT NUMBER THAT FORCE IS APPLIED TO
C  KHT(I)            COLUMN HEIGHT OF FULL TANGENT STIFFNESS FOR DEGREE
C                      OF FREEDOM I
C  KT(5)             KT(I) IS GREATER THAN ZERO TO INCLUDE THE
C                      CORRESPONDING CONTRIBUTION TO SKT
C  MAT(NAT)          MEMBER ACTION TYPE:
C                      FOR INDTAN = 1 : MAT
C                                       =1 FOR POINT LOAD
C                                       =2 FOR DISTRIBUTED LOAD
C                                       =3 FOR APPLIED MOMENT
C                      FOR INDTAN = 2 OR 3 : MAT
C                                       =1 FOR POINT LOAD
C                                       =2 FOR UNIFORM LOAD
C                                       =3 FOR INCREASING TRIANGULAR LOAD
C                                       =4 FOR DECREASING TRIANGULAR LOAD
C                                       =5 FOR AXIAL POINT LOAD
C                                       =6 FOR UNIFORM AXIAL LOAD
C  MAXA(I)           STORES ADDRESSES OF DIAGONAL TERMS IN THE COLUMN
C                      VECTOR REPRESENTATION OF THE STIFFNESS MATRIX
C  MCODE(6,MX)       MEMBER CODE MATRIX:
C                      MCODE(I,J) IS THE DEGREE OF FREEDOM NUMBER IN
C                      THE ITH GLOBAL DIRECTION OF MEMBER J
C  MINC(2,MX)        MEMBER INCIDENCE MATRIX = JOINT NUMBERS AT EACH END
C                      OF MEMBER
C  MN(NAT)           MEMBER NUMBER TO WHICH ELEMENT ACTION IS APPLIED
```

```
C  NAT              TOTAL NUMBER OF MEMBER ACTIONS
C  NC               COUNTER FOR NUMBER OF ITERATIONS BETWEEN UPDATING SKT
C  NE               NUMBER OF ELEMENTS
C  NEGPIV           NUMBER OF NEGATIVE PIVOTS ENCOUNTERED IN THE
C                      FACTORIZATION OF SKT = NO. OF NEGATIVE EIGENVALUES
C  NEQ              NUMBER OF EQUATIONS (D.O.F.)
C  NJ               NUMBER OF JOINTS
C  P(3,MX)          JOINT FORCE MATRIX
C                      P(I,J)= FORCE AT JOIN J IN DIRECTION I
C  Q(NEQ)           APPLIED FORCE DISTRIBUTION (UNFACTORED)
C  QI               MULTIPLIER OF Q FOR CURRENT LOAD LEVEL OR INCREMENT
C                      (SOME REFERENCES CALL THIS LAMBDA)
C  QIMAX            MAXIMUM ALLOWED LOAD LEVEL
C  QL               AXIAL FORCE / ELENG
C  QT(NEQ)          APPLIED LOAD VECTOR FOR CURRENT LOAD INCREMENT
C  QR               RATIO OF AXIAL FORCE TO EULER BUCKLING LOAD
C  R1(MX),R2(MX)    ROTATION AT A- AND B-END OF MEMBER
C  RM1,RM2          MOMENTS AT A- AND B-END OF MEMBER
C  RM12(MX)         RM1+RM2 / ELENG
C  RMR,RMT          MAXIMUM ROTATION, TRANSLATION IN STRUCTURE
C  SIGN(NEGPIV+1)   +1.DO FOR A LOADING REGION
C                   -1.DO FOR AN UNLOADING REGION
C  SKT(I)           SYSTEM TANGENT STIFFNESS MATRIX
C  SP1(MX),SP2(MX)  FIRST DERIVATIVE OF ST1, ST2 WITH RESPECT TO QR
C  ST1(MX),ST2(MX)  STABILITY FUNCTIONS
C  UL               AXIAL DISPLACEMENT DIVIDED BY INITIAL LENGTH
C                      OF MEMBER(POS FOR SHORTENING)
C  U(MX)            AXIAL LENGTHENING OF MEMBER
C                      ( = -UL*ELENG )
C  X(1,J),X(2,J)    GLOBAL 1,2-COORDINATES OF JOINT J
C  ZI(I)            MOMENT OF INERTIA ABOUT LOCAL Z-AXIS OF ELEM. I
C
C
C
C
C
C        DATA CARDS
C        **** *****
C
C
C *NOTE: ALL VARIABLES USE STANDARD WATFIV DEFAULT TYPING; I.E. IF THE
C        FIRST LETTER OF THE VARIABLE NAME IS BETWEEN I AND N INCLUSIVE,
C        THE VARIABLE IS AN INTEGER.
C
C   QI,QIMAX,DQI
C   IMP
C   SIGN(I)
C    :   :
C    :   :
C   (FOR TOTAL NUMBER OF NEGATIVE EIGENVALUES POSSIBLE + 1)
C   0.DO
C   KT(I),FOR I=1 TO 5
C   NE,NJ
C   MINC(1,I),MINC(2,I),FOR I=1 TO NE
C   JOINT NUMBER, CONSTRAINT GLOBAL DIRECTION
```

```
C             :               :
C             :               :
C          (FOR ALL CONSTRAINTS)
C    0,0
C    X(1,J),X(2,J),FOR J=1 TO NJ
C    AREA(I),EMOD(I),ZI(I)
C         :               :
C         :               :
C     (FOR ALL MEMBERS)
C    JNUM , JDIR , FORCE
C       :               :
C       :               :
C   (FOR ALL JOINT FORCES)
C    0,0,0.D0
C    MN(I),MAT(I),A1(I),A2(I),DIST(I),DIST2(I)
C         :               :
C         :               :
C     (FOR ALL MEMBER ACTIONS)
C    0,0,0.D0,0.D0,0.D0,0.D0,0.D0
C
C
C
C
C
C      ************************************************************
C      *                                                          *
C      *                          MAIN                            *
C      *                                                          *
C      ************************************************************
C
C      MAIN INITIALIZES AND READS IN PARAMETERS, AND INITIALIZES THE
C      CONFIGURATION TO THE UNDEFORMED STATE. DATA AND EITHER NEWRAP
C      OR RIKWEM ARE CALLED.
C
       IMPLICIT REAL*8(A-H,O-Z)
       DIMENSION A1(21),A2(21),AREA(20),BP1(20),BP2(20),C1(20),C2(20),D(6
      10),DD(60),DD1(60),DE(6),DIST(21),DJ(3,20),ELENG(20),EMOD(20),F(60)
      2,FF(6,20),FG(6,20),FL(6,20),FP(60),FPI(60),G(10),INDEX(6,6),JCODE(
      33,20),KHT(61),MAT(21),MAXA(61),MCODE(6,20),MINC(2,20),MN(21),P(3,2
      40),Q(60),QT(60),R1(60),R2(60),SKT(1860),SP1(20),SP2(20),ST1(20),ST
      52(20),X(2,20),ZI(20),CI1(20),CI2(20),DD2(60),Z(60),SIGN(10),DOT1(6
      60),DOT2(60),DD0(60),DD01(60),RM12(20),QL(20),U(20),KT(5),DIST2(21)
      7,QJ(60)
C
C      COVERGENCE PARAMETERS
C
C     CPDB : ICI+10
       CPDB=1.D-03
C      1E-06 < CPDC< 1E-02
C     CPDC : ICI+1
       CPDC=1.D-02
C     CPF : ICI+100
       CPF=1.D-01
C     CPE : ICI+1000
       CPE=1.D-03+1
```

```
      DP=1.D00
      DQQR=1.D-06
      FQR=1.D-09
C
C     INDTAN = 1 FOR BEAM-COLUMN TANGENT STIFFNESS
C            = 2 FOR FINITE ELEMENT TANGENT STIFFNESS(CONVECTED COORDS)
C            = 3 FOR FINITE ELEMENT TANGENT STIFFNESS
      INDTAN=3
C
C     ITDES = NUMBER OF ITERATIONS DESIRED FOR CONVERGENCE IN
C             MODIFIED RIKS WEMPNER METHOD
      ITDES=2
C
C     ITIND = 0 FOR NEWTON RAPHSON ITERATION
C           = 1 FOR MODIFIED RIKS WEMPNER METHOD
      ITIND=1
C
C     MAXIMUM LIMITS
C
      IMAX=100
      ITMAX=100
      MX=20
      MXNA=20
      MXNEQ=3*MX
C
C     NUMBER OF ITERATIONS BETWEEN UPDATING
      NUPD=1
C
C    NPRINT = 0 FOR EQUILIBRIUM PATH PLOT
C             1 FOR EQUILIBRIUM PATH PLOT WITH TANGENTS
C             2 FOR EQUILIBRIUM PATH RESULTS
C             3 FOR FULL DEBUGGING OUTPUT
C             4 FOR FULL FINAL OUTPUT
      NPRINT=1
C
      IN=0
      READ,QI,QIMAX,DQI
C     IMP = IMPORTANT DEGREE OF FREEDOM (D.O.F. FOR WHICH PRINTOUT IS
C           DESIRED)
      READ,IMP
      PRINT 100,NUPD,CPDB,CPDC,CPF,CPE,DP,DQQR,FQR,IMAX,ITMAX,QI,QIMAX,
     1DQI,ITDES,ITIND,INDTAN,IMP,NPRINT
  100 FORMAT('-NUPD = ',I4/'0CPDB = ',F10.6/'0CPDC = ',F10.6/'0CPF  = ',
     1F10.6/'0CPE  = ',F10.6/'0DP   = ',F10.6/'0DQQR = ',D15.7,'  FQR =
     2',D15.7/'0IMAX = ',I6/'0ITMAX= ',I6/'0QI   = ',F15.5/'0QIMAX= ',
     3F15.5/'0DQI  = ',F15.5/'0ITDES= ',I6,'  ITIND= ',I6/'0INDTAN= ',I3
     4/'   RESULTS FOR D.O.F.  ',I4,'   NPRINT= ',I3)
C
        PRINT 200
  200   FORMAT('0  I',10X,'SIGN(I)')
        I=1
        READ,SIGN(I)
        WHILE(SIGN(I).NE.0.D00) DO
          PRINT 300,I,SIGN(I)
  300     FORMAT(' ',I3,5X,F10.1)
```

```
          I=I+1
          READ,SIGN(I)
        END WHILE
C
        READ,(KT(J),J=1,5)
        PRINT 400,(KT(J),J=1,5)
  400   FORMAT(' KT=',5(3X,I4))
C
      CALL DATA(A1,A2,AREA,C1,C2,DIST,ELENG,EMOD,FF,FL,JCODE,KHT,MAT,MAX
     1A,MCODE,MINC,MN,Q,QT,X,ZI,MX,MXNA,MXNEQ,NAT,NE,NEQ,NJ,NKT,QI,CI1,C
     2I2,DIST2)
C
      DO 10 I=1,NEQ
        F(I)=0.D00
        FP(I)=0.D00
        FPI(I)=0.D00
        D(I)=0.D00
        Z(I)=0.D00
        QJ(I)=Q(I)
   10 CONTINUE
      DO 15 J=1,NE
        FL(1,J)=0.D0
   15 CONTINUE
C
      IELS=0
      IF(ITIND.EQ.0) THEN DO
        CALL NEWRAP(AREA,BP1,BP2,C1,C2,D,DD,DD1,DE,ELENG,EMOD,F,FG,FL,FP
     1,FPI,G,JCODE,MAXA,MCODE,QT,R1,R2,SKT,SP1,SP2,ST1,ST2,ZI,CPDB,CPDC,
     2CPE,CPF,DP,DQQR,ICI,IELS,IMAX,ITMAX,NE,NEQ,NJ,NKT,NUPD,A1,A2,DIST,
     3FF,MAT,MN,Q,NAT,QI,CI1,CI2,FQR,IN,QIMAX,DQI,DJ,MINC,P,ITIND,RM12,Q
     4L,U,INDTAN,KT,IMP,NPRINT,DIST2,QJ)
      ELSE DO
        QI=0.D0
        CALL RIKWEM(AREA,BP1,BP2,C1,C2,D,DD,DD1,DE,ELENG,EMOD,F,FG,FL,FP
     1,FPI,G,JCODE,MAXA,MCODE,QT,R1,R2,SKT,SP1,SP2,ST1,ST2,ZI,CPDB,CPDC,
     2CPE,CPF,DP,DQQR,ICI,IELS,IMAX,ITMAX,NE,NEQ,NJ,NKT,NUPD,A1,A2,DIST,
     3FF,MAT,MN,Q,NAT,QI,CI1,CI2,FQR,IN,DD2,Z,SIGN,ITDES,QIMAX,DQI,DJ,MI
     4NC,P,DOT1,DOT2,ITIND,DD0,DD01,RM12,QL,U,INDTAN,KT,IMP,NPRINT,DIST2
     5,QJ)
      END IF
      STOP
      END
C
C     ***********************************************************
C     *                                                         *
C     *                         DATA                            *
C     *                                                         *
C     ***********************************************************
C
C     DATA READS THE NUMBER OF ELEMENTS AND JOINTS,TESTS THESE
C     AGAINST THE MAXIMUM, AND CALLS STRUCT AND LOAD.
C
      SUBROUTINE DATA(A1,A2,AREA,C1,C2,DIST,ELENG,EMOD,FF,FL,JCODE,KHT,M
     1AT,MAXA,MCODE,MINC,MN,Q,QT,X,ZI,MX,MXNA,MXNEQ,NAT,NE,NEQ,NJ,NKT,QI
     2,CI1,CI2,DIST2)
```

```
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A1(1),A2(1),AREA(1),C1(1),C2(1),DIST(1),ELENG(1),EMOD(1)
     1,FF(6,1),FL(6,1),JCODE(3,1),KHT(1),MAT(1),MAXA(1),MCODE(6,1),MINC(
     22,1),MN(1),Q(1),QT(1),X(2,1),ZI(1),CI1(1),CI2(1),DIST2(1)
C
      READ,NE,NJ
      PRINT 100,NE,NJ
  100 FORMAT('-NE = ',I4,7X,'NJ = ',I4)
C
      IF(NE.LE.MX.AND.NJ.LE.MX) THEN DO
        CALL STRUCT(AREA,C1,C2,ELENG,EMOD,JCODE,KHT,MAXA,MCODE,MINC,X,ZI
     1,MXNEQ,NE,NEQ,NJ,NKT,CI1,CI2)
        CALL LOAD(A1,A2,C1,C2,DIST,ELENG,EMOD,FF,FL,JCODE,MAT,MCODE,MN,Q
     1,QT,ZI,MXNA,NAT,NE,NEQ,QI,DIST2)
      ELSE DO
        PRINT 200
  200   FORMAT('-***NE OR NJ EXCEDES MX; REDIMENSION ARRAYS')
        STOP
      END IF
      RETURN
      END
C
C     ********************************************************
C     *                                                      *
C     *                        STRUCT                        *
C     *                                                      *
C     ********************************************************
C
C     STRUCT READS MINC AND JOINT CONSTRAINTS, JCODE IS INITIALIZED
C     TO UNITY AND THEN SET TO ZERO WHERE CONSTRAINTS OCCUR.
C     CODES, DETMAX, AND PROP ARE CALLED.
C
      SUBROUTINE STRUCT(AREA,C1,C2,ELENG,EMOD,JCODE,KHT,MAXA,MCODE,MINC,
     1X,ZI,MXNEQ,NE,NEQ,NJ,NKT,CI1,CI2)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION AREA(1),C1(1),C2(1),ELENG(1),EMOD(1),JCODE(3,1),KHT(1),M
     1AXA(1),MCODE(6,1),MINC(2,1),X(2,1),ZI(1),CI1(1),CI2(1)
C
      READ,(MINC(1,I),MINC(2,I),I=1,NE)
      PRINT 100
  100 FORMAT('-',7X,'MINC')
      DO 10 I=1,2
        PRINT 200,(MINC(I,J),J=1,NE)
  200   FORMAT('0',22(2X,I4))
C
   10 CONTINUE
      DO 30 J=1,NJ
        DO 20 I=1,3
          JCODE(I,J)=1
   20   CONTINUE
   30 CONTINUE
C
      PRINT 300
  300 FORMAT('-',18X,'JOINT NUMBER',10X,'CONSTRAINT DIRECTION')
C
```

```
      READ,JNUM,JDIR
      WHILE(JNUM.NE.0) DO
        PRINT 400,JNUM,JDIR
  400   FORMAT('0',21X,I4,24X,I2)
        JCODE(JDIR,JNUM)=0
        READ,JNUM,JDIR
      END WHILE
C
      CALL CODES(JCODE,MCODE,MINC,MXNEQ,NE,NEQ,NJ)
      CALL DETMAX(KHT,MAXA,MCODE,NE,NEQ,NKT)
      CALL PROP(AREA,C1,C2,ELENG,EMOD,MINC,X,ZI,NE,NJ,CI1,CI2)
C
      RETURN
      END
C
C     ********************************************************
C     *                                                      *
C     *                       CODES                          *
C     *                                                      *
C     ********************************************************
C
C     CODES ASSIGNS DEGREE OF FREEDOM NUMBERS TO JCODE AND GENERATES
C     MCODE USING JCODE AND MINC.
C
      SUBROUTINE CODES(JCODE,MCODE,MINC,MXNEQ,NE,NEQ,NJ)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION JCODE(3,1),MCODE(6,1),MINC(2,1)
C
      NEQ=0
      DO 20 J=1,NJ
        DO 10 L=1,3
          IF(JCODE(L,J).NE.0) THEN DO
            NEQ=NEQ+1
            JCODE(L,J)=NEQ
          END IF
   10   CONTINUE
   20 CONTINUE
C
      IF(NEQ.GT.MXNEQ) THEN DO
        PRINT 100
  100   FORMAT('- *** NEQ EXCEDES MXNEQ ; REDIMENSION ARRAYS ***')
        STOP
      END IF
      PRINT 200
  200 FORMAT('-',7X,'JCODE')
      DO 30 I=1,3
        PRINT 300,(JCODE(I,J),J=1,NJ)
  300   FORMAT('0',22(2X,I4))
   30 CONTINUE
C
      DO 50 I=1,NE
        J=MINC(1,I)
        K=MINC(2,I)
        DO 40 L=1,3
          MCODE(L,I)=JCODE(L,J)
```

```
            MCODE(L+3,I)=JCODE(L,K)
   40     CONTINUE
   50   CONTINUE
C
        PRINT 400
  400   FORMAT('-',7X,'MCODE')
        DO 60 I=1,6
          PRINT 500,(MCODE(I,J),J=1,NE)
  500     FORMAT('0',22(2X,I4))
   60   CONTINUE
        RETURN
        END
C
C       ********************************************************
C       *                                                      *
C       *                        DETMAX                        *
C       *                                                      *
C       ********************************************************
C
C       DETMAX DETERMINES KHT USING MCODE, AND DETERMINES MAXA
C       FROM KHT.
C
        SUBROUTINE DETMAX(KHT,MAXA,MCODE,NE,NEQ,NKT)
        IMPLICIT REAL*8(A-H,O-Z)
        DIMENSION KHT(1),MAXA(1),MCODE(6,1)
C
        DO 10 I=1,NEQ
          KHT(I)=0
   10   CONTINUE
C
C       GENERATE KHT
C
        DO 30 I=1,NE
          J=1
          WHILE(MCODE(J,I).EQ.0) DO
            J=J+1
          END WHILE
          MIN=MCODE(J,I)
          J=J+1
          DO 20 L=J,6
            K=MCODE(L,I)
            IF(K.NE.0) THEN DO
              KHT(K)=MAXO(KHT(K),(K-MIN))
            END IF
   20     CONTINUE
   30   CONTINUE
        PRINT 100
  100   FORMAT(//'-',11X,'I',10X,'KHT(I)',10X,'MAXA(I)')
C
C       GENERATE MAXA
C
        MAXA(1)=1
        DO 40 I=1,NEQ
          PRINT 200,I,KHT(I),MAXA(I)
  200     FORMAT('0',7X,I5,9X,I5,11X,I5)
```

```
        MAXA(I+1)=MAXA(I)+KHT(I)+1
   40 CONTINUE
      NKT=MAXA(NEQ+1)-1
      I=NEQ+1
      PRINT 300,I,MAXA(I),NKT
  300 FORMAT('0',7X,I5,25X,I5//7X,'NKT = ',I5)
      RETURN
      END
C
C     ******************************************************
C     *                                                    *
C     *                        PROP                        *
C     *                                                    *
C     ******************************************************
C
C     PROP READS JOINT COORDINATES AND ELEMENT PROPERTIES.
C     ELEMENT LENGTHS AND DIRECTION COSINES ARE GENERATED.
C
      SUBROUTINE PROP(AREA,C1,C2,ELENG,EMOD,MINC,X,ZI,NE,NJ,CI1,CI2)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION AREA(1),C1(1),C2(1),ELENG(1),EMOD(1),MINC(2,1),X(2,1),ZI
     1(1),CI1(1),CI2(1)
C
      READ,(X(1,J),X(2,J),J=1,NJ)
C
      PRINT 100
  100 FORMAT('-',7X,'GLOBAL JOINT COORDINATES')
      DO 10 I=1,2
        PRINT 200,I,(X(I,J),J=1,NJ)
  200   FORMAT('0',I1,'-COORD:',10(2X,F10.4))
   10 CONTINUE
      PRINT 300
  300 FORMAT('-',7X,'ELEMENT PROPERTIES'//9X,'NO.',9X,'AREA',10X,'EMOD',
     110X,'ZI',12X,'ELENG')
C
      DO 20 I=1,NE
        J=MINC(1,I)
        K=MINC(2,I)
        EL1=X(1,K)-X(1,J)
        EL2=X(2,K)-X(2,J)
        ELENG(I)=DSQRT(EL1**2+EL2**2)
        C1(I)=EL1/ELENG(I)
        C2(I)=EL2/ELENG(I)
        CI1(I)=C1(I)
        CI2(I)=C2(I)
C
        READ,AREA(I),EMOD(I),ZI(I)
        PRINT 400,I,AREA(I),EMOD(I),ZI(I),ELENG(I)
  400   FORMAT('0',6X,I4,7X,F8.3,2(2X,D15.7),4X,D15.7)
   20 CONTINUE
      RETURN
      END
C
C     ******************************************************
C     *                                                    *
```

```
C     *                          LOAD                                 *
C     *                                                               *
C     *****************************************************************
C
C     LOAD INITIALIZES Q AND FF ARRAYS TO ZERO AND CALLS JLOAD
C     AND MACT.
C
      SUBROUTINE LOAD(A1,A2,C1,C2,DIST,ELENG,EMOD,FF,FL,JCODE,MAT,MCODE,
     1MN,Q,QT,ZI,MXNA,NAT,NE,NEQ,QI,DIST2)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A1(1),A2(1),C1(1),C2(1),DIST(1),ELENG(1),EMOD(1),FF(6,1)
     1,FL(6,1),JCODE(3,1),MAT(1),MCODE(6,1),MN(1),Q(1),QT(1),ZI(1),DIST2
     2(1)
C
      DO 10 K=1,NEQ
        Q(K)=0.D00
   10 CONTINUE
      DO 30 I=1,NE
        DO 20 L=1,6
          FF(L,I)=0.D00
   20   CONTINUE
   30 CONTINUE
C
      CALL JLOAD(JCODE,Q)
      CALL MACT(A1,A2,C1,C2,DIST,ELENG,EMOD,FF,FL,MAT,MCODE,MN,Q,QT,ZI,M
     1XNA,NAT,NE,NEQ,QI,DIST2)
C
      RETURN
      END
C
C     *****************************************************************
C     *                                                               *
C     *                          JLOAD                                *
C     *                                                               *
C     *****************************************************************
C
C     JLOAD READS AND STORES THE JOINT FORCE VECTOR.
C
      SUBROUTINE JLOAD(JCODE,Q)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION JCODE(3,1),Q(1),QT(1)
C
      READ,JNUM,JDIR,FORCE
      IF(JNUM.NE.0) THEN DO
        PRINT 100
  100   FORMAT('-',8X,'JOINT NUMBER',10X,'GLOBAL DIRECTION',10X,'APPLIED
     1 FORCE')
      END IF
      WHILE(JNUM.NE.0) DO
        PRINT 200,JNUM,JDIR,FORCE
  200   FORMAT('0',11X,I4,22X,I2,13X,F16.5)
        K=JCODE(JDIR,JNUM)
        Q(K)=FORCE
        READ,JNUM,JDIR,FORCE
      END WHILE
```

```
      RETURN
      END
C
C     ********************************************************
C     *                                                      *
C     *                        MACT                          *
C     *                                                      *
C     ********************************************************
C
C     MACT READS AND STORES THE ELEMENT ACTION DATA.
C
      SUBROUTINE MACT(A1,A2,C1,C2,DIST,ELENG,EMOD,FF,FL,MAT,MCODE,MN,Q,Q
     1T,ZI,MXNA,NAT,NE,NEQ,QI,DIST2)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A1(1),A2(1),C1(1),C2(1),DIST(1),ELENG(1),EMOD(1),FF(6,1)
     1,FL(6,1),MAT(1),MCODE(6,1),MN(1),Q(1),QT(1),ZI(1),DIST2(1)
C
      I=1
      READ,MN(I),MAT(I),A1(I),A2(I),DIST(I),DIST2(I)
C
      IF(MN(I).NE.0) THEN DO
        PRINT 100
  100   FORMAT('-MEMBER NO.  ACTION TYPE',11X,'A1',16X,'A2',13X,'DISTANC
     1E')
      END IF
C
      WHILE(MN(I).NE.0) DO
        PRINT 200,MN(I),MAT(I),A1(I),A2(I),DIST(I),DIST2(I)
  200   FORMAT('0',2X,I4,8X,I4,5X,4(2X,F16.5))
        IF(I.GT.MXNA) THEN DO
          PRINT 300
  300     FORMAT('-*** NUMBER OF MEMBER ACTIONS EXCEDES MXNA; REDIMENSIO
     1N ARRAYS ***')
          STOP
        END IF
C
        I=I+1
        READ,MN(I),MAT(I),A1(I),A2(I),DIST(I),DIST2(I)
      END WHILE
C
      NAT=I-1
      RETURN
      END
C
C     ********************************************************
C     *                                                      *
C     *                      NEWRAP                          *
C     *                                                      *
C     ********************************************************
C
C     NEWRAP PERFORMS NEWTON-RAPHSON ITERATION OR MODIFIED
C     NEWTON-RAPHSON ITERATION. STIFF, SOLVE, FORCES, TEST,
C     RESULT, AND UPDATE ARE CALLED.
C
      SUBROUTINE NEWRAP(AREA,BP1,BP2,C1,C2,D,DD,DD1,DE,ELENG,EMOD,F,FG,F
```

```
     1L,FP,FPI,G,JCODE,MAXA,MCODE,QT,R1,R2,SKT,SP1,SP2,ST1,ST2,ZI,CPDB,C
     2PDC,CPE,CPF,DP,DQQR,ICI,IELS,IMAX,ITMAX,NE,NEQ,NJ,NKT,NUPD,A1,A2,D
     3IST,FF,MAT,MN,Q,NAT,QI,CI1,CI2,FQR,IN,QIMAX,DQI,DJ,MINC,P,ITIND,RM
     412,QL,U,INDTAN,KT,IMP,NPRINT,DIST2,QJ)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION AREA(1),BP1(1),BP2(1),C1(1),C2(1),D(1),DD(1),DD1(1),DE(1
     1),ELENG(1),EMOD(1),F(1),FG(6,1),FL(6,1),FP(1),FPI(1),G(1),JCODE(3,
     21),MAXA(1),MCODE(6,1),MN(1),Q(1),QT(1),R1(1),R2(1),SKT(1),SP1(1),S
     3P2(1),ST1(1),ST2(1),ZI(1),A1(1),A2(1),DIST(1),FF(6,1),MAT(1),CI1(1
     4),CI2(1),DJ(3,1),MINC(2,1),P(3,1),RM12(1),QL(1),U(1),KT(1),DIST2(1
     5),QJ(1)
C
      CALL FORCES(AREA,BP1,BP2,C1,C2,D,DE,ELENG,EMOD,F,FG,FL,MCODE,QT,R1
     1,R2,SP1,SP2,ST1,ST2,ZI,DQQR,IMAX,NE,NEQ,A1,A2,DIST,FF,MAT,MN,Q,NAT
     2,QI,CI1,CI2,FQR,IN,RM12,QL,U,INDTAN,NPRINT,DIST2,QJ)
C
      CALL UPDATE(QT,Q,NEQ,QI)
C
      WHILE(QI.LE.QIMAX) DO
        NC=NUPD
        ITCT=0
        ICI=1
C
        WHILE(ICI.NE.0.AND.ITCT.LE.ITMAX) DO
C
          IF(NC.GE.NUPD) THEN DO
            CALL STIFF(AREA,BP1,BP2,C1,C2,ELENG,EMOD,G,MAXA,MCODE,R1,R2,
     1      SKT,SP1,SP2,ST1,ST2,ZI,IELS,NE,NKT,RM12,QL,U,INDTAN,KT,NPRIN
     2      T)
            NC=0
          END IF
C
          CALL SOLVE(DD,F,MAXA,QT,SKT,NC,NEQ,NEGPIV,NKT,NPRINT)
          DO 10 I=1,NEQ
            D(I)=D(I)+DD(I)
   10     CONTINUE
C
          IF(ITCT.EQ.0) THEN DO
            DO 20 I=1,NEQ
              DD1(I)=DD(I)
   20       CONTINUE
          END IF
C
      CALL FORCES(AREA,BP1,BP2,C1,C2,D,DE,ELENG,EMOD,F,FG,FL,MCODE,QT,R1
     1,R2,SP1,SP2,ST1,ST2,ZI,DQQR,IMAX,NE,NEQ,A1,A2,DIST,FF,MAT,MN,Q,NAT
     2,QI,CI1,CI2,FQR,IN,RM12,QL,U,INDTAN,NPRINT,DIST2,QJ)
        CALL TEST(AREA,BP1,BP2,C1,C2,D,DD,DD1,DE,ELENG,EMOD,F,FG,FL,FP,F
     1PI,JCODE,MCODE,QT,R1,R2,SP1,SP2,ST1,ST2,ZI,CPDB,CPDC,CPE,CPF,DP,DQ
     2QR,ICI,IMAX,NE,NEQ,NJ,A1,A2,DIST,FF,MAT,MN,Q,NAT,QI,CI1,CI2,FQR,IN
     3,ITIND,NPRINT)
C
      IF(NPRINT.EQ.3) THEN DO
            PRINT 100,ITCT
  100     FORMAT('-',7X,'NEWTON-RAPHSON ITERATION ',I5//10X,'QT',20X,'F'
     1    ,20X,'D')
```

```
            DO 30 I=1,NEQ
              PRINT 200,QT(I),F(I),D(I)
  200         FORMAT('0',3(5X,D15.7))
   30       CONTINUE
        END IF
            NC=NC+1
            ITCT=ITCT+1
            DO 40 I=1,NEQ
              FPI(I)=F(I)
   40       CONTINUE
          END WHILE
          IF(NPRINT.EQ.3) THEN DO
            PRINT 300,ICI,ITCT
  300       FORMAT('0ICI=',I4,5X,'ITCT=',I4)
          END IF
          DO 50 I=1,NEQ
            FP(I)=F(I)
   50     CONTINUE
          IN=0
C
          CALL RESULT(A1,A2,C1,C2,D,DIST,DJ,ELENG,EMOD,FF,FG,FL,JCODE,MAT,
     1MCODE,MINC,MN,P,Q,QT,ZI,NAT,NE,NEQ,NJ,QI,IMP,NPRINT)
C
          QI=QI+DQI
C
          IF(ICI.NE.0) THEN DO
            PRINT 400
  400       FORMAT(' *** LAST SOLUTION IS NOT CONVERGED ***')
            STOP
          END IF
          CALL UPDATE(QT,Q,NEQ,QI)
        END WHILE
        RETURN
        END
C
C       ********************************************************
C       *                                                      *
C       *                      RIKWEM                          *
C       *                                                      *
C       ********************************************************
C
C       RIKWEM PERFORMS MODIFIED RIKS/WEMPNER ITERATION ON A NORMAL
C       STIFF, SOLVE, FORCES, TEST, RESULT, UPDATE, AND DOTPRD ARE
C       CALLED
C
        SUBROUTINE RIKWEM(AREA,BP1,BP2,C1,C2,D,DD,DD1,DE,ELENG,EMOD,F,FG,F
       1L,FP,FPI,G,JCODE,MAXA,MCODE,QT,R1,R2,SKT,SP1,SP2,ST1,ST2,ZI,CPDB,C
       2PDC,CPE,CPF,DP,DQQR,ICI,IELS,IMAX,ITMAX,NE,NEQ,NJ,NKT,NUPD,A1,A2,D
       3IST,FF,MAT,MN,Q,NAT,QI,CI1,CI2,FQR,IN,DD2,Z,SIGN,ITDES,QIMAX,DQI,D
       4J,MINC,P,DOT1,DOT2,ITIND,DD0,DD01,RM12,QL,U,INDTAN,KT,IMP,NPRINT,D
       5IST2,QJ)
        IMPLICIT REAL*8(A-H,O-Z)
        DIMENSION AREA(1),BP1(1),BP2(1),C1(1),C2(1),D(1),DD(1),DD1(1),DE(1
       1),ELENG(1),EMOD(1),F(1),FG(6,1),FL(6,1),FP(1),FPI(1),G(1),JCODE(3,
       21),MAXA(1),MCODE(6,1),MN(1),Q(1),QT(1),R1(1),R2(1),SKT(1),SP1(1),S
```

```
     3P2(1),ST1(1),ST2(1),ZI(1),A1(1),A2(1),DIST(1),FF(6,1),MAT(1),CI1(1
     4),CI2(1),DD2(1),Z(1),SIGN(1),DJ(3,1),MINC(2,1),P(3,1),DOT1(1),DOT2
     5(1),DD0(1),DD01(1),RM12(1),QL(1),U(1),KT(1),DIST2(1),QJ(1)
C
      ITCT=0
      NEGPIV=0
      CALL FORCES(AREA,BP1,BP2,C1,C2,D,DE,ELENG,EMOD,F,FG,FL,MCODE,QT,R1
     1,R2,SP1,SP2,ST1,ST2,ZI,DQQR,IMAX,NE,NEQ,A1,A2,DIST,FF,MAT,MN,Q,NAT
     2,QI,CI1,CI2,FQR,IN,RM12,QL,U,INDTAN,NPRINT,DIST2,QJ)
C
      WHILE(QI.LE.QIMAX.AND.ITCT.LE.ITMAX) DO
        CALL STIFF(AREA,BP1,BP2,C1,C2,ELENG,EMOD,G,MAXA,MCODE,R1,R2,SKT,
     1  SP1,SP2,ST1,ST2,ZI,IELS,NE,NKT,RM12,QL,U,INDTAN,KT,NPRINT)
        CALL SOLVE(DD01,Z,MAXA,Q,SKT,0,NEQ,NEGPIV,NKT,NPRINT)
C
C     COMPUTE DS FOR THE FIRST STEP OF THE FIRST ITERATION,
C     COMPUTE DQI FOR ALL REMAINING ITERATIONS.
C
        IF(ITCT.EQ.0) THEN DO
          DS=DQI*DSQRT(DOTPRD(DD01,DD01,NEQ)+1.D0)
        ELSE DO
          DQI=SIGN(NEGPIV+1)*DS/DSQRT(DOTPRD(DD01,DD01,NEQ)+1.D0)
          IF(NPRINT.EQ.3) THEN DO
            PRINT 250,SIGN(NEGPIV+1),NEGPIV,DS,DQI
  250       FORMAT(' SIGN=',D15.7,'  NEGPIV=',I4/' DS=',D15.7,'  DQI=',D
     1      15.7)
          END IF
        END IF
C
C     SAVE THE VALUES FOR THE FIRST TRIAL CONFIGURATION;
C     UPDATE THE CONFIGURATION.
C
        DQI1=DQI
        DO 10.I=1,NEQ
          DD0(I)=DQI*DD01(I)
          D(I)=D(I)+DD0(I)
   10   CONTINUE
        QI=QI+DQI
C
        IF(NPRINT.EQ.1) THEN DO
          PRINT 400,QI,D(IMP)
  400     FORMAT(' ',20X,F13.7/' ',20X,F14.8)
        END IF
        IF(NPRINT.EQ.3) THEN DO
          PRINT 500,QI,D(IMP)
  500     FORMAT(' QI INIT = ',F13.7,'  D INIT = ',F14.8)
        END IF
C
      CALL FORCES(AREA,BP1,BP2,C1,C2,D,DE,ELENG,EMOD,F,FG,FL,MCODE,QT,R1
     1,R2,SP1,SP2,ST1,ST2,ZI,DQQR,IMAX,NE,NEQ,A1,A2,DIST,FF,MAT,MN,Q,NAT
     2,QI,CI1,CI2,FQR,IN,RM12,QL,U,INDTAN,NPRINT,DIST2,QJ)
        ICI=1
        NC=NUPD
        IT=0
C
```

```
C      ITERATE ALONG THE NORMAL
C
         WHILE(ICI.NE.0.AND.IT.LE.IMAX) DO
           CALL UPDATE(QT,Q,NEQ,QI)
C
           IF(NC.GE.NUPD) THEN DO
             CALL STIFF(AREA,BP1,BP2,C1,C2,ELENG,EMOD,G,MAXA,MCODE,R1,R2,
     1       SKT,SP1,SP2,ST1,ST2,ZI,IELS,NE,NKT,RM12,QL,U,INDTAN,KT,NPRIN
     2       T)
             NC=0
           END IF
C
           CALL SOLVE(DD1,Z,MAXA,Q,SKT,NC,NEQ,NEGPIV,NKT,NPRINT)
           CALL SOLVE(DD2,F,MAXA,QT,SKT,1,NEQ,NEGPIV,NKT,NPRINT)
C
C      APPLY THE CONSTRAINT EQUATION
C
           DQI=-(DOTPRD(DD0,DD2,NEQ))/(DOTPRD(DD0,DD1,NEQ)+DQI1)
C
C      UPDATE THE CONFIGURATION
C
           DO 20 I=1,NEQ
             DD(I)=DQI*DD1(I)+DD2(I)
             D(I)=D(I)+DD(I)
    20     CONTINUE
        CALL FORCES(AREA,BP1,BP2,C1,C2,D,DE,ELENG,EMOD,F,FG,FL,MCODE,QT,R1
       1,R2,SP1,SP2,ST1,ST2,ZI,DQQR,IMAX,NE,NEQ,A1,A2,DIST,FF,MAT,MN,Q,NAT
       2,QI,CI1,CI2,FQR,IN,RM12,QL,U,INDTAN,NPRINT,DIST2,QJ)
           QI=QI+DQI
C
          CALL TEST(AREA,BP1,BP2,C1,C2,D,DD,DD1,DE,ELENG,EMOD,F,FG,FL,FP,F
       1PI,JCODE,MCODE,QT,R1,R2,SP1,SP2,ST1,ST2,ZI,CPDB,CPDC,CPE,CPF,DP,DQ
       2QR,ICI,IMAX,NE,NEQ,NJ,A1,A2,DIST,FF,MAT,MN,Q,NAT,QI,CI1,CI2,FQR,IN
       3,ITIND,NPRINT)
           NC=NC+1
           IT=IT+1
C
           IF(NPRINT.EQ.3) THEN DO
             PRINT 600,IT,QI,D(IMP),ICI
   600       FORMAT(' ITERATION ',I3,' QI=',F13.7,'   D= ',F13.7,'  ICI='
     1       ,I5)
           END IF
         END WHILE
         ITCT=ITCT+1
         DO 50 I=1,NEQ
           FP(I)=F(I)
    50   CONTINUE
         IF(NPRINT.EQ.3) THEN DO
           PRINT 700,IT
   700     FORMAT(' ',I5,' ITERATIONS')
         END IF
C
          CALL RESULT(A1,A2,C1,C2,D,DIST,DJ,ELENG,EMOD,FF,FG,FL,JCODE,MAT,
       1MCODE,MINC,MN,P,Q,QT,ZI,NAT,NE,NEQ,NJ,QI,IMP,NPRINT)
C
```

```
        IF(ICI.NE.0) THEN DO
          PRINT 800
  800     FORMAT(' *** LAST SOLUTION IS NOT CONVERGED ***')
          STOP
        END IF
C
C     SCALE DS TO CONTROL NUMBER OF ITERATIONS BETWEEN
C     EQUILIBRIUM POINTS
C
        DS=DS*DSQRT(DFLOAT(ITDES)/DFLOAT(IT))
C
        IF(NPRINT.EQ.3) THEN DO
          PRINT 350,DS
  350     FORMAT(' DS= ',F14.8,'    ***')
        END IF
      END WHILE
      RETURN
      END
C
C     ********************************************************
C     *                                                      *
C     *                      STIFF                           *
C     *                                                      *
C     ********************************************************
C
C     STIFF INITIALIZES THE TANGENT STIFFNESS MATRIX TO ZERO
C     AND CALLS ONE OF THE ELEMS SUBROUTINES AND ASSEMS FOR EACH
C     ELEMENT.
C
      SUBROUTINE STIFF(AREA,BP1,BP2,C1,C2,ELENG,EMOD,G,MAXA,MCODE,R1,R2,
     1SKT,SP1,SP2,ST1,ST2,ZI,IELS,NE,NKT,RM12,QL,U,INDTAN,KT,NPRINT)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION AREA(1),BP1(1),BP2(1),C1(1),C2(1),ELENG(1),EMOD(1),G(1),
     1MAXA(1),MCODE(6,1),R1(1),R2(1),SKT(1),SP1(1),SP2(1),ST1(1),ST2(1),
     2ZI(1),RM12(1),QL(1),U(1),KT(1)

      IF(NPRINT.EQ.3) THEN DO
        PRINT 100
  100   FORMAT(' STIFF CALLED')
      END IF
C
      DO 10 I=1,NKT
        SKT(I)=0.D00
   10 CONTINUE
C
      I=1
      WHILE(I.LE.NE) DO
C
        IF(INDTAN.EQ.1) THEN DO
          CALL ELEMS1(AREA,BP1,BP2,C1,C2,ELENG,EMOD,G,R1,R2,SP1,SP2,ST1,
     1    ST2,ZI,I,IELS,NE,RM12,QL,NPRINT,KT)
        END IF
C
        IF(INDTAN.EQ.2) THEN DO
          CALL ELEMS2(AREA,C1,C2,ELENG,EMOD,G,R1,R2,U,I,ZI,KT,NPRINT)
```

```
        END IF
C
        IF(INDTAN.EQ.3) THEN DO
          CALL ELEMS3(AREA,C1,C2,ELENG,EMOD,G,R1,R2,U,I,ZI,KT,NPRINT,QL)
        END IF
C
        CALL ASSEMS(G,MAXA,MCODE,SKT,I,NKT,NPRINT)
        I=I+1
      END WHILE
C
      IF(NPRINT.EQ.3) THEN DO
        PRINT 200
  200   FORMAT('0',7X,'STIFFNESS MATRIX')
        PRINT 300,(SKT(K),K=1,NKT)
  300   FORMAT(' ',6(5X,D15.7))
      END IF
      RETURN
      END
C
C     ************************************************************
C     *                                                          *
C     *                        ELEMS1                            *
C     *                                                          *
C     ************************************************************
C
C     ELEMS1 COMPUTES THE COMPONENTS OF THE SYSTEM TANGENT STIFFNESS
C     MATRIX FOR THE BEAM-COLUMN MODEL.
C
      SUBROUTINE ELEMS1(AREA,BP1,BP2,C1,C2,ELENG,EMOD,G,R1,R2,SP1,SP2,ST
     11,ST2,ZI,I,IELS,NE,RM12,QL,NPRINT,KT)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION AREA(1),BP1(1),BP2(1),C1(1),C2(1),ELENG(1),EMOD(1),G(1),
     1R1(1),R2(1),SP1(1),SP2(1),ST1(1),ST2(1),ZI(1),RM12(1),QL(1),KT(1)
      PI=3.141592653589793
C
C     COMPUTE VALUES FOR UNDEFORMED CONFIGURATION
C
      IF(IELS.EQ.0) THEN DO
        ST1(I)=4.D00
        ST2(I)=2.D00
        G1=0.D00
        G2=0.D00
        H=(PI**2*ZI(I))/(AREA(I)*ELENG(I)**2)
        IF(I.EQ.NE) THEN DO
          IELS=1
        END IF
C
C     COMPUTE SUBSEQUENT VALUES
C
      ELSE DO
        G1=SP1(I)*R1(I)+SP2(I)*R2(I)
        G2=SP2(I)*R1(I)+SP1(I)*R2(I)
        H=(PI**2*ZI(I))/(AREA(I)*ELENG(I)**2)+BP1(I)*(R1(I)+R2(I))**2+BP
     12(I)*(R1(I)-R2(I))**2
      END IF
```

```
C
      GP1=PI**2/(H*ELENG(I)**2)
      GP2=(G1+G2)/(H*ELENG(I)**2)
      GP3=G1/(H*ELENG(I))
      GP4=G2/(H*ELENG(I))
      GP5=2.D00*(ST1(I)+ST2(I))/ELENG(I)**2+(G1**2+2.D00*G1*G2+G2**2)/(P
     1I**2*H*ELENG(I)**2)
      GP6=(ST1(I)+ST2(I))/ELENG(I)+(G1**2+G1*G2)/(PI**2*H*ELENG(I))
      GP7=(ST1(I)+ST2(I))/ELENG(I)+(G1*G2+G2**2)/(PI**2*H*ELENG(I))
      ALPHA=EMOD(I)*ZI(I)/ELENG(I)
C
C     COMPUTE CONTRIBUTIONS TO SKT
C
      G(1)=(C1(I)**2*GP1-2.D00*C1(I)*C2(I)*GP2+C2(I)**2*GP5)*ALPHA
      G(2)=((C1(I)**2-C2(I)**2)*GP2+C1(I)*C2(I)*(GP1-GP5))*ALPHA
      G(3)=(C1(I)**2*GP5+2.D00*C1(I)*C2(I)*GP2+C2(I)**2*GP1)*ALPHA
      G(4)=(C1(I)*GP3-C2(I)*GP6)*ALPHA
      G(5)=(C2(I)*GP3+C1(I)*GP6)*ALPHA
      G(6)=(ST1(I)+G1**2/(PI**2*H))*ALPHA
      G(7)=(C1(I)*GP4-C2(I)*GP7)*ALPHA
      G(8)=(C1(I)*GP7+C2(I)*GP4)*ALPHA
      G(9)=(ST2(I)+G1*G2/(PI**2*H))*ALPHA
      G(10)=(ST1(I)+G2**2/(PI**2*H))*ALPHA
C
C     COMPUTE INITIAL STRESS CONTRIBUTIONS TO SKT IF DESIRED
C
      IF(KT(2).GT.0) THEN DO
        G(1)=G(1)-2.D0*C1(I)*C2(I)*RM12(I)-QL(I)*C2(I)*C2(I)
        G(2)=G(2)+(C1(I)*C1(I)-C2(I)*C2(I))*RM12(I)+C1(I)*C2(I)*QL(I)
        G(3)=G(3)+2.D0*C1(I)*C2(I)*RM12(I)-C1(I)*C1(I)*QL(I)
      END IF
C
      IF(NPRINT.EQ.3) THEN DO
        PRINT 100,I,(G(KI),KI=1,10)
  100   FORMAT('-',7X,'G(10) FOR ELE.',I3/5(5X,D15.7)/5(5X,D15.7))
      END IF
      RETURN
      END
C
C     ********************************************************
C     *                                                      *
C     *                       ELEMS2                         *
C     *                                                      *
C     ********************************************************
C
C     ELEMS2 COMPUTES THE COMPONENTS OF THE SYSTEM TANGENT STIFFNESS
C     MATRIX FOR THE FINITE ELEMENT MODEL WHICH NEGLECTS THE
C     VARIATION OF THE COORDINATE TRANSFORMATION MATRIX.
C     THIS MODEL WAS DEVELOPED USING CONVECTED COORDINATES.
C     *** THIS IS NOT A GOOD REPRESENTATION OF SKT FOR ALL CASES ***
C
      SUBROUTINE ELEMS2(AREA,C1,C2,ELENG,EMOD,G,R1,R2,U,I,ZI,KT,NPRINT)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION AREA(1),C1(1),C2(1),ELENG(1),EMOD(1),G(1),R1(1),R2(1),U(
     11),ZI(1),KT(1)
```

```
C
      A=AREA(I)
      E=EMOD(I)
      ELI=ELENG(I)
      Z=ZI(I)
      E1=U(I)
      E2=R1(I)
      E3=R2(I)
      C=C1(I)
      S=C2(I)
C
      T1=T2=T3=T4=T5=T6=0.D0
C
      IF(KT(1).GT.0) THEN DO
        T1=T1+E*A/ELI
        T4=T4+4.D0*E*Z/ELI
        T5=T5+2.D0*E*Z/ELI
        T6=T6+4.D0*E*Z/ELI
      END IF
C
      IF(KT(2).GT.0) THEN DO
        T2=T2+E*A/30.D0*(4.D0*E2-E3)
        T3=T3+E*A/30.D0*(4.D0*E3-E2)
      END IF
C
      IF(KT(3).GT.0) THEN DO
        T4=T4+E*A*ELI/210.D0*(12.D0*E2**2-3.D0*E2*E3+E3**2)
        T5=T5+E*A*ELI/420.D0*(-3.D0*E2**2+4.D0*E2*E3-3.D0*E3**2)
        T6=T6+E*A*ELI/210.D0*(E2**2-3.D0*E2*E3+12.D0*E3**2)
      END IF
C
      IF(KT(4).GT.0) THEN DO
        T4=T4+2.D0*E*A*E1/15.D0
        T5=T5-E*A*E1/30.D0
        T6=T6+2.D0*E*A*E1/15.D0
      END IF
C
      IF(KT(5).GT.0) THEN DO
        T4=T4+E*A*ELI/420.D0*(12.D0*E2**2-3.D0*E2*E3+E3**2)
        T5=T5+E*A*ELI/840.D0*(-3.D0*E2**2+4.D0*E2*E3-3.D0*E3**2)
        T6=T6+E*A*ELI/420.D0*(E2**2-3.D0*E2*E3+12.D0*E3**2)
      END IF
C
C     COMPUTE THE GLOBAL STIFFNESS CONTRIBUTIONS
C
        G(1)=C*C*T1+2.D0*C*S*(T2+T3)/ELI+S*S*(T4+2.D0*T5+T6)/ELI**2
        G(2)=C*S*T1-(C*C-S*S)*(T2+T3)/ELI-C*S*(T4+2.D0*T5+T6)/ELI**2
        G(3)=S*S*T1-2.D0*C*S*(T2+T3)/ELI+C*C*(T4+2.D0*T5+T6)/ELI**2
        G(4)=-C*T2-S*(T4+T5)/ELI
        G(5)=-S*T2+C*(T4+T5)/ELI
        G(6)=T4
        G(7)=-C*T3-S*(T5+T6)/ELI
        G(8)=-S*T3+C*(T5+T6)/ELI
        G(9)=T5
        G(10)=T6
```

```
C
      IF(NPRINT.EQ.3) THEN DO
        PRINT 100,I,(G(KI),KI=1,10)
  100   FORMAT('-',7X,'G(10) FOR ELE.',I3/5(5X,D15.7)/5(5X,D15.7))
      END IF
      RETURN
      END
C
C     ********************************************************
C     *                                                      *
C     *                        ELEMS3                        *
C     *                                                      *
C     ********************************************************
C
C     ELEMS3 COMPUTES THE COMPONENTS OF THE SYSTEM TANGENT STIFFNESS
C     MATRIX FOR THE FINITE ELEMENT MODEL WHICH IS DEVELOPED
C     FOR SIX LOCAL D.O.F. (NOT USING CONVECTED COORDINATES)
C
C    ELEMS3
C
      SUBROUTINE ELEMS3(AREA,C1,C2,ELENG,EMOD,G,R1,R2,U,I,ZI,KT,NPRINT,Q
     1L)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION AREA(1),C1(1),C2(1),ELENG(1),EMOD(1),G(1),R1(1),R2(1),U(
     11),ZI(1),KT(1),QL(1)
C
      A=AREA(I)
      E=EMOD(I)
      EL=ELENG(I)
      Z=ZI(I)
      E1=U(I)
      E2=R1(I)
      E3=R2(I)
      C=C1(I)
      S=C2(I)
C
      T1=T2=T3=T4=T5=0.D0
C
C     COMPUTE THE CONTRIBUTION OF THE CONVENTIONAL LINEAR, STATIC
C     STIFFNESS MATRIX.
C
      IF(KT(1).GT.0) THEN DO
        T1=T1+E*A/EL
        T2=T2+12.D0*E*Z/EL**3
        T3=T3+6.D0*E*Z/EL**2
        T4=T4+4.D0*E*Z/EL
        T5=T5+2.D0*E*Z/EL
      END IF
C
C     COMPUTE THE CONTRIBUTIONS OF THE INITIAL STRESS STIFFNESS MATRIX
C
      IF(KT(2).GT.0) THEN DO
        T2=T2+6.D0*QL(I)/(5.D0*EL)
        T3=T3+QL(I)/10.D0
        T4=T4+2.D0*QL(I)*EL/15.D0
```

```
        T5=T5-QL(I)*EL/30.D0
      END IF
C
C     COMPUTE THE GLOBAL STIFFNESS COEFFICIENTS
C
      G(1)=C*C*T1+S*S*T2
      G(2)=C*S*(T1-T2)
      G(3)=S*S*T1+C*C*T2
      G(4)=-S*T3
      G(5)=C*T3
      G(6)=T4
      G(7)=-S*T3
      G(8)=C*T3
      G(9)=T5
      G(10)=T4
      RETURN
      END
C
C     ********************************************************
C     *                                                      *
C     *                        ASSEMS                        *
C     *                                                      *
C     ********************************************************
C
C     ASSEMS ASSEMBLES THE CONTRIBUTIONS FROM EACH ELEMENT
C     INTO THE GLOBAL SYSTEM STIFFNESS MATRIX.
C
      SUBROUTINE ASSEMS(G,MAXA,MCODE,SKT,I,NKT,NPRINT)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION G(1),MAXA(1),MCODE(6,1),SKT(1)
      INTEGER INDEX(6,6)/1,2,4,-1,-2,7,2,3,5,-2,-3,8,4,5,6,-4,-5,9,-1,-2
     1,-4,1,2,-7,-2,-3,-5,2,3,-8,7,8,9,-7,-8,10/
C
      DO 20 JE=1,6
        J=MCODE(JE,I)
        IF(J.NE.0) THEN DO
          DO 10 NE=1,JE
            N=MCODE(NE,I)
            IF(N.NE.0) THEN DO
              K= MAXA(J)+J-N
              L=INDEX(NE,JE)
C
              IF(L.GT.0) THEN DO
                SKT(K)=SKT(K)+G(L)
              ELSE DO
                SKT(K)=SKT(K)-G(-L)
              END IF
C
            END IF
   10     CONTINUE
        END IF
   20 CONTINUE
C
      IF(NPRINT.EQ.3) THEN DO
        PRINT 100,I,(SKT(KI),KI=1,NKT)
```

```
  100   FORMAT('0',7X,'STIFFNESS FROM ELEMENT ',I3,3(/5(5X,D15.7)))
      END IF
      RETURN
      END
C
C     ******************************************************
C     *                                                    *
C     *                      FORCES                        *
C     *                                                    *
C     ******************************************************
C
C     FORCES CALLS ELEMF FOR EACH MEMBER AND CALLS FIXEND OR FIXEN2
C     IF THERE ARE MEMBER ACTIONS.
C
      SUBROUTINE FORCES(AREA,BP1,BP2,C1,C2,D,DE,ELENG,EMOD,F,FG,FL,MCODE
     1,QT,R1,R2,SP1,SP2,ST1,ST2,ZI,DQQR,IMAX,NE,NEQ,A1,A2,DIST,FF,MAT,MN
     2,Q,NAT,QI,CI1,CI2,FQR,IN,RM12,QL,U,INDTAN,NPRINT,DIST2,QJ)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION AREA(1),BP1(1),BP2(1),C1(1),C2(1),D(1),DE(1),ELENG(1),EM
     1OD(1),F(1),FG(6,1),FL(6,1),MCODE(6,1),QT(1),R1(1),R2(1),SP1(1),SP2
     2(1),ST1(1),ST2(1),ZI(1),A1(1),A2(1),DIST(1),FF(6,1),MAT(1),MN(1),Q
     3(1),CI1(1),CI2(1),RM12(1),QL(1),U(1),DIST2(1),QJ(1)
C
      DO 10 I=1,NEQ
        F(I)=0.D00
   10 CONTINUE
C
      I=1
      WHILE(I.LE.NE) DO
        IF(NPRINT.EQ.3) THEN DO
          PRINT 100,I
  100     FORMAT('0ELEMENT ',I3)
        END IF
C
        CALL ELEMF(AREA,BP1,BP2,C1,C2,D,DE,ELENG,EMOD,F,FG,FL,MCODE,R1,R
     12,SP1,SP2,ST1,ST2,ZI,DQQR,I,IMAX,CI1,CI2,FQR,IN,RM12,QL,U,INDTAN,N
     2PRINT)
C
        I=I+1
      END WHILE
C
      IF(NAT.NE.0) THEN DO
       IF(INDTAN.EQ.1) THEN DO
        CALL FIXEND(A1,A2,C1,C2,DIST,ELENG,EMOD,FF,FL,MAT,MCODE,MN,Q,F,Z
     1  I,NAT,NE,NEQ,QI,QJ,NPRINT)
       END IF
       IF(INDTAN.EQ.2.OR.INDTAN.EQ.3) THEN DO
        CALL FIXEN2(A1,A2,C1,C2,DIST,ELENG,EMOD,FF,FL,MAT,MCODE,MN,Q,F,Z
     1  I,NAT,NE,NEQ,QI,DIST2,QJ,NPRINT)
       END IF
      END IF
C
      RETURN
      END
C
```

```
C     ******************************************************
C     *                                                    *
C     *                       ELEMF                        *
C     *                                                    *
C     ******************************************************
C
C     ELEMF COMPUTES THE INTERNAL ELEMENT FORCES DUE TO THE
C     DEFORMATIONS AND TRANSFORMS THEM TO GLOBAL FORCES.
C     BOWCOR IS CALLED FOR THE BEAM-COLUMN MODEL.
C     ELEMF IS EXECUTED FOR EACH MEMBER, I.
C
      SUBROUTINE ELEMF(AREA,BP1,BP2,C1,C2,D,DE,ELENG,EMOD,F,FG,FL,MCODE,
     1R1,R2,SP1,SP2,ST1,ST2,ZI,DQQR,I,IMAX,CI1,CI2,FQR,IN,RM12,QL,U,INDT
     2AN,NPRINT)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION AREA(1),BP1(1),BP2(1),C1(1),C2(1),D(1),DE(1),ELENG(1),EM
     1OD(1),F(1),FG(6,1),FL(6,1),MCODE(6,1),R1(1),R2(1),SP1(1),SP2(1),ST
     21(1),ST2(1),ZI(1),CI1(1),CI2(1),RM12(1),QL(1),U(1)
      IF(NPRINT.EQ.3) THEN DO
        PRINT 100,I
  100   FORMAT(' DE FOR ELEMENT',I3)
      END IF
C
C     GENERATE THE DISPLACEMENTS FOR THE GLOBAL ELEMENT D.O.F.
C
      DO 10 J=1,6
        DE(J)=0.D00
        K=MCODE(J,I)
        IF(K.NE.0) THEN DO
          DE(J)=D(K)
        END IF
C
        IF(NPRINT.EQ.3) THEN DO
          PRINT 200,DE(J)
  200     FORMAT(5X,D15.7)
        END IF
   10 CONTINUE
C
      RL10=ELENG(I)*CI1(I)
      RL20=ELENG(I)*CI2(I)
C
C     UPDATE THE CONFIGURATION
C
      RL1=RL10+DE(4)-DE(1)
      RL2=RL20+DE(5)-DE(2)
      RL=DSQRT(RL1**2+RL2**2)
      C1(I)=RL1/RL
      C2(I)=RL2/RL
      B=DARCOS(RL1/RL)
      BO=DARCOS(RL10/ELENG(I))
C
      STRN=(RL-ELENG(I))/ELENG(I)
      RIGBR=B-BO
      IF(NPRINT.EQ.3) THEN DO
        PRINT 300,I,RL,STRN,B,BO,RIGBR
```

```
  300   FORMAT(' ELEMENT',I4,'  DEF. LENGTH=',D15.7,'  STRAIN=',D15.7/
     1  ' B=',D15.7,'  BO=',D15.7,'  RIGD BOD ROT=',D15.7)
       END IF
C
       IF(RL2.LT.0.D00) THEN DO
         B=-B
       END IF
       IF(RL20.LT.0.D00) THEN DO
         BO=-BO
       END IF
C
       UL=(ELENG(I)-RL)/ELENG(I)
       R1(I)=DE(3)+BO-B
       R2(I)=DE(6)+BO-B
C
       IF(INDTAN.EQ.1) THEN DO
C
         CALL BOWCOR(AREA,BP1,BP2,ELENG,R1,R2,SP1,SP2,ST1,ST2,ZI,CB,DQQR,
     1   I,IMAX,UL,FQR,EMOD,IN,NPRINT)
         RM1=EMOD(I)*ZI(I)*(ST1(I)*R1(I)+ST2(I)*R2(I))/ELENG(I)
         RM2=EMOD(I)*ZI(I)*(ST2(I)*R1(I)+ST1(I)*R2(I))/ELENG(I)
         RM12(I)=(RM1+RM2)/(ELENG(I)**2)
         FL(1,I)=EMOD(I)*AREA(I)*(UL-CB)
         QL(I)=FL(1,I)/ELENG(I)
C
       ELSE DO
C
         U(I)=RL-ELENG(I)
         E1=U(I)
         E2=R1(I)
         E3=R2(I)
         A=AREA(I)
         E=EMOD(I)
         EL=ELENG(I)
         Z=ZI(I)
         RM1=E*A*(4.D0*E2-E3)*E1/60.D0+2.D0*E*Z*(2.D0*E2+E3)/EL+E*A*E1*E2
     1   /20.D0+E*A*EL*((12.D0*E2*E2-3.D0*E2*E3+E3*E3)*E2/420.D0+(-3.D0*E
     2   2*E2+4.D0*E2*E3-3.D0*E3*E3)*E3/840.D0)
         RM2=E*A*(4.D0*E3-E2)*E1/60.D0+2.D0*E*Z*(2.D0*E3+E2)/EL+E*A*E1*E2
     1   /20.D0+E*A*EL*((12.D0*E3*E3-3.D0*E2*E3+E2*E2)*E3/420.D0+(-3.D0*E
     2   2*E2+4.D0*E2*E3-3.D0*E3*E3)*E2/840.D0)
         FL(1,I)=(-E*A)*(E1/EL+(4.D0*E2-E3)*E2/60.D0+(4.D0*E3-E2)*E3/60.D
     1   0)
         QL(I)=-FL(1,I)
C
       END IF
C
C      COMPUT REMAINING LOCAL ELEMENT FORCES
C
       FL(2,I)=(RM1+RM2)/ELENG(I)
       FL(3,I)=RM1
       FL(4,I)=-FL(1,I)
       FL(5,I)=-FL(2,I)
       FL(6,I)=RM2
C
```

```
      IF(NPRINT.EQ.3) THEN DO
      PRINT 400,I,RL,B,BO,UL,RM1,RM2,ST1(I),ST2(I),CB,R1(I),R2(I),RL10,R
     1L1,RL2,RL20
  400 FORMAT('0ELEMENT ',I3,'   RL= ',D15.7/'   B=',D15.7,'   BO=',D15.7
     1,'   UL=',D15.7/'   M1=',D15.7,'   M2=',D15.7/'  ST1=',D15.7,'  ST
     22=',D15.7/'   CB=',D15.7,'  R1=',D15.7,'  R2=',D15.7/'   RL10=',D1
     35.7,'  RL1=',D15.7/'   RL2=',D15.7,'  RL20=',D15.7)
      PRINT 500
  500 FORMAT('0 LOCAL ELEMENT FORCES')
      DO 20 J=1,6
        PRINT 600,FL(J,I)
  600   FORMAT('0',5X,D15.7)
   20 CONTINUE
      END IF
C
C     TRANSFORM LOCAL TO GLOGAL ELEMENT FORCES
C
      FG(1,I)=C1(I)*FL(1,I)-C2(I)*FL(2,I)
      FG(2,I)=C2(I)*FL(1,I)+C1(I)*FL(2,I)
      FG(3,I)=FL(3,I)
      FG(4,I)=C1(I)*FL(4,I)-C2(I)*FL(5,I)
      FG(5,I)=C2(I)*FL(4,I)+C1(I)*FL(5,I)
      FG(6,I)=FL(6,I)
C
C     ACCUMULATE THE ELEMENT FORCE VECTOR
C
      DO 30 J=1,6
        K=MCODE(J,I)
        IF(K.NE.0) THEN DO
          F(K)=F(K)+FG(J,I)
        END IF
   30 CONTINUE
      RETURN
      END
C
C     ************************************************************
C     *                                                          *
C     *                        BOWCOR                            *
C     *                                                          *
C     ************************************************************
C
C     BOWCOR USES A NEWTON METHOD TO FIND THE ROOT OF THE EQUATION
C     THAT DETERMINES QR, THE RATIO OF THE AXIAL FORCE TO THE
C     EULER BUCLING LOAD.
C
      SUBROUTINE BOWCOR(AREA,BP1,BP2,ELENG,R1,R2,SP1,SP2,ST1,ST2,ZI,CB,D
     1QQR,I,IMAX,UL,FQR,EMOD,IN,NPRINT)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION AREA(1),BP1(1),BP2(1),ELENG(1),R1(1),R2(1),SP1(1),SP2(1)
     1,ST1(1),ST2(1),ZI(1),EMOD(1)
      PI=3.141592653589793
C
C     THE INITIAL GUESS CORRESPONDS TO ZERO AXIAL DEFORMATION.
C
      QR=UL*AREA(I)*ELENG(I)**2/(ZI(I)*PI**2)
```

```
C
      IC=0
      WHILE(IC.LT.IMAX) DO
C
        IF(QR.GT.0.D00) THEN DO
          W=QR*PI**2
          W=DSQRT(W)
          ST1(I)=W*(DSIN(W)-W*DCOS(W))/(2.D00-2.D00*DCOS(W)-W*DSIN(W))
          ST2(I)=W*(W-DSIN(W))/(2.D00-2.D00*DCOS(W)-W*DSIN(W))
        END IF
C
        IF(QR.EQ.0.D00)THEN DO
          ST1(I)=4.D00
          ST2(I)=2.D00
          SP1(I)=0.D00
          SP2(I)=0.D00
          BP1(I)=0.D00
          BP2(I)=0.D00
          CB=UL
          RETURN
        END IF
C
        IF(QR.LT.0.D00) THEN DO
          W=-QR*PI**2
          W=DSQRT(W)
          ST1(I)=W*(W*DCOSH(W)-DSINH(W))/(2.D00-2.D00*DCOSH(W)+W*DSINH(W
     1          ))
          ST2(I)=W*(DSINH(W)-W)/(2.D00-2.D00*DCOSH(W)+W*DSINH(W))
        END IF
C
        BW1=(ST1(I)+ST2(I))*(ST2(I)-2.D00)/(8.D00*QR*PI**2)
        BW2=ST2(I)/(8.D00*(ST1(I)+ST2(I)))
        SP1(I)=-2.D00*PI**2*(BW1+BW2)
        SP2(I)=-2.D00*PI**2*(BW1-BW2)
        BP1(I)=(QR*(ST1(I)*SP2(I)+ST2(I)*SP1(I)+2.D0*ST2(I)*SP2(I)-2.D0*
     1  SP1(I)-2.D0*SP2(I))-(ST1(I)+ST2(I))*(ST2(I)-2.D0))/(8.D0*PI**2*Q
     2  R**2)
        BP2(I)=(SP2(I)*(ST1(I)+ST2(I))-ST2(I)*(SP1(I)+SP2(I)))/(8.D0*(ST
     1  1(I)+ST2(I))**2)
        CB=BW1*(R1(I)+R2(I))**2+BW2*(R1(I)-R2(I))**2
        CBP=BP1(I)*(R1(I)+R2(I))**2+BP2(I)*(R1(I)-R2(I))**2
        RLA=ELENG(I)/(DSQRT(ZI(I)/AREA(I)))
        FQRC=DABS(QR-RLA**2*(UL-CB)/PI**2)
        DQ=-(QR-RLA**2*(UL-CB)/PI**2)/(1.D0+RLA**2*CBP/PI**2)
C
      IF(NPRINT.EQ.3) THEN DO
        PRINT 100,ST1(I),ST2(I),BW1,BW2,SP1(I),SP2(I),BP1(I),BP2(I),CB,C
     1  BP,DQ
  100   FORMAT('  ST1=',D15.7,'  ST2=',D15.7,'  BW1=',D15.7,'  BW2=',D15
     1  .7/'  SP1=',D15.7,'  SP2=',D15.7,'  BP1=',D15.7,'  BP2=',D15.7/
     2  'CB=',D15.7,'  CBP=',D15.7,'  DQ=',D15.7)
      END IF
C
C     UPDATE QR
C
```

```
        QR=QR+DQ
C
        DQQRC=DABS(DQ/QR)
        IC=IC+1
        QPRINT=EMOD(I)*AREA(I)*(UL-CB)
        IF(NPRINT.EQ.3) THEN DO
          PRINT 200,IC,QPRINT
  200     FORMAT(' ITERATION ',I4,'   Q= ',D15.7)
        END IF
        IF(DQQRC.LE.DQQR.AND.FQRC.LE.FQR) THEN DO
          IF(NPRINT.EQ.3) THEN DO
            PRINT 300,IC,I,FQRC,DQQRC
  300       FORMAT(' BOWCOR CONVERGED IN ',I5,' ITERATIONS FOR ELEMENT'
     1      ,I5,'  FQRC=',D15.7,'  DQQRC=',D15.7)
          END IF
          IC=IMAX+1
        END IF
      END WHILE
      IF(DQQRC.GT.DQQR.OR.FQRC.GT.FQR) THEN DO
        PRINT 400,I
  400   FORMAT(' NO CONVERGENCE IN BOWCOR FOR ELEMENT ',I3)
        STOP
      END IF
      RETURN
      END
C
C     ********************************************************
C     *                                                      *
C     *                      FIXEND                          *
C     *                                                      *
C     ********************************************************
C
C     FIXEND COMPUTES THE FIXED END FORCES DUE TO MEMBER ACTIONS
C     FOR THE BEAM-COLUMN MODEL.
C
      SUBROUTINE FIXEND(A1,A2,C1,C2,DIST,ELENG,EMOD,FF,FL,MAT,MCODE,MN,Q
     1,F,ZI,NAT,NE,NEQ,QI,QJ,NPRINT)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A1(1),A2(1),C1(1),C2(1),DIST(1),ELENG(1),EMOD(1),FF(6,1)
     1,FL(6,1),MAT(1),MCODE(6,1),MN(1),Q(1),F(1),ZI(1),QJ(1)
C
C     INITIALIZE FF TO ZERO
C
      DO 20 J=1,NE
        DO 10 I=1,6
          FF(I,J)=0.D00
   10   CONTINUE
   20 CONTINUE
C
      IF(NAT.GT.0) THEN DO
C
C     FOR EACH ELEMENT ACTION
C
        DO 30 I=1,NAT
          J=MN(I)
```

```
            K=MAT(I)
            WA=A1(I)
            WB=A2(I)
            A=DIST(I)
            E=ELENG(J)
            EMA=E-A
C
C       FOR NO AXIAL FORCE IN MEMBER
C
            IF(FL(1,J).EQ.0.D0) THEN DO
              DO CASE K
C
C       CASE 1 : CONCENTRATED FORCE
C
              CASE
                FF(2,J)=FF(2,J)+WA*EMA**2*(E+2.D0*A)/E**3
                FF(3,J)=FF(3,J)+WA*A*EMA**2/E**2
                FF(5,J)=FF(5,J)+WA*A**2*(3.D0*E-2.D0*A)/E**3
                FF(6,J)=FF(6,J)-WA*A**2*EMA/E**2
C
C       CASE 2 : DISTRIBUTED LOAD
C
              CASE
                EMA3=EMA**3
                WMW=WB-WA
              FEF2=WA*EMA3*(E+A)/(2.D0*E**3)+WMW*EMA3*(3.D0*E+2.D0*A)/(20.
     1             D0*E**3)
              FEF3=WA*EMA3*(E+3.D0*A)/(12.D0*E**2)+WMW*EMA3*(2.D0*E+3.D0*A
     1             )/(60.D0*E**2)
                FF(2,J)=FF(2,J)+FEF2
                FF(3,J)=FF(3,J)+FEF3
                FF(5,J)=FF(5,J)+(WA+WB)*EMA/2.D0-FEF2
                FF(6,J)=FF(6,J)+FEF2*E-FEF3-WA*EMA**2/2.D0-WMW*EMA**2/6.D0
C
C       CASE 3 : CONCENTRATED MOMENT
C
              CASE
                FEF2=-6.D0*WA*A*EMA/E**3
                FF(2,J)=FF(2,J)+FEF2
                FF(3,J)=FF(3,J)+WA*(E**2-4.D0*A*E+3.D0*A**2)/E**2
                FF(5,J)=FF(5,J)-FEF2
                FF(6,J)=FF(6,J)+WA*(3.D0*A**2-2.D0*A*E)/E**2
              END CASE
            ELSE DO
C
C       COMPRESSIVE AXIAL FORCE
C
              IF(FL(1,J).GT.0.D0) THEN DO
                RS=DSQRT(FL(1,J)/(EMOD(J)*ZI(J)))
                RS2=DSIN(RS*E)
                RS3=1.D0-DCOS(RS*E)
                RS4=RS*E-RS2
                RA3=1.D0-DCOS(RS*EMA)
                RA4=RS*EMA-DSIN(RS*EMA)
                RA5=RS**2*EMA**2/2.D0-RA3
```

```
              RA6=RS**3*EMA**3/6.D0-RA4
            ELSE DO
C
C     TENSILE AXIAL FORCE
C
              RS=DSQRT(-FL(1,J)/(EMOD(J)*ZI(J)))
              RS2=DSINH(RS*E)
              RS3=DCOSH(RS*E)-1.D0
              RS4=RS2-RS*E
              RA3=DCOSH(RS*EMA)-1.D0
              RA4=DSINH(RS*EMA)-RS*EMA
              RA5=RA3-RS**2*EMA**2/2.D0
              RA6=RA4-RS**3*EMA**3/6.D0
            END IF
            DEN=RS3**2-RS2*RS4
            DO CASE K
C
C     CASE 1 : CONCENTRATED FORCE
C
            CASE
              FEF2=WA*(RS3*RA3-RS2*RA4)/DEN
              FEF3=(WA/RS)*(RS4*RA3-RS3*RA4)/DEN
              FF(2,J)=FF(2,J)+FEF2
              FF(3,J)=FF(3,J)+FEF3
              FF(5,J)=FF(5,J)+WA-FEF2
              FF(6,J)=FF(6,J)-FEF3+FEF2*E-WA*EMA
C
C     CASE 2 : DISTRIBUTED LOAD
C
            CASE
              FEF2=(WA/RS)*(RS3*RA4-RS2*RA5)/DEN+((WB-WA)/(RS**2*EMA))*(
     1             RS3*RA5-RS2*RA6)/DEN
              FEF3=(WA/RS**2)*(RS4*RA4-RS3*RA5)/DEN+((WB-WA)/(RS**3*EMA)
     1             )*(RS4*RA5-RS3*RA6)/DEN
              FF(2,J)=FF(2,J)+FEF2
              FF(3,J)=FF(3,J)+FEF3
              FF(5,J)=FF(5,J)+(WA+WB)*EMA/2.D0-FEF2
              FF(6,J)=FF(6,J)-FEF3+FEF2*E-WA*EMA**2/2.D0-(WB-WA)*EMA**2/
     1                6.D0
C
C     CASE 3 : CONCENTRATED MOMENT
C
            CASE
              FEF2=-WA*RS*(RS3*RA2-RS2*RA3)/DEN
              FEF3=WA*(RS3*RA3-RS4*RA2)/DEN
              FF(2,J)=FF(2,J)+FEF2
              FF(3,J)=FF(3,J)+FEF3
              FF(5,J)=FF(5,J)-FEF2
              FF(6,J)=FF(6,J)+FEF2*E-FEF3+WA
            END CASE
          END IF
   30   CONTINUE
        END IF
C
      CALL ASSEMF(C1,C2,FF,MCODE,Q,F,NE,NEQ,QI,QJ,NPRINT)
```

```
C
      RETURN
      END
C
C     ********************************************************
C     *                                                      *
C     *                        FIXEN2                        *
C     *                                                      *
C     ********************************************************
C
C     FIXEN2 COMPUTES THE FIXED END FORCES DUE TO MEMBER ACTIONS
C     FOR THE FINITE ELEMENT MODEL.
C
      SUBROUTINE FIXEN2(A1,A2,C1,C2,DIST,ELENG,EMOD,FF,FL,MAT,MCODE,MN,Q
     1,F,ZI,NAT,NE,NEQ,QI,DIST2,QJ,NPRINT)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A1(1),A2(1),C1(1),C2(1),DIST(1),ELENG(1),EMOD(1),FF(6,1)
     1,FL(6,1),MAT(1),MCODE(6,1),MN(1),Q(1),F(1),ZI(1),DIST2(1),QJ(1)
C
      DO 20 J=1,NE
        DO 10 I=1,6
          FF(I,J)=0.D00
   10   CONTINUE
   20 CONTINUE
C
      IF(NAT.GT.0) THEN DO
        DO 30 I=1,NAT
          J=MN(I)
          K=MAT(I)
          D1=DIST(I)
          D2=DIST2(I)
C         P1=A1(I)*QI
C         P2=A2(I)*QI
          P1=A1(I)
          P2=A2(I)
          DIV1=D1/ELENG(J)
          DIV2=D2/ELENG(J)
            DO CASE K
C
C   CASE 1   CONCENTRATED LOAD
C
      CASE
      FF(2,J)=FF(2,J)-P1*(1.+DIV1**2*(2.*DIV1-3.))
      FF(3,J)=FF(3,J)-P1*ELENG(J)*DIV1*(1.-DIV1)**2
      FF(5,J)=FF(5,J)+P1*DIV1**2*(2.*DIV1-3.)
      FF(6,J)=FF(6,J)+P1*ELENG(J)*DIV1**2*(1.-DIV1)
C
C   CASE 2   UNIFORM LOAD
C
      CASE
      D5P=(P1*ELENG(J)/24.)*(DIV2**3*(4.-DIV2)-DIV1**3*(4.-DIV1))
      D6P=(P1/6.)*(DIV2-DIV1)*(DIV2**2+DIV1**2+DIV1*DIV2)
      TEMF5=6.*ELENG(J)*D6P-12.*D5P
      TEMF6=6.*ELENG(J)*D5P-4.*ELENG(J)**2*D6P
      FF(5,J)=FF(5,J)+TEMF5
```

```
      FF(6,J)=FF(6,J)+TEMF6
      FF(2,J)=FF(2,J)-TEMF5-P1*ELENG(J)*(DIV2-DIV1)
      FF(3,J)=FF(3,J)-ELENG(J)*TEMF5-TEMF6-P1*ELENG(J)**2*(DIV2*
     1*2-DIV1**2)/2.
C
C    CASE 3    INCREASING TRIANGULAR LOAD
C
      CASE
      D5P=P2*ELENG(J)*(DIV2-DIV1)*(DIV2**2*(15.-4.*DIV2)+DIV2*DIV1*(10.
     1-3.*DIV2)+DIV1**2*(5.-2.*DIV2)-DIV1**3)/120.
      D6P=P2*(DIV2-DIV1)*(3.*DIV2**2+2.*DIV2*DIV1+DIV1**2)/24.
      TEMF5=6.*ELENG(J)*D6P-12.*D5P
      TEMF6=6.*ELENG(J)*D5P-4.*ELENG(J)**2*D6P
      FF(5,J)=FF(5,J)+TEMF5
      FF(6,J)=FF(6,J)+TEMF6
      FF(2,J)=FF(2,J)-TEMF5-P2*(D2-D1)/2.
      FF(3,J)=FF(3,J)-TEMF5*ELENG(J)-TEMF6-P2*(D2-D1)*(2*D2+D1)/6
     1.0
C
C    CASE 4    DECREASING TRIANGULAR LOAD
C
      CASE
      X31=ELENG(J)-D2
      X32=ELENG(J)-D1
      P31=P2
      P32=P1
      DIV31=X31/ELENG(J)
      DIV32=X32/ELENG(J)
      D5P=P32*ELENG(J)*(DIV32-DIV31)*(DIV32**2*(15.-4.*DIV32)+DIV32*DIV
     131*(10.-3.*DIV32)+DIV31**2*(5.-2.*DIV32)-DIV31**3)/120.
      D6P=P32*(DIV32-DIV31)*(3.*DIV32**2+2.*DIV32*DIV31+DIV31**2)/24.
      TEMF2=6.*ELENG(J)*D6P-12.*D5P
      TEMF3=4.*ELENG(J)**2*D6P-6.*ELENG(J)*D5P
      FF(2,J)=FF(2,J)+TEMF2
      FF(3,J)=FF(3,J)+TEMF3
      FF(5,J)=FF(5,J)-TEMF2-P32*ELENG(J)*(DIV32-DIV31)/2.
      FF(6,J)=FF(6,J)+TEMF2*ELENG(J)-TEMF3+P32*(X32-X31)*(2*X32+X
     131)/6.
C
C    CASE 5    CONCENTRATED AXIAL LOAD
C
      CASE
      FF(1,J)=FF(1,J)+P1*D1/ELENG(J)-P1
      FF(4,J)=FF(4,J)-P1*D1/ELENG(J)
C
C    CASE 6    UNIFORM AXIAL LOAD
C
      CASE
      FF(1,J)=FF(1,J)-P1*ELENG(J)/2
      FF(4,J)=FF(4,J)-P1*ELENG(J)/2
      END CASE
   30 CONTINUE
      END IF
C
      CALL ASSEMF(C1,C2,FF,MCODE,Q,F,NE,NEQ,QI,QJ,NPRINT)
```

```
C
      RETURN
      END
C
C     ******************************************************
C     *                                                    *
C     *                       ASSEMF                       *
C     *                                                    *
C     ******************************************************
C
C     ASSEMF ASSEMBLES THE FIXED END FORCES AND APPLIED JOINT LOAD
C     VECTOR INTO THE TOTAL APPLIED LOAD DISTRIBUTION VECTOR, Q.
C
      SUBROUTINE ASSEMF(C1,C2,FF,MCODE,Q,F,NE,NEQ,QI,QJ,NPRINT)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION C1(1),C2(1),FF(6,1),MCODE(6,1),Q(1),F(1),QJ(1)
C
C     STATEMENT FUNCTION TO TRANSFORM FIXED END FORCES TO GLOBAL
C     REFERENCE FRAME.
C
      FGF(C1I,C2I,FLX,FLY)=C1I*FLX+C2I*FLY
      DO 5 I=1,NEQ
        Q(I)=QJ(I)
    5 CONTINUE
      DO 20 I=1,NE
        DO 10 L=1,6
          K=MCODE(L,I)
          IF(K.EQ.0) GO TO 20
          DO CASE L
          CASE
            Q(K)=Q(K)-FGF(C1(I),-C2(I),FF(1,I),FF(2,I))
          CASE
            Q(K)=Q(K)-FGF(C2(I),C1(I),FF(1,I),FF(2,I))
          CASE
            Q(K)=Q(K)-FF(3,I)
          CASE
            Q(K)=Q(K)-FGF(C1(I),-C2(I),FF(4,I),FF(5,I))
          CASE
            Q(K)=Q(K)-FGF(C2(I),C1(I),FF(4,I),FF(5,I))
          CASE
            Q(K)=Q(K)-FF(6,I)
          END CASE
   10   CONTINUE
   20 CONTINUE
C
      IF(NPRINT.EQ.3) THEN DO
        PRINT 100
  100   FORMAT('    LOCAL FIXED END FORCES FOR EACH ELEMENT')
        DO 30 I=1,6
          PRINT 200,(FF(I,J),J=1,NE)
  200     FORMAT(' ',6(5X,D15.7))
   30   CONTINUE
      END IF
      RETURN
      END
```

```
C
C     ******************************************************
C     *                                                    *
C     *                       SOLVE                        *
C     *                                                    *
C     ******************************************************
C
C     SOLVE CALLS FACTOR IF THE STIFFNESS MATRIX HAS BEEN UPDATED
C     AND CALLS REDUCE AND BACSUB TO SOLVE A SYSTEM OF SIMULTANEOUS
C     LINEAR EQUATIONS BY GAUSS ELIMINATION.
C
      SUBROUTINE SOLVE(DD,F,MAXA,QT,SKT,NC,NEQ,NEGPIV,NKT,NPRINT)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION DD(1),F(1),MAXA(1),QT(1),SKT(1)
C
      IF(NPRINT.EQ.3) THEN DO
        PRINT 100
  100   FORMAT(' SOLVE CALLED')
      END IF
C
      IF(NC.EQ.0) THEN DO
        CALL FACTOR(MAXA,SKT,NEQ,NEGPIV,NPRINT)
        IF(NPRINT.EQ.3) THEN DO
          PRINT 200
  200     FORMAT(' FACTORED STIFFNESS MATRIX')
          PRINT 300,(SKT(I),I=1,NKT)
  300     FORMAT(' ',6(5X,D15.7))
        END IF
      END IF
      DO 10 I=1,NEQ
        DD(I)=QT(I)-F(I)
   10 CONTINUE
C
      CALL REDUCE(DD,MAXA,SKT,NEQ)
      CALL BACSUB(DD,MAXA,SKT,NEQ)
C
      RETURN
      END
C
C     ******************************************************
C     *                                                    *
C     *                       FACTOR                       *
C     *                                                    *
C     ******************************************************
C
C     FACTOR PERFORMS THE LDU FACTORIZATION OF THE STIFFNESS MATRIX.
C
      SUBROUTINE FACTOR(MAXA,SKT,NEQ,NEGPIV,NPRINT)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION MAXA(1),SKT(1)
C
      NEGPIV=0
      IF(NPRINT.EQ.3) THEN DO
        PRINT 100
  100   FORMAT(' FACTOR CALLED   DIAGONALS OF FACTORIZATION:')
```

```
      END IF
      DO 80 N=1,NEQ
        KN=MAXA(N)
        KL=KN+1
        KU=MAXA(N+1)-1
        KH=KU-KL
        IF(KH) 70,50,10
   10   K=N-KH
        IC=0
        KLT=KU
        DO 40 J=1,KH
          IC=IC+1
          KLT=KLT-1
          KI=MAXA(K)
          ND=MAXA(K+1)-KI-1
          IF(ND) 40,40,20
   20     KK=MINO(IC,ND)
          C=0.D00
          DO 30 L=1,KK
   30     C=C+SKT(KI+L)*SKT(KLT+L)
          SKT(KLT)=SKT(KLT)-C
   40   K=K+1
   50   K=N
        B=0.D00
        DO 60 KK=KL,KU
          K=K-1
          KI=MAXA(K)
          C=SKT(KK)/SKT(KI)
          B=B+C*SKT(KK)
   60   SKT(KK)=C
        SKT(KN)=SKT(KN)-B
C
C     STOP EXECUTION IF A ZERO PIVOT IS DETECTED
C
   70   IF(SKT(KN).EQ.0.D00) THEN DO
          PRINT 200,N,SKT(KN)
  200     FORMAT('-STIFFNESS MATRIX IS NOT POSITIVE DEFINITE'/'0PIVOT IS
     1 ZERO FOR D.O.F. ',I4/'0PIVOT = ',D15.8)
          STOP
        END IF
C
C     COUNT THE NUMBER OF NEGATIVE PIVOTS ENCOUNTERED.
C
        IF(SKT(KN).LT.0.D00) THEN DO
          NEGPIV=NEGPIV+1
          IF(NPRINT.EQ.3) THEN DO
            PRINT 300,NEGPIV
  300       FORMAT(' NEGATIVE PIVOT ENCOUNTERED IN FACTOR',I5)
          END IF
        END IF
C
        IF(NPRINT.EQ.3) THEN DO
          PRINT 400,SKT(KN)
  400     FORMAT(' ',D15.7)
        END IF
```

```
   80 CONTINUE
      RETURN
      END
C
C     ***********************************************************
C     *                                                         *
C     *                        REDUCE                           *
C     *                                                         *
C     ***********************************************************
C
C     REDUCE REDUCES THE RIGHT-SIDE LOAD VECTOR.
C
      SUBROUTINE REDUCE(DD,MAXA,SKT,NEQ)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION DD(1),MAXA(1),SKT(1)
C
      DO 20 N=1,NEQ
        KL=MAXA(N)+1
        KU=MAXA(N+1)-1
        KH=KU-KL
        IF(KH.GE.0) THEN DO
          K=N
          C=0.D00
          DO 10 KK=KL,KU
            K=K-1
   10     C=C+SKT(KK)*DD(K)
        DD(N)=DD(N)-C
        END IF
   20 CONTINUE
      RETURN
      END
C
C     ***********************************************************
C     *                                                         *
C     *                        BACSUB                           *
C     *                                                         *
C     ***********************************************************
C
C     BACSUB PERFORMS BACK-SUBSTITUTION TO OBTAIN THE SOLUTION.
C
      SUBROUTINE BACSUB(DD,MAXA,SKT,NEQ)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION DD(1),MAXA(1),SKT(1)
C
      DO 10 N=1,NEQ
        K=MAXA(N)
        DD(N)=DD(N)/SKT(K)
   10 CONTINUE
      IF(NEQ.EQ.1) RETURN
      N=NEQ
      DO 30 L=2,NEQ
        KL=MAXA(N)+1
        KU=MAXA(N+1)-1
        KH=KU-KL
        IF(KH.GE.0)THEN DO
```

```
            K=N
            DO 20 KK=KL,KU
              K=K-1
              DD(K)=DD(K)-SKT(KK)*DD(N)
   20       CONTINUE
          END IF
          N=N-1
   30   CONTINUE
        RETURN
        END
C
C       ********************************************************
C       *                                                      *
C       *                        TEST                          *
C       *                                                      *
C       ********************************************************
C
C       TEST CALLS ENERGY, DISPLC, DISPLB, AND UNBALF AS INDICATED TO
C       TEST FOR CONVERGENCE TO AN EQUILIBRIUM POINT.
C
        SUBROUTINE TEST(AREA,BP1,BP2,C1,C2,D,DD,DD1,DE,ELENG,EMOD,F,FG,FL,
       1FP,FPI,JCODE,MCODE,QT,R1,R2,SP1,SP2,ST1,ST2,ZI,CPDB,CPDC,CPE,CPF,D
       2P,DQQR,ICI,IMAX,NE,NEQ,NJ,A1,A2,DIST,FF,MAT,MN,Q,NAT,QI,CI1,CI2,FQ
       3R,IN,ITIND,NPRINT)
        IMPLICIT REAL*8(A-H,O-Z)
        DIMENSION AREA(1),BP1(1),BP2(1),C1(1),C2(1),D(1),DD(1),DD1(1),DE(1
       1),ELENG(1),EMOD(1),F(1),FG(6,1),FL(6,1),FP(1),FPI(1),JCODE(3,1),MC
       2ODE(6,1),QT(1),R1(1),R2(1),SP1(1),SP2(1),ST1(1),ST2(1),ZI(1),A1(1)
       3,A2(1),DIST(1),FF(6,1),MAT(1),MN(1),Q(1),CI1(1),CI2(1)
C
        ICI=0
        IF(ITIND.EQ.0) THEN DO
        CALL ENERGY(DD,DD1,FP,FPI,QT,CPE,DP,ICI,NEQ,ITIND)
        END IF
C
        IF(CPDC.LT.1.D00) THEN DO
          CALL DISPLC(D,DD,JCODE,CPDC,ICI,NJ)
        END IF
C
        IF(CPDB.LT.1.D00) THEN DO
          CALL DISPLB(D,DD,CPDB,ICI,NEQ)
        END IF
C
        IF(CPF.LT.1.D00) THEN DO
          CALL UNBALF(F,FP,QT,CPF,ICI,NEQ)
        END IF
C
        IF(NPRINT.EQ.3) THEN DO
          PRINT 100,ICI
  100     FORMAT('    ICI= ',I5)
        END IF
        RETURN
        END
C
C       ********************************************************
```

```
C     *                                                     *
C     *                       ENERGY                        *
C     *                                                     *
C     *******************************************************
C
C     ENERGY TESTS FOR CONVERGENCE TO AN EQUILIBRIUM CONFIGURATION
C     USING INTERNAL ENERGY CRITERIA.
C
      SUBROUTINE ENERGY(DD,DD1,FP,FPI,QT,CPE,DP,ICI,NEQ,ITIND)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION DD(1),DD1(1),FP(1),FPI(1),QT(1)
C
      CD=0.D0
      CN=0.D0
      DO 10 I=1,NEQ
        CN=CN+DD(I)*(QT(I)-FPI(I))
        CD=CD+DD1(I)*(QT(I)-FP(I))
   10 CONTINUE
        C=DABS(CN/CD)
C
      IF(ITIND.EQ.0.AND.C.GT.DP) THEN DO
        PRINT 100,C
  100   FORMAT('-DIVERGENCE DETECTED IN ENERGY **   C= ',F13.2//'PRESENT
     1 LOAD VECTOR ,DD,FPI,DD1,FP')
        DO 20 I=1,NEQ
          PRINT 200,QT(I),DD(I),FPI(I),DD1(I),FP(I)
  200     FORMAT(5(5X,D15.7))
   20   CONTINUE
        STOP
      END IF
C
      IF(CPE.LT.1.D00)THEN DO
        IF(C.GT.CPE) THEN DO
          ICI=ICI+1000
        END IF
      END IF
C
      RETURN
      END
C
C     *******************************************************
C     *                                                     *
C     *                      DISPLC                         *
C     *                                                     *
C     *******************************************************
C
C     DISPLC TESTS FOR CONVERGENCE TO AN EQUILIBRIUM CONFIGURATION
C     USING THE INFINITY VECTOR NORM OF DISPLACEMENTS.
C
      SUBROUTINE DISPLC(D,DD,JCODE,CPDC,ICI,NJ)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION D(1),DD(1),JCODE(3,1)
C
      RMT=0.D00
      RMR=0.D00
```

```
      CR=0.D00
      CT=0.D00
      I=1
      WHILE(I.LE.3) DO
        J=1
        WHILE(J.LE.NJ) DO
          K=JCODE(I,J)
          IF(K.NE.0) THEN DO
            IF(I.EQ.3) THEN DO
              RMR=DMAX1(DABS(D(K)),RMR)
              CR=DMAX1(DABS(DD(K)),CR)
            ELSE DO
              RMT=DMAX1(DABS(D(K)),RMT)
              CT=DMAX1(DABS(DD(K)),CT)
            END IF
          END IF
          J=J+1
        END WHILE
        I=I+1
      END WHILE
      IF(RMR.NE.0.D00) THEN DO
        CPR=CR/RMR
C
        IF(CPR.GT.CPDC) THEN DO
          ICI=ICI+1
        END IF
C
      END IF
      IF(RMT.NE.0.D00)THEN DO
        CPT=CT/RMT
C
        IF(CPT.GT.CPDC) THEN DO
          ICI=ICI+1
        END IF
C
      END IF
      IF(RMR.EQ.0.D00.AND.RMT.EQ.0.D00) THEN DO
        ICI=ICI+1
        PRINT 100,CR,RMR,CT,CMT
  100   FORMAT(' MAX D IN DISPLC = 0  CR=',D15.7,'RMR=',D15.7,' CT=',D15
     1.7,' CMT=',D15.7)
        STOP
      END IF
      RETURN
      END
C
C     ************************************************************
C     *                                                          *
C     *                       DISPLB                             *
C     *                                                          *
C     ************************************************************
C
C     DISPLB TESTS FOR CONVERGENCE TO AN EQUILIBRIUM CONFIGURATION
C     USING THE EUCLIDEAN VECTOR NORM OF DISPLACEMENTS.
C
```

```
      SUBROUTINE DISPLB(D,DD,CPDB,ICI,NEQ)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION D(1),DD(1)
C
      CN=0.D00
      CD=0.D00
      DO 10 I=1,NEQ
        CN=CN+(DD(I))**2
        CD=CD+(D(I))**2
   10 CONTINUE
C
      IF(CD.EQ.0.D00) THEN DO
        ICI=ICI+10
        RETURN
      END IF
C
      C=(DSQRT(CN))/(DSQRT(CD))
      IF(C.GT.CPDB) THEN DO
        ICI=ICI+10
      END IF
C
      RETURN
      END
C
C     ***********************************************************
C     *                                                         *
C     *                       UNBALF                            *
C     *                                                         *
C     ***********************************************************
C
C     UNBALF TESTS FOR CONVERGENCE TO AN EQUILIBRIUM CONFIGURATION
C     USING THE UNBALANCED FORCE CRITERIA.
C
      SUBROUTINE UNBALF(F,FP,QT,CPF,ICI,NEQ)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION F(1),FP(1),QT(1)
C
      CN=0.D00
      CD=0.D00
      DO 10 I=1,NEQ
        CN=CN+(QT(I)-F(I))**2
        CD=CD+(QT(I)-FP(I))**2
   10 CONTINUE
      IF(CD.EQ.0.D00) THEN DO
        ICI=ICI+100
        RETURN
      END IF
      C=(DSQRT(CN))/(DSQRT(CD))
C
      IF(C.GT.CPF) THEN DO
        ICI=ICI+100
      END IF
C
      RETURN
      END
```

```
C
C     ******************************************************
C     *                                                    *
C     *                      RESULT                        *
C     *                                                    *
C     ******************************************************
C
C     RESULT ZEROS THE JOINT FORCE MATRIX AND CALLS JOINTF
C     AND OUTPUT.
C
      SUBROUTINE RESULT(A1,A2,C1,C2,D,DIST,DJ,ELENG,EMOD,FF,FG,FL,JCODE,
     1MAT,MCODE,MINC,MN,P,Q,QT,ZI,NAT,NE,NEQ,NJ,QI,IMP,NPRINT)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A1(1),A2(1),C1(1),C2(1),D(1),DIST(1),DJ(3,1),ELENG(1),EM
     1OD(1),FF(6,1),FG(6,1),FL(6,1),JCODE(3,1),MAT(1),MCODE(6,1),MINC(2,
     21),MN(1),P(3,1),Q(1),QT(1),ZI(1)
C
      DO 20 J=1,NJ
        DO 10 I=1,3
          P(I,J)=0.D00
   10   CONTINUE
   20 CONTINUE
C
      CALL JOINTF(FG,MINC,P,NE)
      CALL OUTPUT(D,DJ,FL,JCODE,P,QT,NE,NEQ,NJ,QI,IMP,NPRINT)
C
      RETURN
      END
C
C     ******************************************************
C     *                                                    *
C     *                      JOINTF                        *
C     *                                                    *
C     ******************************************************
C
C     JOINTF GENERATES THE JOINT FORCE MATRIX.
C
      SUBROUTINE JOINTF(FG,MINC,P,NE)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION FG(6,1),MINC(2,1),P(3,1)
C
      DO 20 I=1,NE
        J=MINC(1,I)
        K=MINC(2,I)
        DO 10 L=1,3
          P(L,J)=P(L,J)+FG(L,I)
          P(L,K)=P(L,K)+FG(L+3,I)
   10   CONTINUE
   20 CONTINUE
      RETURN
      END
C
C     ******************************************************
C     *                                                    *
C     *                      OUTPUT                        *
```

```
C     *                                                              *
C     ****************************************************************
C
C     OUTPUT PRINTS THE RESULTS FOR EACH EQUILIBRIUM POINT.
C
      SUBROUTINE OUTPUT(D,DJ,FL,JCODE,P,QT,NE,NEQ,NJ,QI,IMP,NPRINT)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION D(1),DJ(3,1),FL(6,1),JCODE(3,1),P(3,1),QT(1)
      DO 20 J=1,NJ
        DO 10 I=1,3
          DJ(I,J)=0.D00
          K=JCODE(I,J)
          IF(K.NE.0) THEN DO
            DJ(I,J)=D(K)
          END IF
   10   CONTINUE
   20 CONTINUE
C
      IF(NPRINT.LE.1) THEN DO
        PRINT 100,QI,D(IMP)
  100   FORMAT(' ',F17.9/' ',F17.9)
      END IF
C
      IF(NPRINT.EQ.2.OR.NPRINT.EQ.3) THEN DO
        PRINT 200,QI,D(IMP)
  200   FORMAT(' QI EQUILIB =',F17.9,'  D EQUILIB =',F17.9)
      END IF
C
      IF(NPRINT.EQ.4) THEN DO
      PRINT 300,QI
  300 FORMAT('-',80('_')//31X,'QI=',F15.7//34X,'LOAD VECTOR'/34X,11('*')
     1)
      DO 30 I=1,NEQ
        PRINT 400,QT(I)
  400   FORMAT('0',31X,F16.5)
   30 CONTINUE
      PRINT 500
  500 FORMAT('-',26X,'GLOBAL JOINT DISPLACEMENTS'/27X,26('*')//10X,'JOIN
     1T',7X,'1-DIRECTION',8X,'2-DIRECTION',8X,'3-DIRECTION')
      DO 40 J=1,NJ
        PRINT 600,J,(DJ(I,J),I=1,3)
  600   FORMAT(' ',8X,I4,3(2X,F17.9))
   40 CONTINUE
      PRINT 700
  700 FORMAT('-',29X,'LOCAL ELEMENT FORCES'/30X,20('*')/' ELEMENT',14X,'
     1A-END',31X,'B-END'/' NUMBER',6X,'1',11X,'2',11X,'3',11X,'4',11X,'5
     2',11X,'6')
      DO 50 J=1,NE
        PRINT 800,J,(FL(I,J),I=1,6)
  800 FORMAT('0',I4,6(1X,F11.4))
   50 CONTINUE
      PRINT 900
  900 FORMAT('-',33X,'JOINT FORCES'/34X,12('*')/10X,'JOINT',7X,'1-DIRECT
     1ION',8X,'2-DIRECTION',8X,'3-DIRECTION')
      DO 60 J=1,NJ
```

```
        PRINT 600,J,(P(I,J),I=1,3)
   60 CONTINUE
      PRINT 1000
 1000 FORMAT('0',80('_'))
      END IF
      RETURN
      END
C
C     ********************************************************
C     *                                                      *
C     *                      DOTPRD                          *
C     *                                                      *
C     ********************************************************
C
C     DOTPRD COMPUTES THE DOT PRODUCT OF DOT1 AND DOT2.
C
      FUNCTION DOTPRD(DOT1,DOT2,N)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION DOT1(1),DOT2(1)
C
      DOTPRD=0.D00
      DO 10 I=1,N
        DOTPRD=DOTPRD+DOT1(I)*DOT2(I)
   10 CONTINUE
      RETURN
      END
C
C     ********************************************************
C     *                                                      *
C     *                      UPDATE                          *
C     *                                                      *
C     ********************************************************
C
C     UPDATE COMPUTES THE APPLIED LOAD VECTOR FROM THE LOAD
C     DISTRIBUTION VECTOR AND THE LOAD INCREMENT.
C
      SUBROUTINE UPDATE(QT,Q,NEQ,QI)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION QT(1),Q(1)
C
      DO 10 I=1,NEQ
        QT(I)=Q(I)*QI
   10 CONTINUE
      RETURN
      END
C
C
//DATA
```

The vita has been removed from the scanned document