Comparison of Nonlinear Finite Element Formulations: Application to Trusses

by

Christopher J. Earls

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute & State University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE
in
Civil Engineering

Approved:

S. M. Holzer, Chairman

R. H. Plaut

W. S. Easterling

July, 1992

Blacksburg, Virginia

Comparison of Nonlinear Finite Element Formulations: Applications to Trusses

by

Christopher J. Earls

Committee Chairman: Siegfried M. Holzer

Civil Engineering

(ABSTRACT)

Two prominent continuum mechanics based incremental nonlinear finite element formulations are reviewed. An introduction to different material response measures suitable for nonlinear analysis, in addition to an overview of the Total and Updated Lagrangian reference frames, serve as the starting point for this review.

The two nonlinear formulations are specialized for use with a geometrically nonlinear plane truss finite element. The truss formulations are then implemented into separate geometrically nonlinear finite element codes.

Numerical comparisons of five test structures are carried out using ABAQUS and both programs. ABAQUS serves as the bench-mark by which the solution accuracy of the two programs is judged.

Acknowledgements

The Author expresses his sincerest thanks to his advisor, Dr. Siegfried M. Holzer. His patience, encouragement, dedication, and insights into the ways of the world will not be forgotten. To Dr. Samuel Easterling the utmost gratitude is extended for his advice, valuable assistance, and the remarkable enthusiasm which served to inspire the author. Special thanks go to Dr. Raymond Plaut for his generosity in reviewing this thesis and serving on the committee. The author would also like to thank Dr. Ayodele O. Abatan for his helpful discussions, insights, and the generous use of his computer. To a debt of gratitude is owed for his countless hours of assistance with programming and MACSYMA.

To my fellow graduate students I extend thanks, especially to , and for their discussions and opinions. A hearty thanks is also due for his thoughts, Philosophy and for smiling the smile of reason.

Finally, the author extends a very special thanks to for her kindness, understanding, advice, faith, and love. It is to that this thesis is dedicated with fond appreciation.

Table of Contents

List of Figures

Chapter 1
Introduction

1.1 Purpose and Scope

The primary purpose of this thesis is to give a presentation and comparison of the two most prominent continuum mechanics based incremental nonlinear finite element formulations. The first formulation, known as the B-notation, is attributed to Zienkiewicz (1973) and is by far the most commonly used. This formulation is compared to the formulation presented by Bathe et al. (1975). The motivation for this study comes from a lack of discussion in the literature concerning the formulation presented in the landmark paper by Bathe et al. (1975). An investigation into the literature reveals that the paper has been cited in 87 separate journal publications. Upon examining a large number of these publications,listed in the bibliography, it became apparent that the application of the formulation, and not its verification, was the thrust of most of the work. Based on this fact it was viewed that work done in comparing the formulations of Zienkiewicz and Bathe may be of some merit.

Both formulations are specialized for use with a geometrically nonlinear plane truss finite element. These truss formulations are implemented into two separate nonlinear finite element Fortran codes. A comparison of the two formulations is carried out in two parts:

(1) Formulations of incremental iterative equations

(2) Numerical solutions of truss problems

1.2 Overview

An introduction to nonlinear analysis in addition to discussions on common material response measures and the Total and Updated Lagrangian are presented in Chapter 2. The formulation of the nonlinear plane truss in the B-notation and the notation of Bathe is given in chapter 3; in addition the formulation of the 2-D nonlinear frame element is presented in Chapter 4. The programs used in this study are discussed in Chapter 5 and the program listings themselves appear in Appendix B. Results from the test problems are given in Chapter 6 and conclusions and recommendations are contained in Chapter 7.

Chapter 2
Nonlinear Finite Element Analysis

2.1 Overview

This chapter serves as an introduction to nonlinear finite element analysis. A discussion of a simple one degree of freedom truss example will serve as a basis from which the Newton/Raphson algorithm and the Lagrangian reference frames may be introduced. Also included in this chapter is a brief discussion and comparison of different stress and strain measures in addition to an introduction to some common finite element notations.

2.2 Introduction to nonlinear finite element analysis

According to Bathe (1982, p.302) there are three types or sources of nonlinearity: geometric, material and contact problems. Geometric nonlinearity is caused by nonlinear strain-displacement relations and equilibrium being expressed for the deformed state. It is this type of nonlinearity which is the focus of the work presented in this thesis.

In nonlinear analysis the primary objective is to find the state of equilibrium of a structure corresponding to a set of applied loads. This concept may be best illustrated when applied to a simple problem. For this discussion, closely following that of Crisfield (1991), the structure in Fig. 2-1 will be used. The load P is applied to the structure which in turn causes the deformation W. From Crisfield (1991, p.2), vertical equilibrium may be satisfied by

$$P = N \sin\theta = \frac{N(Z+W)}{L_1} \simeq \frac{N(Z+W)}{L_0} \tag{2-1}$$

where N is the axial force in the bar and θ is assumed small. From Pythagorean's theorem

$$\epsilon = \frac{\left((Z+W)^2 + L^2\right)^{\frac{1}{2}} - \left(Z^2 + L^2\right)^{\frac{1}{2}}}{\left(Z^2 + L^2\right)^{\frac{1}{2}}} \tag{2-2}$$

$$= \left(1 + \left(\frac{Z+W}{L}\right)^2\right)^{\frac{1}{2}} \left(1 + \left(\frac{Z}{L}\right)^2\right)^{-\frac{1}{2}} - 1 \tag{2-3}$$

$$\simeq \left(1 + \frac{1}{2}\left(\frac{Z+W}{L}\right)^2\right) \left(1 - \frac{1}{2}\left(\frac{Z}{L}\right)^2\right) - 1 \tag{2-4}$$

$$= \left(\frac{Z}{L}\right)\left(\frac{W}{L}\right) + \frac{1}{2}\left(\frac{W}{L}\right)^2 \simeq \left(\frac{Z}{L_0}\right)\left(\frac{W}{L_0}\right) + \frac{1}{2}\left(\frac{W}{L_0}\right)^2 \tag{2-5}$$

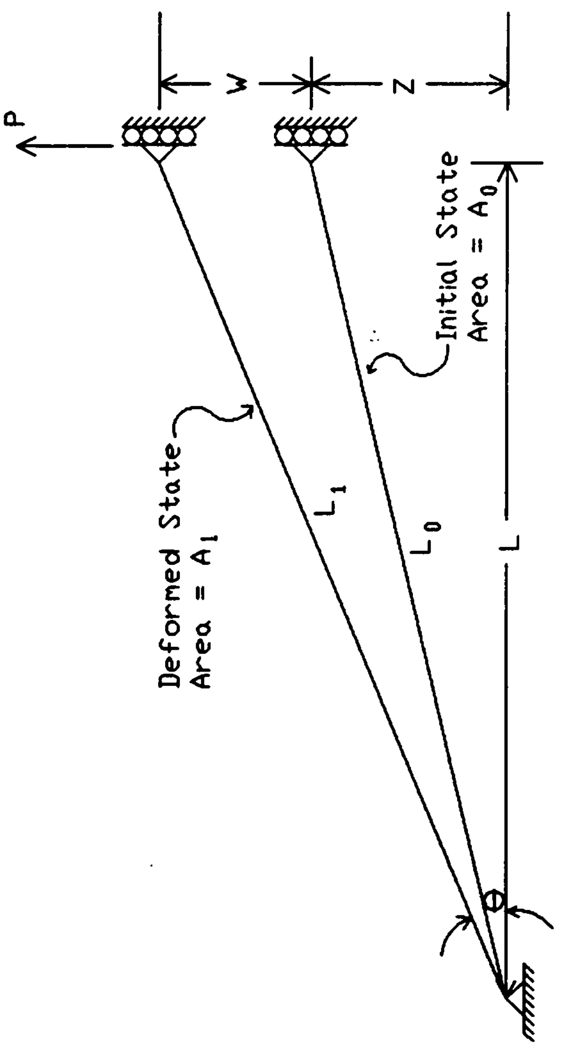


Figure (2-1) One Degree of Freedom Example

From equation 2-5, the internal force in the truss is

$$N = E A \epsilon = E A \left(\left(\frac{Z}{L_0}\right)\left(\frac{W}{L_0}\right) + \frac{1}{2}\left(\frac{W}{L_0}\right)^2 \right) \tag{2-6}$$

where E is the modulus of elasticity of the material that the truss is made out of, and A is the cross-sectional area of the truss. Substituting equation 2-6 into 2-1 yields

$$P = \frac{EA}{(L_0)^3} \left(Z^2W + \frac{3}{2}ZW^2 + \frac{1}{2}W^3 \right) \tag{2-7}$$

Using equation 2-7, the load versus deflection (or equilibrium) path of the structure can be plotted as is done in Fig. 2-2.

In nonlinear analysis the tangent stiffness matrix is used as a means for relating changes in load with changes in displacement. For the example being discussed, Crisfield (1991, p.4) demonstrates how this matrix becomes a scalar

$$K_t = \frac{dP}{dW} = \frac{(Z+W)}{L_0}\frac{dN}{dW} + \frac{N}{L_0} \tag{2-8}$$

$$= \frac{EA}{L_0}\left(\frac{Z+W}{L_0}\right)^2 + \frac{N}{L_0} \tag{2-9}$$

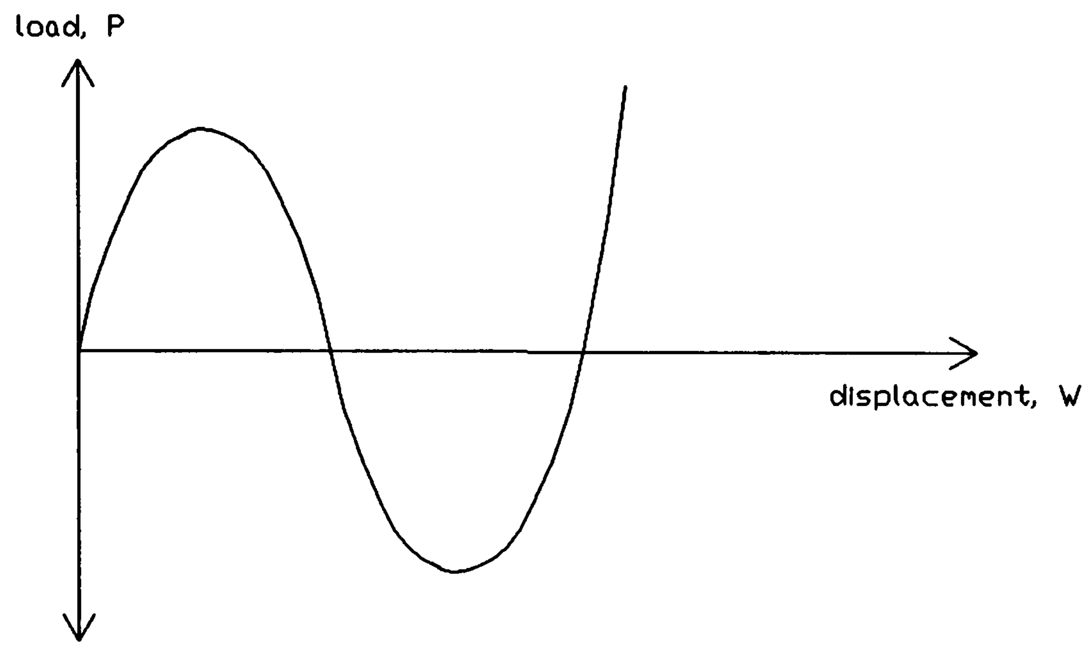


Figure (2-2) Equilibrium Path of Example Structure

$$= \frac{EA}{L_0}\left(\frac{Z}{L_0}\right)^2 + \frac{EA}{L_0}\left(\frac{2ZW + W^2}{(L_0)^2}\right) + \frac{N}{L_0} \tag{2-10}$$

The expression of the internal force in conjunction with the K_t may be used to plot the nonlinear equilibrium path of the structure.

2.3 The Newton-Raphson Method

There are several algorithms currently used in the incremental solution of nonlinear problems. The method used throughout this thesis is the Newton-Raphson method. This algorithm traces the response of the nonlinear structural system by the use of piecewise linear stiffness increments. These linear stiffness increments are the tangent stiffness matrices as given in the above discussion. It is felt that the concept being presented can be explained through a simple example. When considering the same simple one degree of freedom example as discussed above, a very clear depiction of the Newton-Raphson algorithm is possible. In figure 2-3, the equilibrium path for this one degree of freedom system is given. It should be noted that for the purposes of this discussion all superscripts refer the quantity in question to the configuration in which it occurs. In the case of figure 2-3, the initial configuration is given by the number 0. The configuration denoted by the letter N is some equilibrium state of the structure corresponding to the load Q^N. The loads Q^0 and Q^N are referred to as load steps.

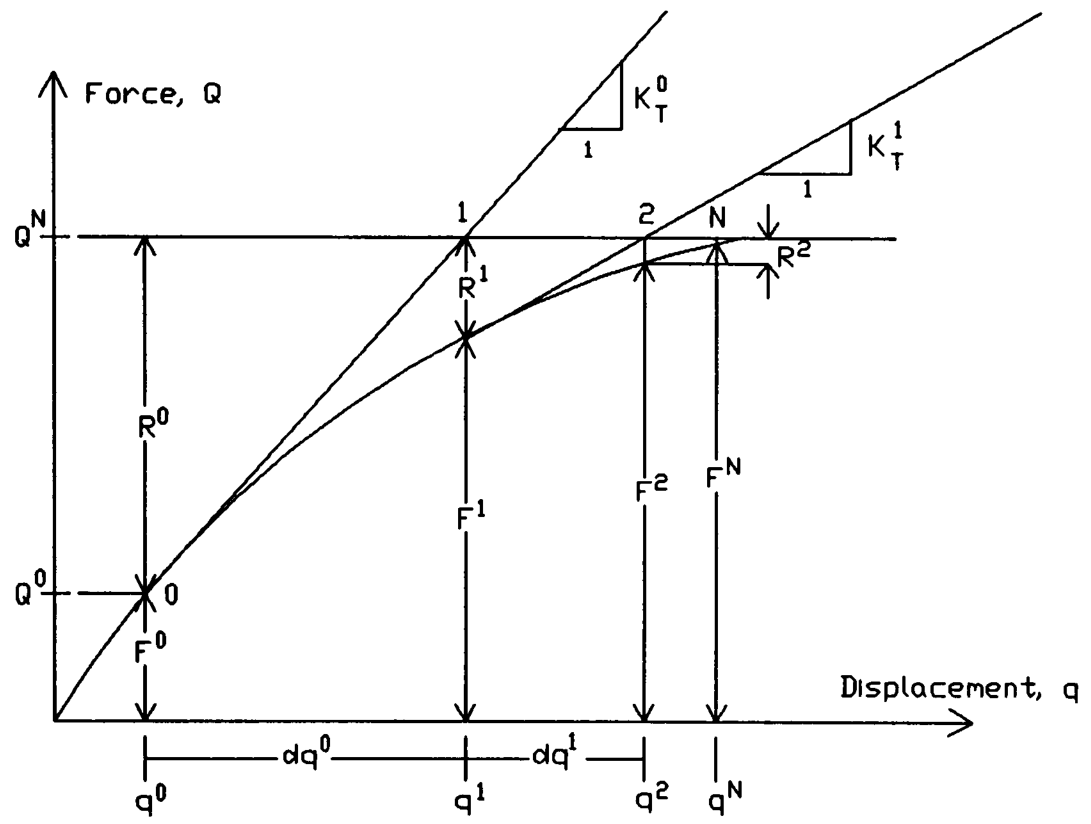


Figure (2-3) Physical Interpretation of the Newton-Raphson Algorithm for a One Degree of Freedom System

They represent specific loads on the structure for which an equilibrium configuration is desired. The numbered configurations occurring between the load steps Q^0 and Q^N are referred to as the iterations between the load steps. These iterations are where the actual mechanics of the nonlinear solution are occurring. For the one degree of freedom example in figure 2-3, it is seen that the configuration at time zero is known and thus is our first reference point. The configuration of the structure at time N is desired. As can be seen from the figure, the internal force at time 0, F^0, falls considerably short of equilibrating an applied load of Q^N. This difference in internal force and applied external load is referred to as the residual. Thus the structure must deform in order to develop the internal resistance necessary to equilibrate the externally applied load of Q^N. The question then becomes how much deformation is required. This question can only be answered by successive approximations. The first step in this approximation procedure is the generation of the tangent stiffness matrix about the first equilibrium point. In the case of figure 2-3, the matrix K_T^0 is generated using equation 2-10. This matrix is a linearized approximation of the equilibrium path based on the internal forces occurring in the structure at the initial configuration. The tangent stiffness matrix represents a line extending from the initial equilibrium point both up and down the equilibrium path. Since the structure is being loaded, the upward segment is of interest, specifically the region that intersects the load level defined by the horizontal line corresponding to Q^N. This load level is of interest since it defines the level of internal force that the structure must achieve to be in equilibrium. If the line corresponding to the tangent stiffness is followed up to this load level, then a corresponding structural displacement can be measured on the abscissa. This procedure can be represented by the simple linear equation

$$R^k = K_T^k \ \Delta q^k \tag{2-11}$$

where R^k is the residual and is given by

$$R^k = Q^N - F^k \tag{2-12}$$

In the Newton-Raphson algorithm, the Δq^k term is the unknown being solved for. This term represents the increment in displacement that the algorithm prescribes for the structure at the next equilibrium point. The next structural configuration can be determined from the simple addition

$$q^{k+1} = q^k + \Delta q^k \tag{2-13}$$

This new configuration defined by the displacement q^{k+1} is used to compute the structural response at this configuration. The displacement q^{k+1} is substituted into the Green-Lagrange strain displacement relationship. In the case of linearly elastic material response, the Second Piola-Kirchhoff stress is obtained by multiplying the Green-Lagrange strain by the material's modulus of elasticity. Once the material response terms have been defined, the new internal force can be computed and the residual found. This whole process is repeated until the user prescribed accuracy is achieved. The measure of accuracy used is the magnitude of the residual. Iteration stops when the residual is sufficiently small, as defined by the analyst. To be able to implement the Newton-Raphson algorithm in the solution of nonlinear finite element analysis, a discussion of stress and strain measures currently used in nonlinear finite element analysis is required.

2.4 Common Strain Measures

A discussion of common strain measures is a good starting point in the consideration of the nonlinear finite element equations of motion. This discussion will focus on the four most common measures; they are the engineering strain, the rotated engineering strain, Green-Lagrange strain and the log strain. The engineering strain is the most familiar strain measure to the engineer since all elementary mechanics courses define strain in terms of this measure. The log strain is the most correct of the strain measures since, according to Crisfield (1991, p.59), it represents the true material response as referred to the current material configuration. The Green-Lagrange strain, as will be seen, is the strain measure of choice when operating within the framework of continuum mechanics based incremental nonlinear finite element analysis. All four of the above mentioned strain measures will be addressed, individually and in relation to each other, in the following discussion.

2.4.1 Engineering Strain

Engineering strain is the most common measure of material deformation. Since strain in general is a point-wise phenomenon, the meaning of engineering strain will be presented in the context of a material fiber of the truss in configuration 0 of Fig 2-1, whose length is reduced to zero through a limit. This presentation follows from what was given by Stippes et al. (1961). On the basis of Fig. 2-4, the normal strain at point P in the x-direction is

$$\epsilon_E = \lim_{\Delta x \to 0} \frac{\Delta x' - \Delta x}{\Delta x} \tag{2-14}$$

From Fig. 2-4 it can be seen that

$$\Delta x' = \Delta x + u(x+\Delta x) - u(x) \tag{2-15}$$

which leads to

$$\Delta x' - \Delta x = u(x+\Delta x) - u(x) \tag{2-16}$$

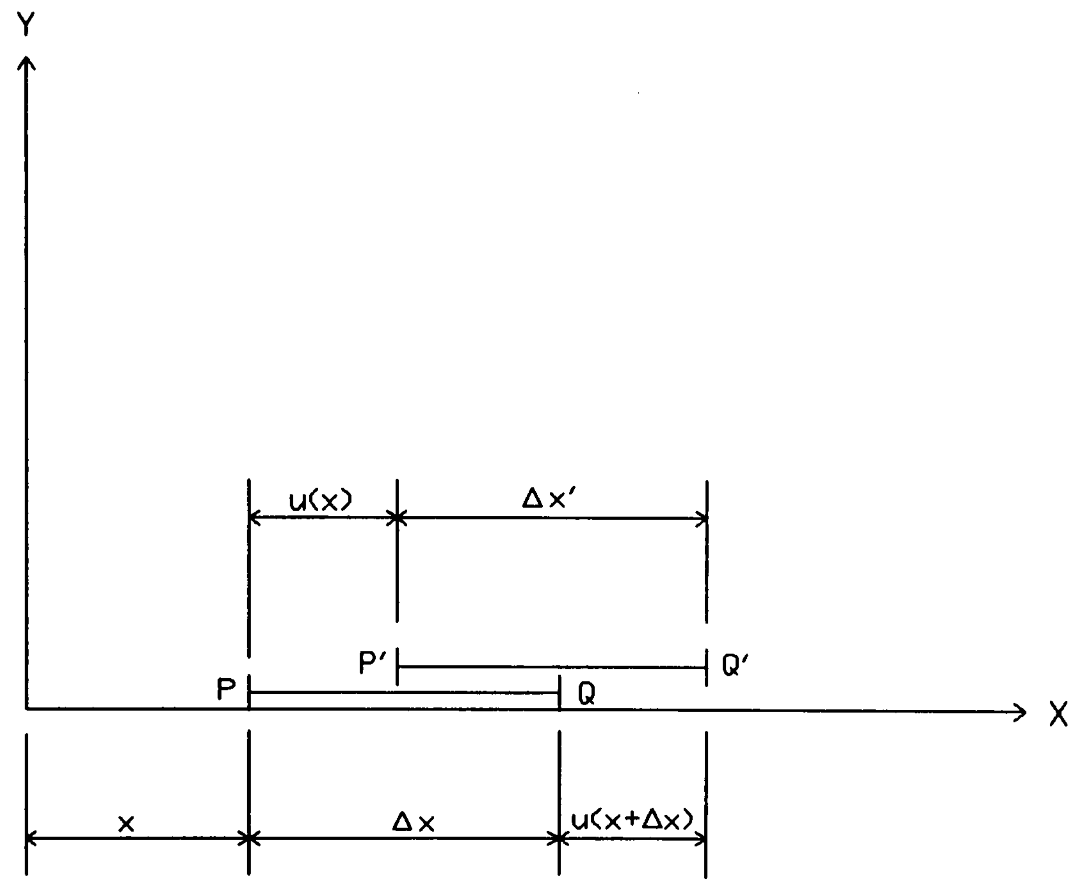


Figure (2-4) Description of Engineering Strain

So,

$$\epsilon_E = \lim_{\Delta x \to 0} \frac{u(x+\Delta x) - u(x)}{\Delta x} = \frac{du}{dx} \qquad (2\text{-}17)$$

This definition of engineering strain is purely extensional in nature and clearly shows its linear dependence on displacement.

2.4.2 Rotated Engineering Strain

The presentation of the rotated engineering strain given by Crisfield (1991, p.58) contains a figure similar to Fig 2-5. According to Crisfield, the rotated engineering strain, commonly used in the Updated Lagrangian formulation, is defined as

$$\epsilon = \frac{L_1 - L_0}{L_0} \qquad (2\text{-}18)$$

where L_1 is the length of the truss in the new configuration and L_0 is the length of the truss in the previous configuration. The rotated engineering strain can then be described as the strain given by equation 2-18 directed along the local x-axis of configuration 0 (see Fig 2-5). In general for rotated engineering strain, the strain

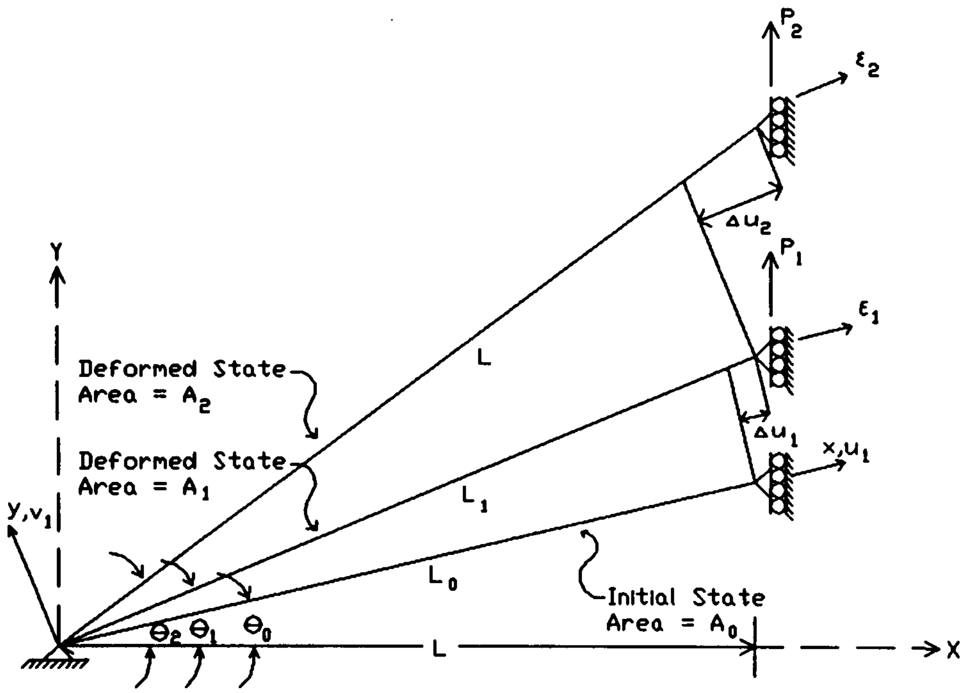


Figure (2-5) One Degree of Freedom Example of Rotated Engineering Strain for the new configuration is given in terms of the local axes of the previous configuration.

2.4.3 Green-Lagrange Strain

The Green-Lagrange strain measure, commonly used in the Total Lagrangian, can be related in terms of what has already been done with the engineering strain. Equation 2-13 can be modified as follows from Crisfield (1991, p.59):

$$\epsilon = \frac{L_1 - L_0}{L_0} = \frac{\left(L_1 - L_0\right)\left(L_1 + L_0\right)}{L_0\left(L_1 + L_0\right)} = \frac{L_1^2 - L_0^2}{L_0^2(2 + \epsilon)} \tag{2-19}$$

If we assume ϵ is small relative to 2, we obtain

$$\epsilon_G = \frac{L_1^2 - L_0^2}{2L_0^2} \tag{2-20}$$

where ϵ_G is the Green-Lagrange strain (Crisfield 1991, p.59). The Green-Lagrange strain in equation 2-20 can be related to the rotated engineering strain of equation 2-18 by

$$\epsilon_G = \epsilon\left(1+\frac{1}{2}\epsilon\right) \tag{2-21}$$

For small strains the rotated engineering strain of equation 2-18 is coincident with the Green-Lagrange strain of equation 2-20. This can be shown to be true since in the small strain case, $L_1 \simeq L_0$ and equation 2-20 can be reduced to

$$\epsilon_G = \frac{\left(L_1 + L_0\right)\left(L_1 - L_0\right)}{2L_0^2} \simeq \frac{2L_0\left(L_1 - L_0\right)}{2L_0^2} \tag{2-22}$$

Thus it can be seen that the Green-Lagrange strain coincides with the rotated engineering strain as shown below:

$$\epsilon_G = \frac{2L_0\left(L_1 - L_0\right)}{2L_0^2} = \frac{L_1 - L_0}{L_0} = \epsilon \tag{2-23}$$

It is only in a state of large strain that differences between the measures become apparent (Crisfield 1991, p.58).

2.4.4 Rotated Log-Strain

The final strain measure to be addressed is the log-strain. Log-strain is commonly referred to as the true strain. This means that it represents the state of strain in the deformed configuration as measured with respect to the local reference frame of the deformed configuration. This measure is often used in applications of large strains. The log-strain is defined in an incremental form as follows (Crisfield 1991, p.59):

$$d\epsilon = \frac{d\mathrm{L}_1}{\mathrm{L}_1} \tag{2-24}$$

where L_1 is the length of the truss in the deformed configuration. Equation 2-24 yields the log strain

$$\epsilon_L = \int_{L_0}^{L_1} d\epsilon = \ln\left(\frac{L_1}{L_0}\right) \tag{2-25}$$

which can be related to the rotated engineering and Green-Lagrange strains as follows (Crisfield 1991, p.60). For the case of rotated engineering strain

$$\epsilon_L = \ln(1+\epsilon) = \ln\left(1 + \frac{L_1 - L_0}{L_0}\right) \tag{2-26}$$

$$= \ln\left(\frac{L_1 - L_0}{L_0}\right) = \ln\frac{L_1}{L_0} \tag{2-27}$$

Considering the Green-Lagrange strain,

$$\epsilon_L = \frac{1}{2}\ln\left(1 + 2\epsilon_G\right) = \frac{1}{2}\ln\left[1 + 2\left(\frac{L_1^2 - L_0^2}{2L_0^2}\right)\right] \tag{2-28}$$

$$= \frac{1}{2}\ln 2\left[\frac{\left(L_1 + L_0\right)\left(L_1 - L_0\right)}{2L_0}\right] = \ln\frac{L_1}{L_0} \tag{2-29}$$

Now that some strain measures for nonlinear analysis have been introduced, the topic of stress measures will be the focus of the following discussion.

2.5 Common Stress Measures

The four most common stress measures are engineering stress, rotated engineering stress, Second Piola-Kirchhoff stress, and Cauchy stress. Each of these stress measures has a complementary strain measure which will be discussed. The issue of whether or not the two measures are energetically conjugate must be addressed when choosing a suitable pairing of stress and strain measures (Bathe 1982, p.327 ; Crisfield 1991, p.65). To be energetically conjugate the scalar product of the stress and strain must produce a valid work term. For the work term to be valid, both of the material response measures must be evaluated in the same equilibrium configuration and have their respective responses given in terms of the same reference frame (Bathe 1982, p.327). In the case of geometric nonlinear finite element analysis, obtaining the stress measure which is energetically conjugate to the strain measure used is straight-forward (Bathe 1982, p.336). This is the case since the material response is in the elastic range and thus Hooke's law is valid. As such we can easily obtain the needed stress measures in terms of the previously derived strain by simply multiplying the strain by the appropriate modulus of elasticity (Bathe 1982, p.336). In the case of the engineering strain we obtain the corresponding energetically conjugate engineering stress from the expression

$$\sigma = E \, \epsilon \tag{2-30}$$

where E is the modulus of elasticity.

For the Green-Lagrange strain we can obtain the energetically conjugate Second Piola-Kirchhoff stress from

$$\mathrm{S} = E\,\epsilon_G \tag{2-31}$$

Similarly for the log-strain we obtain the true stress as

$$\sigma_L = E\,\epsilon_L \tag{2-32}$$

Now that the four most common strain and stress measures have been identified, they must be compared. This can be achieved by once again referring to the truss example, based on Crisfield's work (1991, p.61), given in Fig. 2-6. The vertical forces can be given as

$$P = \frac{A_1\,\text{'}\sigma\text{'}\;W}{L_1} \tag{2-33}$$

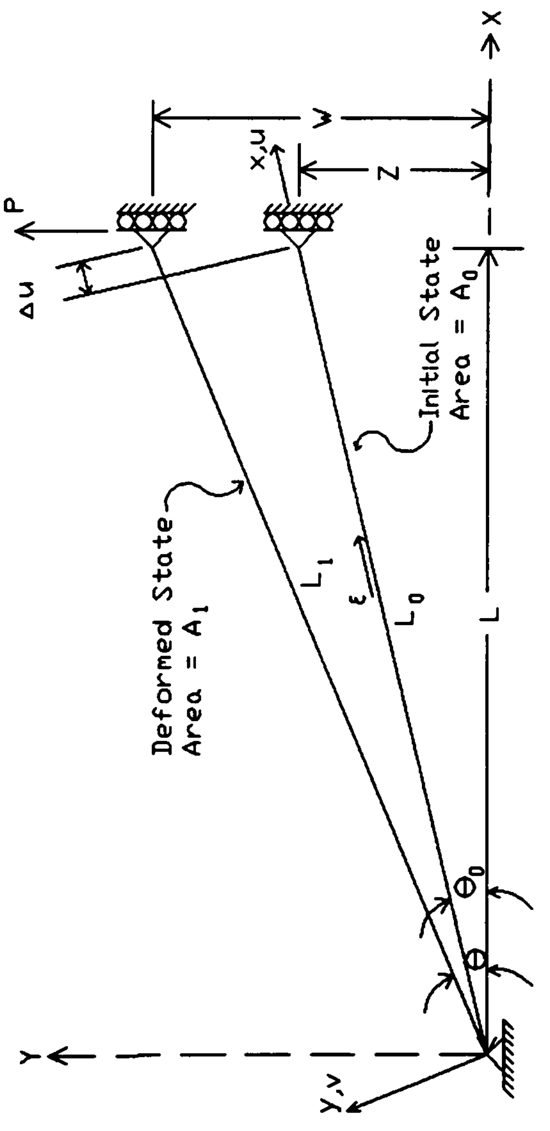


Figure (2-6) One Degree of Freedom Example where 'σ' is the Cauchy or true stress. Since the true strain is the log strain, it follows that the log stress is the true stress; thus

$$\sigma_L = \text{'}\sigma\text{'} \tag{2-34}$$

When considering the rotated engineering stress, the vertical force from Fig. 2-6 is given in Crisfield (1991, p.58) as

$$P = \frac{\sigma A_0 W}{L_1} \tag{2-35}$$

From equations 2-19 and 2-21, the comparison between rotated engineering stress and Cauchy stress can be stated as

$$\sigma = \text{'}\sigma\text{'}\left(\frac{A_1}{A_0}\right) \tag{2-36}$$

A similar comparison for the Second Piola-Kirchhoff stress follows from the resolution of the vertical forces of the truss as

$$P = \frac{\sigma_G A_0 W}{L_0} \tag{2-37}$$

Using equations 2-33 and 2-37, a comparison between the Second Piola-Kirchhoff stress and the Cauchy stress can be given as

$$\sigma_G = \text{'}\sigma\text{'} \left(\frac{A_1 L_0}{A_0 L_1} \right) \tag{2-38}$$

Similarly, a comparison between Second Piola-Kirchhoff stress and rotated engineering stress can be given as

$$\sigma_G = \sigma \left(\frac{L_0}{L_1} \right) \tag{2-39}$$

Now that some common strain measures have been discussed, a suitable pairing of strains and stresses for use in the nonlinear finite element equations of motion must be arrived at. The single most important factor affecting the choice of the appropriate material response pair is the reference frame used in the finite element formulation. In the case of structural analysis, the Lagrangian or material oriented reference frame is chosen. The Lagrangian reference frame is chosen over the Eulerian reference frame because in most analyses of solids and structures the boundaries of the solids are not continuously changing and hence new control volumes do not need to be continuously created as they would in a fluids problem (Bathe 1982, p.315). The most suitable strain measure for use within a Lagrangian reference frame is the Green-Lagrange strain (Bathe, 1975)(Martin 1973, p.166)(Oden, 1972)(Crisfield 1991, p.66). Since the Green-Lagrange strain is chosen as the appropriate strain measure, the Second Piola-Kirchhoff stress becomes the stress measure of choice. These two measures will form the foundation on which the principle of virtual work can be applied to obtain the nonlinear equilibrium equation.

2.6 Nonlinear Equilibrium Equation

The principle of virtual work can be given as follows (Holzer, 1985):

$$\delta W_e - \delta U = 0 \quad \overset{s.c.}{\rightarrow} \quad EQUILIBRIUM \tag{2-40}$$

where δW_e is the virtual work produced by the external forces acting on the structure, and δU is the first variation of the strain energy. The first variation of the strain energy may be stated more explicitly as

$$\delta U = \iiint_V \delta\epsilon^T \sigma \, dV \tag{2-41}$$

where σ is assumed to be an arbitrary stress measure and similarly ϵ is assumed to be an arbitrary strain measure. It is implied in equation 2-41 that the final deformed configuration of the structure is used as the basis for computing the magnitudes and directions of the corresponding stress and strain measures. In the case of the nonlinear finite element analysis this assumption constitutes a problem because the configuration of the structure in the deformed state is unknown. Therefore material response measures used in equation 2-38 must be referred to a previous configuration. It now becomes apparent why a Lagrangian reference frame is chosen for the formulation of the nonlinear finite element equilibrium equations. A Lagrangian reference frame is always referred to a previous equilibrium configuration and as such is quite suitable for use in nonlinear finite element problems. The most common choice for the material response pair of equation 2-41 is the Green-Lagrange strain and the Second Piola-Kirchhoff stress. This is the case since both measures represent the material response in the deformed state referred back to a previously computed equilibrium configuration, and both measures accommodate large rotations. Focussing on the referral to previous equilibrium configurations, the Lagrangian reference frames are discussed. In the case of nonlinear finite element formulations, there are two fundamental Lagrangian reference frames, the Total and the Updated. A discussion of the two reference frames and how they differ from each other follows.

In the literature there are a number of accounts which describe the difference between the Total and Updated Lagrangian reference frames (Bathe, 1975 ; Wunderlich, 1977). These presentations an account of the two reference frames in the most general terms possible. However, sometimes a better understanding of a concept can be gained by applying a general concept to a simple example.

In this presentation, the Total and Updated Lagrangian reference frames will be explained within the framework of the truss element. When a truss element assumes a configuration under a set of externally applied loads it does not do so instantaneously. It passes through many states along the way to the final state of equilibrium. This passage between states is marked by differing geometric configurations. In other words, the truss does not instantly change from its unloaded state to its deformed loaded state. It passes through many intermediate deformed states marked by differing degrees of deformation and rotation (figure 2-5).

In an incremental nonlinear finite element analysis these interim states are simulated through the process of iteration over the unbalanced loads. Each time a new displacement increment is computed and added to the previous displacements, a new configuration of the truss is defined. These configurations mark changes in the response of the truss as it develops internal forces to equilibrate the externally applied loads.

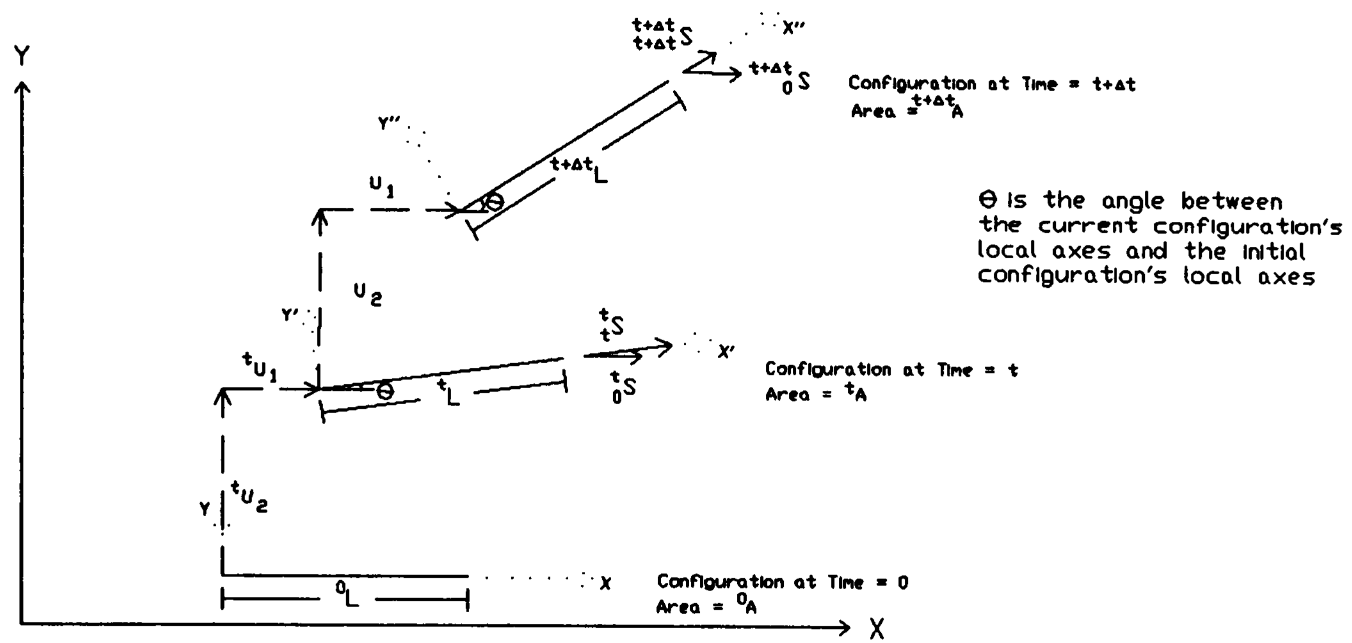
2.7 The Total Lagrangian

In the case of the Total Lagrangian reference frame, all material responses in the current deformed state are referred back to the *initial* configuration. The initial configuration of the truss is defined as the configuration prior to the first load step. In the case of the truss element in figure 2-7, the initial state is defined as the configuration at time 0. At time 0 the truss has the length ${}^{0}L$, the area ${}^{0}A$, and the volume ${}^{0}V$. The local coordinate system at time 0 serves as the basis for evaluating the directional responses of the truss in subsequent configurations. From this it is apparent that the Green-Lagrange strain and Second Piola-Kirchhoff stress, given in equations 2-20 and 2-31 respectively, must have the initial configuration as their reference state. When one begins speaking in terms of referrals it is important to adopt a concise referral notation. For the remainder of this thesis Bathe's (1982, p.317) notation for referral will be used. The case of the Green-Lagrange strain in the Total Lagrangian reference frame would be given as ${}^{t}_{0}\epsilon$. This would be read as the Green-Lagrange strain at time t with respect to the configuration at time 0. The left superscript denotes the configuration of interest, and the left subscript denotes the reference configuration. Based on the described notation, equation 2-41 can be rewritten in a form suitable for the Total Lagrangian as

$$\iiint_{{}^{0}V} {}^{t}_{0}S_{ij} \; \delta^{t}_{0}\epsilon_{ij} \, d^{0}V \; - \; \delta W_e \; = \; 0 \tag{2-42}$$

The physical meaning of the referred material responses can be seen in Fig. 2-7. Equation 2-42 can be written in matrix form as

$$\iiint_{{}^{0}V} \delta^{t}_{0}\epsilon^{\,T} \; {}^{t}_{0}S \; d^{0}V \; - \; \delta W_e \; = \; 0 \tag{2-43}$$



$^{t+\Delta t}_{t+\Delta t}S$
X''
$^{t+\Delta t}_{0}S$
Configuration at Time = t+Δt
Area = $^{t+\Delta t}A$
Y''
U_1
$^{t+\Delta t}L$
Θ
Θ is the angle between the current configuration's local axes and the initial configuration's local axes
U_2
Y'
$^{t}_{t}S$
X'
$^{t}U_1$
Θ
^{t}L
$^{t}_{0}S$
Configuration at Time = t
Area = ^{t}A
Y
$^{t}U_2$
X
Configuration at Time = 0
Area = ^{0}A
^{0}L
Y
X

NOTE: Θ is used for computing the direction cosines necessary for referral to the initial state,

Figure (2-7) The Total Lagrangian Reference Frame

Note that ${}^{0}V$ is the volume of the truss in its initial configuration. In a Total Lagrangian reference frame, all integrations are performed with respect to the initial configuration.

The generation of the tangent stiffness matrix in the Total Lagrangian reference frame requires taking the partial derivative of the internal force vector with respect to the displacements

$$K_T = \frac{\partial F}{\partial u} \tag{2-44}$$

where F is the internal force vector and u is the vector of global displacements.

The characteristics of the Total Lagrangian reference frame which have been thus far mentioned are only the details necessary to understand the processes occurring in the nonlinear solution within this reference frame. To fully understand the referral process it must be implemented. In the discussion to follow the example of a truss is used. For the case of the truss, the strain is a scalar, so at time $t+\Delta t$ it can be stated that

$${}^{t+\Delta t}_{0}\epsilon = {}^{t}_{0}\epsilon + {}_{0}\epsilon \tag{2-45}$$

where ${}_0\epsilon$ is the increment in Green-Lagrange strain between time t and time $t+\Delta t$. Similarly, the Second Piola Kirchhoff stress at time $t+\Delta t$ is given as

$$ {}_0^{t+\Delta t}S = {}_0^{t}S + {}_0S \tag{2-46}$$

where ${}_0S$ is the increment in the Second Piola-Kirchhoff stress between time t and time $t+\Delta t$. The fact that the Green-Lagrange strain and Second Piola-Kirchhoff stress, at time t, are referred to the configuration at time 0 is clear. What may not be clear however is the fact that the increment in these material response measures, between time t and time $t+\Delta t$, are also referred back to the initial configuration. This means that the directions of the incremental material responses coincide with those of the reference configuration hence the expressions of equations 2-45 and 2-46 are possible.

2.8 The Updated Lagrangian

→ Referred to the current config !

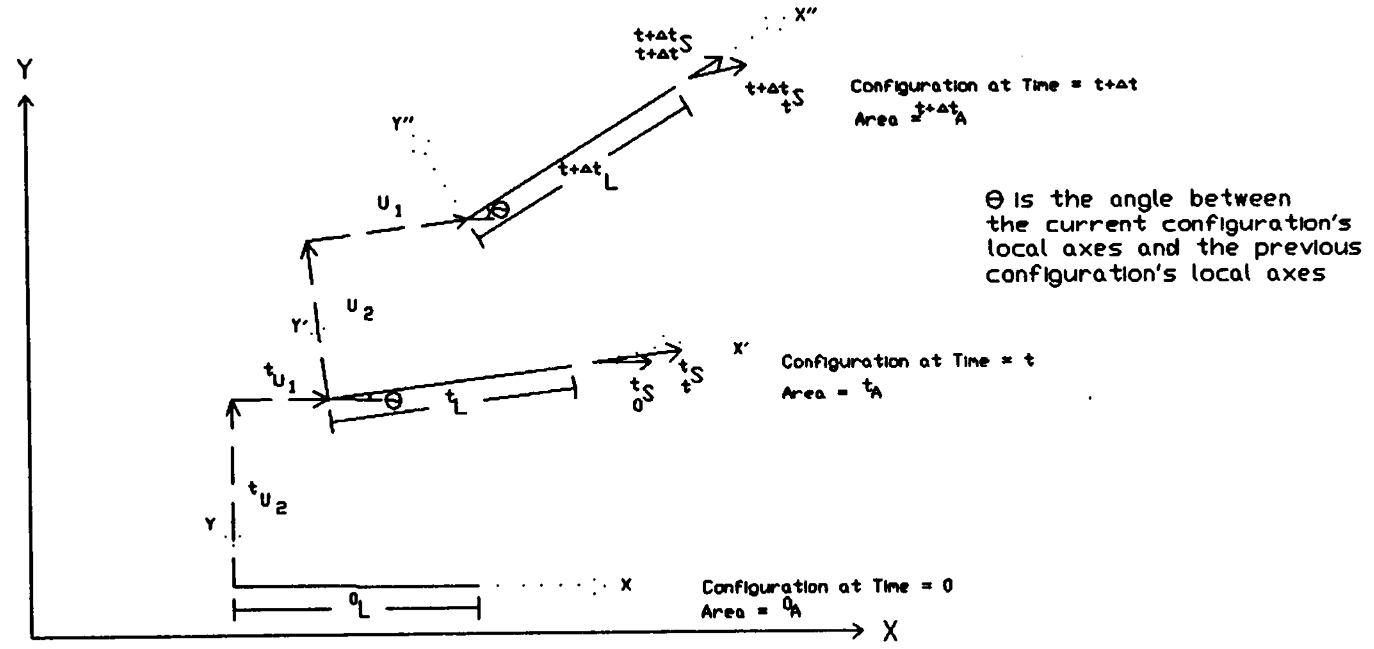
When using the Updated Lagrangian reference frame it is important to realize that the material response of the truss in the current configuration is referred back to the *previous* equilibrium configuration. Since this referral exists, the use of the Green-Lagrange strain and Second Piola-Kirchhoff stress is again warranted. An important characteristic of the Second Piola-Kirchhoff stress is that when it is measured in the same configuration in which it is occurring, it results in the true stress (Bathe 1982, p.341 ; Crisfield 1991, p.59). Referring once again to the truss, this point is expressed by the following equation:

$$ {}^{t}_{t}S = {}_{t}\sigma_{L} \qquad (2\text{-}47) $$

where σ_L is the true stress and the subscript t refers to the configuration of interest, in this case time t. This is an important aspect of the Updated Lagrangian reference frame.

In figure 2-8 the truss element is seen in three distinct configurations. The first configuration, corresponding to time 0, is the configuration of the truss before the first load step. The second configuration corresponds to time t which is the previous equilibrium configuration relative to time $t+\Delta t$. The configuration at time $t+\Delta t$ is the next equilibrium configuration of the structure and is designated as the current configuration. The equilibrium configurations and material responses of the truss at times 0 and t have been previously calculated and are known. The focus of this discussion is to determine the response of the truss in the current configuration.

The starting point of this discussion lies in the determination of the state of stress at time t. As mentioned earlier, in the Updated Lagrangian reference frame, the previous equilibrium configuration of the truss is the reference state for the new configuration. In the case of the truss at time t, the reference state was the configuration at time 0. Since this is the case, the Second Piola-Kirchhoff stress at time t is given in reference to the initial configuration. This referred



Θ is the angle between the current configuration's local axes and the previous configuration's local axes

NOTE: Θ is used for computing the direction cosines necassary for referral to the previous state.

Figure (2-8) The Updated Lagrangian Reference Frame stress must be converted into true stress in order that the configuration at time t may serve as the reference state for the configuration at time $t+\Delta t$. To this end, the following equation, given by Crisfield (1991, p.73) is employed for the truss:

$$ {}_t\sigma = \frac{{}^0A\,{}^tL}{{}^tA\,{}^0L}\,{}_0^tS \tag{2-48} $$

Equation 2-48 allows the Second Piola-Kirchhoff stress at time t with respect to time 0 to be converted to the Cauchy stress at time t. This fact allows the following crucial equation to be formulated:

$$ {}_t^{t+\Delta t}S = {}_t^t\sigma + {}_tS \tag{2-49} $$

where ${}_tS$ is the increment in the Second Piola-Kirchhoff stress between the configuration at time t and the configuration at time $t+\Delta t$. Using the results from above, the equilibrium equation for the Updated Lagrangian may be stated as

$$ \iiint_{{}^tV} \delta_t^{t+\Delta t}\epsilon^T \; {}_t^{t+\Delta t}S \; d^tV \tag{2-50} $$

It should be noted that all integrations take place over the volume of the previous configuration. Furthermore, the local reference frame of the previous equilibrium configuration serves as the reference frame used in computing the increments in material response from the previous configuration to the new configuration.

2.9 Finite Element Formulation

The logical starting point for any discussion of continuum mechanics based incremental nonlinear finite element notation is the representation of the Green-Lagrange strain tensor. The Green-Lagrange strain tensor can be decomposed into two distinct components, one component being linear in displacements and the other being quadratic in displacements. Wood and Schrefler (1976) express this decomposition with the equation

$$\epsilon = \epsilon_0 + \epsilon_L \tag{2-51}$$

where ϵ_0 is the component of the Green-Lagrange strain linearly dependent on displacements, while ϵ_L is the component quadratically dependent on displacements. Bathe (1975) has a similar decomposition given for the increment as follows:

$$ {}_0\epsilon = {}_0e + {}_0\eta \qquad (2\text{-}52) $$

where ${}_0e$ is the component of the Green-Lagrange strain increment linearly dependent on displacements, while ${}_0\eta$ is the component quadratically dependent on displacements.

Focussing on the B-notation of Zienkiewicz, the Green-Lagrange strain may be re-expressed using an expanded form for the nonlinear strain component

$$ \epsilon = \epsilon_0 + \frac{1}{2} \boldsymbol{A} \, \boldsymbol{\theta} \qquad (2\text{-}53) $$

where

$$ \boldsymbol{\theta} = \boldsymbol{G} \, \boldsymbol{u} \qquad (2\text{-}54) $$

and $\boldsymbol{G}$ is a constant matrix composed of differentials of interpolation functions. It should be mentioned that the matrix $\boldsymbol{G}$ of Zienkiewicz's notation corresponds to the matrix $\boldsymbol{B}_{NL}$ in Bathe's notation. The vector $\boldsymbol{u}$ appearing in equation 2-54 is the vector of nodal elemental displacements. It should be further noted that both $\boldsymbol{A}$ and $\boldsymbol{\theta}$ are displacement gradient vectors, where $\boldsymbol{\theta}$ contains all the differentials occurring in the nonlinear strain-displacement relationship and A is constructed such that when post-multiplied by θ and scalar multiplied by $\frac{1}{2}$, the nonlinear portion of the Green-Lagrange strain tensor is obtained. If we now give an expanded form for the linear portion of the Green-Lagrange strain, we may write the strain-displacement relation as

$$\epsilon = L^T G u + \frac{1}{2} A G u \tag{2-55}$$

In equation 2-55, certain submatricies may be defined as follows:

$$B_0 = L^T G \tag{2-56}$$

L is a vector constructed such that when postmultiplied by G and u it yields the linear portion of the strain-displacement matrix,

and

$$B_L = A G \tag{2-57}$$

Thus, the Green-Lagrange strain-displacement may be represented in the B-notation as

$$\epsilon = \left(B_0 + \tfrac{1}{2}B_L\right) u \qquad (2\text{-}58)$$

The matrix B_0 of Zienkiewicz corresponds to the matrix B_{L0} of Bathe, and the matrix B_L of Zienkiewicz has no equivalent in Bathe's notation. Now that the finite element form of the Green-Lagrange strain tensor has been determined, the variation of this quantity can be given as follows:

$$\delta\epsilon = \left(B_0 + B_L\right) \delta u \qquad (2\text{-}59)$$

This variational form of the Green-Lagrange strain tensor is important since it occurs in the volume integration necessary for the computation of the internal force vector.

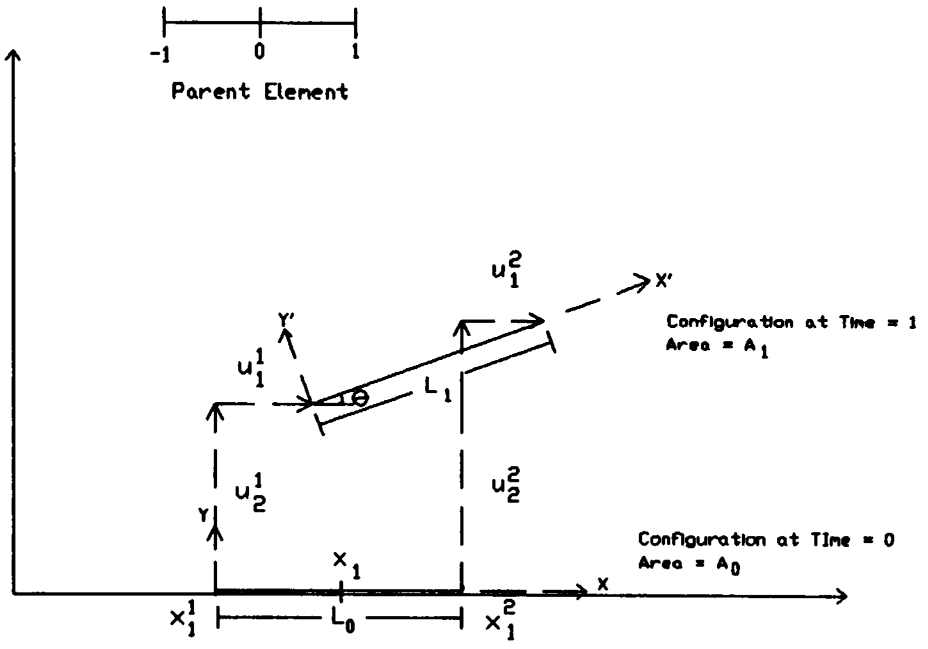
Chapter 3
The Geometrically Nonlinear Plane Truss

3.1 Introduction

A two-dimensional geometrically nonlinear truss element. Two formulations of this element is formulated. One formulation will be carried out in the B-notation of Zienkiewicz, as given by Wood and Schrefler (1978), while the other will follow the method outlined by Bathe et al. (1975).

3.2 The B-notation

The B-notation will be used for the first formulation of the nonlinear plane truss element. This formulation is accompanied by Fig. 3-1 in order that some of the relevant parameters of the formulation may be displayed.



Θ is the angle between the current configuration and the initial configuration

Figure (3-1) The Nonlinear Plane Truss Element for the B-notation

3.2.1 The Strain-displacement Relationship

The root of the nonlinearity in this element lies in the Green-Lagrange strain-displacement relationship given in the index notation as

$$\epsilon_G = u_{1,x} + \frac{1}{2}\left\{(u_{1,x})^2 + (u_{2,x})^2\right\} \tag{3-1}$$

The linear and quadratic dependence of the Green-Lagrange strain-displacement relationship can be seen in equation 3-1.

3.2.2 Interpolation

The Lagrangian interpolation polynomials given as

$$N_1 = \frac{1}{2}\ (1-\xi) \tag{3-2}$$

$$N_2 = \frac{1}{2}\ (1+\xi) \tag{3-3}$$

are used for both coordinate and displacement interpolation stated as

$$x_1 = \sum_{i=1}^{2} N_i \; x_1^i \tag{3-4}$$

and

$$u_1 = \sum_{i=1}^{2} N_i \; u_1^i \tag{3-5}$$

$$u_2 = \sum_{i=1}^{2} N_i \; u_2^i \tag{3-6}$$

where x_1^i and u_1^i, u_2^i are the nodal coordinates and nodal displacements respectively. These quantities are given visually in Fig. 3-1. The displacement field of the truss is given as

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} N_1 & 0 & N_2 & 0 \\ 0 & N_1 & 0 & N_2 \end{bmatrix} \begin{bmatrix} u_1^1 \\ u_2^1 \\ u_1^2 \\ u_2^2 \end{bmatrix} \tag{3-7}$$

where the vector of lower case u's corresponds to the local displacement vector. The right superscript identifies the node of interest while the right subscript identifies the direction in the local reference frame being considered. The result of the matrix operation in equation 3-7 is the vector composed of u_1 and u_2 which are the horizontal and vertical displacements respectively, see Fig. 3-1. In order to accommodate mapping between the parent element and the finite element, and vice versa, it is necessary to evaluate the determinant of the Jacobian matrix. In the case of the nonlinear plane truss element this determinant has the form

$$|J| = \frac{L_0}{2} \tag{3-8}$$

Thus,

$$dx_1 = \frac{L_0}{2}\, d\xi \tag{3-9}$$

Equation 3-9 may be verified as follows. Taking the first derivative of the interpolation functions with respect to ξ yields

$$\frac{dN_1}{d\xi} = -\frac{1}{2} \tag{3-10}$$

$$\frac{dN_2}{d\xi} = \frac{1}{2} \tag{3-11}$$

Equations 3-10 and 3-11 can then be used in defining the first derivative of x with respect to ξ as

$$\frac{dx_1}{d\xi} = \left(\frac{dN_1}{d\xi} x_1^1 + \frac{dN_2}{d\xi} x_1^2 \right) \tag{3-12}$$

so,

$$\frac{dx_1}{d\xi} = \frac{1}{2} \left(-x_1^1 + x_1^2 \right) \tag{3-13}$$

and since in the case of the truss element,

$$L_0 = \left(x_1^2 - x_1^1\right) \tag{3-14}$$

thus equation 3-9 is verified by the combination of equations 3-13 and 3-14:

$$\frac{dx_1}{d\xi} = \frac{L_0}{2} \tag{3-15}$$

3.2.3 The Strain-displacement Matrices

Having identified the form of the Green-Lagrange strain displacement relationship, the strain-displacement matrices can now be constructed so that a discretized form of the strain may be obtained. The first matrix to be constructed is the matrix of Cartesian derivatives of interpolation functions known as the *G* matrix. In the case of the plane truss, it has the following form:

$$G = \frac{1}{L_0}\begin{bmatrix} -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix} \tag{3-16}$$

The matrix of displacement gradients known as the $\boldsymbol{A}$ matrix has the form

$$A^T = \begin{bmatrix} \dfrac{u_1^2 - u_1^1}{L_0} \\ \dfrac{u_2^2 - u_2^1}{L_0} \end{bmatrix} \tag{3-17}$$

Similarly, the vector of displacement gradients known as the $\boldsymbol{\theta}$ matrix has the form

$$\boldsymbol{\theta} = \begin{bmatrix} \dfrac{u_1^2 - u_1^1}{L_0} \\ \dfrac{u_2^2 - u_2^1}{L_0} \end{bmatrix} \tag{3-18}$$

Based on what has been presented above, it is now possible to give the form of the strain-displacement matrices.

The first strain-displacement matrix to be considered is the $\boldsymbol{B}_0$ matrix. The $\boldsymbol{B}_0$ matrix is used to construct the first term of equation 3-1. This term is known as the engineering strain and as such has only linear dependence on displacements. Since this is the case the $\boldsymbol{B}_0$ matrix has no dependence on displacements and is given as

$$B_0 = \frac{1}{L_0} \begin{bmatrix} -1 & 0 & 1 & 0 \end{bmatrix} \tag{3-19}$$

The second strain-displacement matrix is responsible for constructing all other terms present in equation 3-1. This matrix is known as the $\boldsymbol{B}_L$ matrix and has a linear dependence on displacements. The matrix can be constructed using the previously defined matrices $\boldsymbol{A}$ and $\boldsymbol{G}$:

$$\boldsymbol{B}_L = \boldsymbol{A}\,\boldsymbol{G} \tag{3-20}$$

The composition of the $\boldsymbol{B}_L$ matrix is further defined as

$$\boldsymbol{B}_L = \frac{1}{L_0^2} \left[\left(u_1^1 - u_1^2\right) \; \left(u_2^1 - u_2^2\right) \; \left(u_1^2 - u_1^1\right) \; \left(u_2^2 - u_2^1\right) \right] \tag{3-21}$$

The two strain-displacement matrices given in equations 3-19 and 3-21 can be combined in the following manner to obtain the discretized form of the Green-Lagrange strain tensor as

$$\epsilon_G = \left(\boldsymbol{B}_0 + \tfrac{1}{2}\,\boldsymbol{B}_L\right) \boldsymbol{u} \tag{3-22}$$

The variation of the Green-Lagrange strain tensor may be expressed as

$$\delta \epsilon_G = \left(B_0 + B_L \right) \delta u \tag{3-23}$$

3.2.4 The Internal Force Vector

The above matrices can be used in the finite element equilibrium equation given in the B-notation as

$$\delta u^T \int \int \int_{{}^0V} B^T \sigma \, d^0V - \delta u^T F = 0 \tag{3-24}$$

where

$$B = \left(B_0 + B_L \right) \tag{3-25}$$

and the vector F is the internal force vector. Equation 3-24 may be further expanded to obtain the finite element equilibrium equation in the form of

$$\delta u^T \iiint_{^0V} B_0{}^T D\ B_0 + B_L{}^T D\ B_0 + \tfrac{1}{2} B_0{}^T D\ B_L$$

$$+ \tfrac{1}{2} B_L{}^T \mathrm{D}\ B_L \quad d^0V\, u \quad - \quad \delta u^T R \ = 0 \qquad (3\text{-}26)$$

The integrand of equation 3-26 produces the unsymmetric form of the secant stiffness matrix as given by Wood and Schrefler (1978). This secant stiffness matrix, when multiplied by the nodal displacement vector, yields the element internal force vector. The internal force vector plays the key role in the solution of the nonlinear finite element equations. Also figuring prominently in the the nonlinear solution process are the tangent stiffness matrices.

3.2.5 The Tangent Stiffness Matrix

In the case of the nonlinear plane truss finite element formulation given in here, only the linear and the initial stress portion of the incremental stiffness matrix are included. The linear portion of the tangent stiffness matrix, often referred to as K_0, is the stiffness matrix which is used in linear analysis. The familiar form of the matrix can be seen below:

$$K_0 = \gamma \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3-27}$$

where

$$\gamma = \frac{AE}{L_0} \tag{3-28}$$

A being the cross sectional area of the truss and E being the modulus of elasticity of the material of which the truss is made. Similarly, the initial stress matrix K_σ can be given as

$$K_\sigma = \frac{P}{L_0} \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix} \tag{3-29}$$

where P is the axial force in the truss element given as

$$P = \gamma \left(u_1^2 - u_1^1 \right) \tag{3-30}$$

3.3 Bathe's Formulation

The fundamental basis for the continuum mechanics based incremental nonlinear finite element formulation presented by Bathe et al. (1975) is the incremental decomposition of the Green-Lagrange strain and the Second Piola-Kirchhoff stress:

$$ {}^{t+\Delta t}_{0}\epsilon_G = {}^{t}_{0}\epsilon_G + {}_{0}\epsilon_G \tag{3-31} $$

$$ {}^{t+\Delta t}_{0}\mathrm{S} = {}^{t}_{0}\mathrm{S} + {}_{0}\mathrm{S} \tag{3-32} $$

The notation used in equations 3-31, 3-32, and in the rest of this section is that given by Bathe (1982). The left subscript on the quantity in question refers to the configuration that the quantity was measured in while the left superscript refers to the configuration the quantity is actually occurring in. For instance, the quantity ${}^{t+\Delta t}_{0}\mathrm{S}$ is read as the Second Piola-Kirchhoff stress occurring in the element at the time $t+\Delta t$ but coincident with the local axes of the element configuration at time 0. For the case where only a left subscript appears in conjunction with the quantity in question, the increment from time t to time $t+\Delta t$, measured in the configuration of the subscript, is being described. As an example, consider the term ${}_{0}\mathrm{S}$ from equation 3-32; this can be read as the increment in the Second Piola-Kirchhoff stress from time t to time $t+\Delta t$

measured with respect to the local axes of the initial configuration of the element. This type of notation can be used to re-express the Green-Lagrange strain displacement relationship as

$$ {}^{t+\Delta t}_{0}\epsilon_{11} = {}^{t+\Delta t}_{0}u_{1,1} + \frac{1}{2}[({}^{t+\Delta t}_{0}u_{1,1})^2 + ({}^{t+\Delta t}_{0}u_{2,1})^2] \qquad (3\text{-}33)$$

where

$$ {}^{t+\Delta t}_{0}u_{1,1} = \frac{\partial\ {}^{t+\Delta t}u_1}{\partial\ {}^{0}x_1} \qquad (3\text{-}34)$$

Based on the fact that the displacement at time $t+\Delta t$ can be expressed as

$$ {}^{t+\Delta t}_{0}u = {}^{t}_{0}u + {}_{0}u \qquad (3\text{-}35)$$

equation 3-33 can be re-written as

$$ {}^{t+\Delta t}_{0}\epsilon_{11} = {}^{t}_{0}u_{1,1} + {}_{0}u_{1,1} + \frac{1}{2}\left[({}^{t}_{0}u_{1,1} + {}_{0}u_{1,1})^2 + ({}^{t}_{0}u_{2,1} + {}_{0}u_{2,1})^2 \right] \qquad (3\text{-}36)$$

Equation 3-36 can be further expanded to obtain

$$
{}_0^{t+\Delta t}\epsilon_{11} = \underbrace{{}_0^{t}u_{1,1} + \frac{1}{2}\left[({}_0^{t}u_{1,1})^2 + ({}_0^{t}u_{2,1})^2 \right]}_{{}_0^{t+\Delta t}\epsilon_{11}} +
$$

$$
\underbrace{{}_0u_{1,1} + {}_0^{t}u_{1,1}\ {}_0u_{1,1} + {}_0^{t}u_{2,1}\ {}_0u_{2,1}}_{{}_0e_{11}} +
$$

$$
\underbrace{\frac{1}{2}\left[({}_0u_{1,1})^2 + ({}_0u_{2,1})^2 \right]}_{{}_0\eta_{11}} \tag{3-37}
$$

where e and η are the portions of the Green-Lagrange strain displacement relationship linearly and quadratically dependent on displacement respectively.

3.3.1 Interpolation

In Bathe's formulation of the nonlinear plane truss element the same linear Lagrangian interpolation polynomials as used in the B-notation are implemented.

Bathe uses a different choice of variables so the interpolation polynomials can be restated as

$$N_1 = \frac{1}{2}(1 - \xi) \tag{3-38}$$

$$N_2 = \frac{1}{2}(1 + \xi) \tag{3-39}$$

The differentials of the interpolation polynomials are given as

$$\frac{dN_1}{d\xi} = -\frac{1}{2} \tag{3-40}$$

$$\frac{dN_2}{d\xi} = \frac{1}{2} \tag{3-41}$$

Based on the foregoing, the coordinate interpolation may be given as

$${}^0x_1 = \sum_{k=1}^{2} N_k \; {}^0x_1^k \tag{3-42}$$

$$\frac{d\,{}^{0}x_1}{d\,r} = \frac{1}{2}\left(- {}^{0}x_1^1 + {}^{0}x_1^2 \right) = \frac{1}{2}\,{}^{0}L = \frac{1}{2}\,L \tag{3-43}$$

and the displacement interpolation may be given as

$$u_i = \sum_{k=1}^{2} N_k\, u_i^k \tag{3-44}$$

and

$$^{t}u_i = \sum_{k=1}^{2} N_k\; {}^{t}u_i^k \tag{3-45}$$

3.3.2 Strain-displacement matrices

Based on what has been given with regards to the interpolation polynomials and the Green-Lagrange strain-displacement relationship, the discretized form of the strain-displacement relation can be given in the following matrix form. From the chain-rule of calculus it can be said that for the displacement increment,

$$\frac{\partial u_i}{\partial\, {}^0x_i} = \frac{\partial u_i}{\partial \xi}\,\frac{d\xi}{d\, {}^0x_1} = \frac{\partial u_i}{\partial \xi}\,\frac{2}{L} \tag{3-46}$$

which leads to

$$\frac{\partial u_i}{\partial\, {}^0x_1} = \frac{1}{L}\left(-u_i^1 + u_i^2 \right) \tag{3-47}$$

This can be expressed in matrix form as

$$\frac{\partial u_1}{\partial\, {}^0x_1} = \underbrace{\frac{1}{L}\begin{bmatrix} -1 & 0 & 1 & 0 \end{bmatrix}}_{{}_0^t B_{L0}} \begin{bmatrix} u_1^1 \\ u_2^1 \\ u_1^2 \\ u_2^2 \end{bmatrix} \tag{3-48}$$

$$\frac{\partial u_2}{\partial\, {}^0x_1} = \frac{1}{L}\begin{bmatrix} 0 & -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_1^1 \\ u_2^1 \\ u_1^2 \\ u_2^2 \end{bmatrix} \tag{3-49}$$

Now changing the focus from the incremental displacements to the accumulated displacements from time 0 to time t, the following is obtained:

$$\frac{\partial\, {}^tu_i}{\partial\, {}^0x_1} = \frac{1}{L}\left(- {}^tu_i^1 + {}^tu_i^2 \right) \tag{3-50}$$

$$\frac{\partial\, {}^tu_1}{\partial\, {}^0x_1} = \frac{1}{L}\left(- {}^tu_1^1 + {}^tu_1^2 \right) \tag{3-51}$$

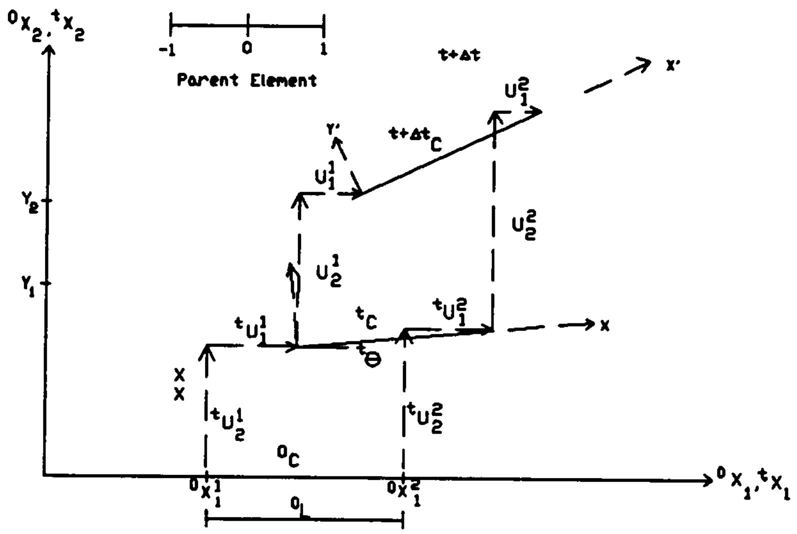
$$\frac{\partial\, {}^tu_2}{\partial\, {}^0x_1} = \frac{1}{L}\left(- {}^tu_2^1 + {}^tu_2^2 \right) \tag{3-52}$$

Based on the geometry of the truss shown in Fig. 3-2, equations 3-51 and 3-52 may be re-written as

$$\frac{\partial\, {}^tu_1}{\partial\, {}^0x_1} = \frac{1}{L}(L \cos{}^t\theta - L) = (\cos{}^t\theta - 1) \tag{3-53}$$

and

$$\frac{\partial\, {}^tu_2}{\partial\, {}^0x_1} = \frac{1}{L}(L \sin{}^t\theta) = \sin{}^t\theta \tag{3-54}$$



θ is the angle between the previous equilibrium configuration and the initial configuration.

Figure (3-2) The Nonlinear Plane Truss Element for Bathe's Notation the initial displacement portions of the incremental strain-displacement relationship can be given in matrix form as

$$\frac{\partial\, {}^{t}u_1}{\partial\, {}^{0}x_1}\frac{\partial\, u_1}{\partial\, {}^{0}x_1} = (\, \cos {}^{t}\theta - 1\,)\ \frac{1}{L}\begin{bmatrix} -1 & 0 & 1 & 0 \end{bmatrix}\{d\} \tag{3-55}$$

and

$$\frac{\partial\, {}^{t}u_2}{\partial\, {}^{0}x_1}\frac{\partial\, u_2}{\partial\, {}^{0}x_1} = \sin {}^{t}\theta\ \frac{1}{L}\begin{bmatrix} 0 & -1 & 0 & 1 \end{bmatrix}\{d\} \tag{3-56}$$

Assembling these matrices according to the algebraic initial displacement expression yields

$$\frac{\partial\, {}^{t}u_1}{\partial\, {}^{0}x_1}\frac{\partial\, u_1}{\partial\, {}^{0}x_1} + \frac{\partial\, {}^{t}u_2}{\partial\, {}^{0}x_1}\frac{\partial\, u_2}{\partial\, {}^{0}x_1} =$$

$$\underbrace{\frac{1}{L}\begin{bmatrix} -\cos {}^{t}\theta + 1 & -\sin {}^{t}\theta & \cos {}^{t}\theta - 1 & \sin {}^{t}\theta \end{bmatrix}}_{{}^{t}_{0}B_{L1}}\{d\} \tag{3-57}$$

where $\{d\}$ is the incremental displacement vector between the configurations at time t and $t+\Delta t$. Combining what has been given above, the linear portion of the Green-Lagrange strain in the increment may be given in matrix form as

$$ {}_0e = \underbrace{\left({}^t_0B_{L0} + {}^t_0B_{L1} \right)}_{{}^t_0B_L} \{d\} \tag{3-58} $$

where

$$ {}^t_0B_L = \frac{1}{L} \left[\ -\cos {}^t\theta \quad -\sin {}^t\theta \quad \cos {}^t\theta \quad \sin \theta \ \right] \tag{3-59} $$

The angle ${}^t\theta$ is the angle between the last known equilibrium configuration of the truss and the initial configuration of the truss (see Fig. 3-2).

3.3.3 Internal Force Vector

Using the the previously derived strain-displacement matrices, the internal force vector may, according to Bathe (1982), be presented as

$$ {}^{t}_{0}F = \int_{{}^{0}V} {}^{t}_{0}B_L^{T} \; {}^{t}_{0}\hat{S}_{11} \; d^{0}V \tag{3-60} $$

In general, the stress term appearing in equation 3-60 is a vector composed of components of the Second Piola-Kirchhoff stress tensor. For the case of the truss, however, the stress vector is reduced to one entry given by

$$ {}^{t+\Delta t}_{0}S_{11} = {}^{t}_{0}S_{11} + ({}_{0}e_{11} + {}_{0}\eta_{11}) E \tag{3-61} $$

where E is the modulus of elasticity of the material from which the truss is constructed.

The tangent stiffness matrices given earlier in this chapter, in the B-notation formulation, are equally valid for use in Bathe's notation; thus the matrices are not represented here.

3.4 Differences between Zienkiewicz's and Bathe's Formulation

When addressing the fundamental differences between the continuum mechanics based incremental nonlinear finite element formulations of Bathe and Zienkiewicz, it soon becomes apparent that differences are more far-reaching than the dissimilar notation. It is these fundamental differences, and how they impact the basic understanding of nonlinear finite element analysis, which makes them of interest.

Both notations agree as to the starting point in the nonlinear finite element formulation. This starting point is the previously mentioned variational statement of equilibrium

$$\iiint_{{}^0V} \delta\epsilon^T S \; \mathrm{d}^0V = \delta W_e \tag{3-62}$$

In equation 3-62 it is seen that the Total Lagrangian reference frame is being used. Throughout this discussion the Total Lagrangian will be used since it is the reference frame used in Ziekiewicz's formulation.

In the B-notation, equation 3-62 forms the foundation of the formulation with regards to the structural response. The internal force is arrived at by using the following equation:

$$\mathbf{F} = \iiint_{{}^0V} B^T S \, \mathrm{d}^0V \tag{3-63}$$

where

$$B = B_0 + B_L \tag{3-64}$$

From this point, it can then be said that the tangent stiffness matrix is obtained from Crisfield (1991, p.69) as

$$\mathrm{K}_T = \frac{\partial F}{\partial u} \tag{3-65}$$

where F is the internal force vector and u is the vector of local nodal displacements. Represented in matrix form, the tangent stiffness matrix of the B-notation is

$$\mathrm{K}_T = \mathrm{K}_0 + \mathrm{K}_\sigma + \mathrm{K}_L \tag{3-66}$$

The individual terms appearing in equation 3-66 are further expanded in the following:

$$\mathbf{K_0} = \iiint_{{}^0V} B_0{}^T E\, B_0 \; \mathrm{d}^0V \tag{3-67}$$

and E is the modulus of elasticity of the material. $\mathbf{K_0}$ is the familiar stiffness matrix used in linear analysis. The next stiffness term to be expressed is often referred to as the initial stress matrix. This is a fitting name since a stress term is present in every entry of the matrix:

$$\mathbf{K}_\sigma = \iiint_{{}^0V} G^T S\, G \; \mathrm{d}^0V \tag{3-68}$$

The third component matrix to be presented is the initial displacement matrix. This name is suitable due to the linear and quadratic dependence of this matrix on displacements:

$$\mathrm{K}_L = \int\int\int_{^0V} B_0{}^T E\, B_L\, \mathrm{d}^0V + \int\int\int_{^0V} B_L{}^T E\, B_0\, \mathrm{d}^0V + \int\int\int_{^0V} B_L{}^T E\, B_L\, \mathrm{d}^0V \tag{3-69}$$

The literature refers to the sum of K_0, K_σ, and K_L as the tangent stiffness matrix. Typically the K_L matrix is neglected since its effects are negligible except for cases of very large displacements. Thus

$$\mathrm{K}_T = \mathrm{K}_0 + \mathrm{K}_\sigma \tag{3-70}$$

As will be seen in the following presentation, Bathe also uses the ~~form of the~~ tangent stiffness matrix as given in equation 3-70.

As previously mentioned, Bathe uses the variational statement of equilibrium, given in equation 3-62, as the starting point for his formulation (1982, p.336). The notion of the incremental decomposition of the Second Piola-Kirchhoff stress marks the second major step in Bathe's formulation and is given as follows:

$${}^{t+\Delta t}_{\ 0}S = {}^{t}_{0}S + {}_{0}S \tag{3-71}$$

In conjunction with the incremental decomposition of the Second Piola-Kirchhoff stress tensor is the similar decomposition of the Green-Lagrange strain tensor stated as

$$ {}_0^{t+\Delta t}\epsilon = {}_0^{t}\epsilon + {}_0\epsilon \tag{3-72}$$

Since ${}_0^{t}\epsilon$ and ${}_0^{t}S$ represent the material response in the previous equilibrium configuration, they are considered as known quantities. Thus it can be said that

$$ \delta\, {}_0^{t+\Delta t}\epsilon = \delta\, {}_0\epsilon \tag{3-73}$$

Based on the above observations, the variational equilibrium equation can be rendered as

$$\iiint_{{}^0V} E\ {}_0\epsilon_{ij}\ \delta\, {}_0\epsilon_{ij}\ \mathrm{d}^0V\ +\ \iiint_{{}^0V} {}_0^{t}S_{ij}\ \delta\, {}_0\eta_{ij}\ \mathrm{d}^0V$$

$$\iiint_{{}^0V} {}_0^{t}S_{ij}\ \delta\, {}_0e_{ij}\ \mathrm{d}^0V\ =\ \delta W_e \tag{3-74}$$

where e and η are respectively the components of the Green-Lagrange strain increment linearly and quadratically dependent on the displacements. According to Bathe (1982, p.336)

$${}_0S = E\ {}_0\epsilon \qquad \text{(assumed !)} \tag{3-75}$$

Using the approximation of Bathe (1982, p.336)

$$\delta_0\epsilon = \delta_0 e \tag{3-76}$$

the variational equilibrium equation may be presented in its linearized form as follows (Bathe 1982, p.336):

$$\int\int\int_{{}^0V} E\,{}_0e_{ij}\,\delta_0e_{ij}\;\mathrm{d}^0V \;+\; \int\int\int_{{}^0V} {}^t_0S_{ij}\,\delta_0\eta_{ij}\;\mathrm{d}^0V \;=$$

$$\delta W_e \;-\; \int\int\int_{{}^0V} {}^t_0S_{ij}\;\delta_0\epsilon_{ij}\;\mathrm{d}^0V \tag{3-77}$$

The integrals on the ~~right~~ left hand side of equation 3-77 lead directly to the tangent stiffness matrix of equation 3-70. The integral on the ~~left~~ right hand side of the equation leads to the internal force vector. This last point is a crucial one since the internal force vector in equation 3-77 differs from Zienkiewicz's internal force vector given in equation 3-63. The difference lies in the variational Green-Lagrange strain term. In Bathe's formulation, the variation of the linear portion of the incremental strain plus an initial displacement effect is considered, while in the B-notation the full strain is used. This difference may be more readily apparent if the finite element form of the internal force vector is given in Bathe's notation

$$\int\int\int_{{}^0V} {}^{t+\Delta t}_{0}B_L{}^T\; {}^{t+\Delta t}_{0}\hat{S}\;\mathrm{dv}_0 \tag{3-78}$$

where

$${}^{t+\Delta t}_{0}B_L \;=\; {}^{t+\Delta t}_{0}B_{L0} \;+\; {}^{t+\Delta t}_{0}B_{L1} \tag{3-79}$$

B_{L0} being the linear portion of the strain displacement matrix and B_{L1} representing the initial displacement matrix.

When comparing equation 3-78 to equation 3-63, the previously mentioned difference in the internal force vector can be seen. This discrepancy in internal force vector composition may prove important since "in an incremental procedure with iteration over unbalanced loads only the accuracy of internal force evaluation is of importance for the accuracy of the final solution (Osterrieder and Ramm, 1987)." Thus the effect of the varying internal force vector composition on the accuracy of the solution obtained from the nonlinear finite element analysis must be investigated. This difference is rooted in the fact that differing forms of the variation in the Green-Lagrange strain are used. In the case of Zienkiewicz's formulation, the total variational term is given as

$$\delta\epsilon = \delta e + \delta\eta \qquad (3\text{-}80)$$

? Zienkiewicz ignored this term! see reference (15) or (1)

In the case of Bathe's formulation, the incremental variational term is of the form

after linearize

$$\delta_0\epsilon = \delta_0 e \qquad (3\text{-}81)$$

The variational term presented by Zienkiewicz is the the exact form. The form

bull shit

NO

given by Bathe is an approximation. In chapter 6 of this thesis this difference is shown to be insignificant when we consider the small displacements that occur in the increment.

For the readers information :

- Bathe's formulation is exactly the same as Zienkiewicz's

- Bathe linearized $\delta\varepsilon \to \delta e$ at the virtual work / equation of motion level

* Zienkiewicz linearized $\delta\varepsilon$ by ignoring $O(\delta D^2)$

$$\underset{\sim}{\varepsilon} = \frac{1}{2}\{\underset{\sim}{D} + \underset{\sim}{D}^T\} + \frac{1}{2}\underset{\sim}{D}^T\underset{\sim}{D}$$

$\uparrow$
tensor

$$\delta\underset{\sim}{e} = \frac{1}{2}\{\delta\underset{\sim}{D} + \delta\underset{\sim}{D}^T\} + \frac{1}{2}\delta\underset{\sim}{D}^T\underset{\sim}{D} + \frac{1}{2}\underset{\sim}{D}^T\delta\underset{\sim}{D} + \underbrace{O(\delta D^2)}_{\to \delta\eta \text{ term}} \to \text{ignored.}$$

* and they arrived at exactly the same formulation !
(try to expand all the matrices in Bathe's text, and you'll get exactly Zienkiewicz's matrices)
Note that the arrangement of the displacement matrix in these two formulations are not the same!

Chapter 4
The Geometrically Nonlinear Plane Frame Element

4.1 Introduction

Due to the complexity involved in performing the mathematical operations necessary for the formulation of the geometrically nonlinear Bernoulli-Euler plane frame element, the program MACSYMA was used. MACSYMA is a symbolic mathematics package which can perform many of the tedious mathematical operations arising in a formulation of this type.

The formulation of this element, carried out in the B-notation, is given in the following sections.

4.2 The Strain-displacement Relationship

The source of the nonlinearity in this element can be traced back to the Green-Lagrange strain-displacement relationship given approximately in the index notation as

$$\epsilon_G = u_{1,x} - y\,(u_{2,xx}) + \tfrac{1}{2}\,(u_{2,x})^2 \qquad (4\text{-}1)$$

The nonlinear dependence on displacements can be seen in equation 4-1. The approximation mentioned earlier is due to the fact that the term $\frac{1}{2}\,(u_{1,x})^2$ is neglected. This is a valid approximation except in cases of very large axial deformations.

4.3 Interpolation

A combination of linear Lagrangian and cubic Hermitian interpolation polynomials (Holzer 1990) is used in the formulation of this element and is given below as

$$N_1 = \tfrac{1}{2}(1 - \xi) \qquad (4\text{-}2)$$

$$N_2 = \tfrac{1}{4}(2 + \xi)(1 - \xi)^2 \qquad (4\text{-}3)$$

$$N_3 = \tfrac{L}{8}(1 + \xi)(1 - \xi)^2 \qquad (4\text{-}4)$$

$$N_4 = \tfrac{1}{2}(1+\xi) \tag{4-5}$$

$$N_5 = \tfrac{1}{4}(2-\xi)(1+\xi)^2 \tag{4-6}$$

$$N_6 = \tfrac{L}{8}(\xi-1)(1+\xi)^2 \tag{4-7}$$

The linear Lagrangian interpolation polynomials are used for coordinate interpolation expressed as

$$x_1 = \sum_{i=1,4} N_i\, x_1^k \tag{4-8}$$

where x_1 represents the nodal coordinates. For displacements, the linear Lagrangian and cubic Hermitian interpolation polynomials are used for the axial degrees of freedom while the cubic Hermitian interpolation polynomials are used for the transverse degrees of freedom. Based on this, the displacements may be expressed as

$$u_1 = \sum_{i=1}^{6} N_i \; u_1^k \tag{4-10}$$

and

$$u_2 = \sum_{i=1}^{6} N_i \; u_2^k \tag{4-11}$$

The previously mentioned quantities appearing in equations 4-8 to 4-11 may be observed in Fig. 4-1. Focusing on displacements, the displacement field of the plane frame can be given as

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} N_1 & -\frac{\partial N_2}{\partial \xi} y & -\frac{\partial N_3}{\partial \xi} y & N_4 & -\frac{\partial N_5}{\partial \xi} y & -\frac{\partial N_6}{\partial \xi} y \\ 0 & N_2 & N_3 & 0 & N_5 & N_6 \end{bmatrix} \begin{bmatrix} u_1^1 \\ u_2^1 \\ u_3^1 \\ u_1^2 \\ u_3^2 \\ u_3^2 \end{bmatrix} . \tag{4-11}$$

where the vector of u's corresponds to the local element nodal displacement vector. The result of the matrix operation in equation 4-11 is the vector composed of the horizontal displacement u_1 and the vertical displacement u_2 (see Fig. 4-1). In order to accommodate mapping between the parent element and the finite element, and vice versa, it is necessary to evaluate the determinant of the Jacobian matrix. In the case of the nonlinear Bernoulli-Euler plane frame element this determinant has the form

$$|J| = \frac{L}{2} \tag{4-12}$$

thus,

$$dx = \frac{L}{2}\, d\xi \tag{4-13}$$

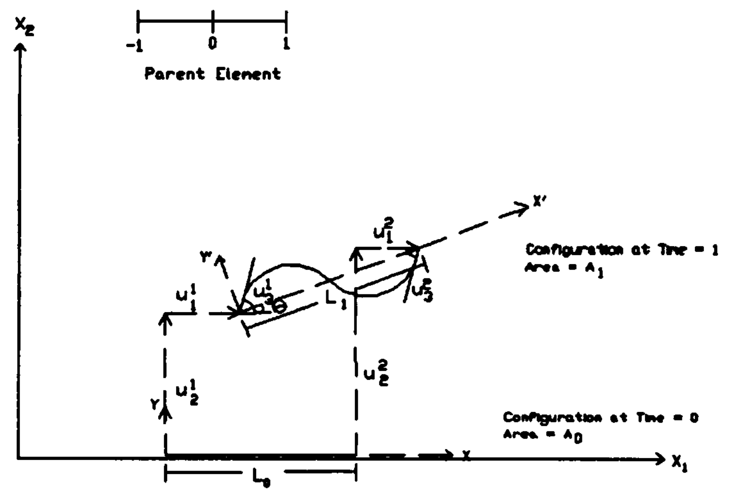
This was shown to be the case for the plane truss element in the previous chapter. The same presentation is also valid in the case of the Bernoulli-Euler plane frame.

4.4 The Strain-displacement Matrices

Based on the foregoing presentation, the Green-Lagrange strain-displacement relationship may be presented in discretized matrix form suitable for use with the nodal displacement vector. These strain-displacement matrices may be given in terms of various other sub-matrices to be presented below. The first of these sub-matrices to be presented is the matrix of Cartesian derivatives of interpolation functions known as the G matrix. In the case of the nonlinear plane frame element, this matrix has the form

$$G = \begin{bmatrix} \frac{2}{L}\frac{\partial N_1}{\partial \xi} & 0 & 0 & \frac{2}{L}\frac{\partial N_4}{\partial \xi} & 0 & 0 \\ 0 & \frac{2}{L}\frac{\partial N_2}{\partial \xi} & \frac{2}{L}\frac{\partial N_3}{\partial \xi} & 0 & \frac{2}{L}\frac{\partial N_5}{\partial \xi} & \frac{2}{L}\frac{\partial N_6}{\partial \xi} \\ 0 & \frac{4}{L^2}\frac{\partial^2 N_2}{\partial \xi^2} & \frac{4}{L^2}\frac{\partial^2 N_3}{\partial \xi^2} & 0 & \frac{4}{L^2}\frac{\partial^2 N_5}{\partial \xi^2} & \frac{4}{L}\frac{\partial^2 N_6}{\partial \xi^2} \end{bmatrix}$$

(4-14)



Θ is the angle between the local axes of the last known equilibrium configuration and the local axes of the initial configuration

Figure (4-1) The Nonlinear Plane Frame Element

The vector of displacement gradients $\boldsymbol{\theta}$ is given as

$$\boldsymbol{\theta} = \begin{bmatrix} \frac{2}{L}\frac{\partial N_1}{\partial \xi}d_1 + \frac{2}{L}\frac{\partial N_4}{\partial \xi}d_4 \\ \frac{2}{L}\frac{\partial N_2}{\partial \xi}d_2 + \frac{2}{L}\frac{\partial N_3}{\partial \xi}d_3 + \frac{2}{L}\frac{\partial N_5}{\partial \xi}d_5 + \frac{2}{L}\frac{\partial N_6}{\partial \xi}d_6 \\ \frac{4}{L^2}\frac{\partial^2 N_2}{\partial \xi^2}d_2 + \frac{4}{L^2}\frac{\partial N_3}{\partial \xi^2} + \frac{4}{L^2}\frac{\partial^2 N_5}{\partial \xi^2}d_5 + \frac{4}{L^2}\frac{\partial^2 N_6}{\partial \xi^2}d_6 \end{bmatrix} \tag{4-15}$$

The last of the sub-matrices to be presented is the $\boldsymbol{H}$ matrix. Its form is given by Rajasekaran (1973) as

$$\boldsymbol{H} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{4-16}$$

Based on the sub-matrices presented above, the strain-displacement matrices can now be constructed.

The first strain-displacement matrix considered is the $\boldsymbol{B}_0$ matrix. This matrix is used to construct the first term of the Green-Lagrange strain displacement relationship given in equation 4-1. This term is known as the engineering strain and as such has only linear dependence on displacements and can be given in terms of the interpolation polynomials as

$$B_0 = \left[\begin{array}{cccccc} \frac{2}{L}\frac{\partial N_1}{\partial \xi} & -\frac{4y}{L^2}\frac{\partial^2 N_2}{\partial \xi} & -\frac{4y}{L^2}\frac{\partial^2 N_3}{\partial \xi^2} & \frac{2}{L}\frac{\partial N_4}{\partial \xi} & -\frac{4y}{L^2}\frac{\partial^2 N_5}{\partial \xi^2} & -\frac{4y}{L^2}\frac{\partial^2 N_6}{\partial \xi^2} \end{array} \right]$$

(4-17)

The second strain-displacement matrix is responsible for constructing the portion of the Green-Lagrange strain which is quadratically dependent on displacements. This matrix may be constructed, according to Wood and Schrefler (1978), by using equations 4-14 through 4-16 as

$$B_L = \theta^T H \; G \qquad (4\text{-}18)$$

This may be re-expressed as

$$B_L = \left[\begin{array}{cccccc} 0 & \frac{2}{L}\frac{\partial N_2}{\partial \xi}\theta_2 & \frac{2}{L}\frac{\partial N_3}{\partial \xi}\theta_2 & 0 & \frac{2}{L}\frac{\partial N_5}{\partial \xi}\theta_2 & \frac{2}{L}\frac{\partial N_6}{\partial \xi}\theta_2 \end{array} \right]$$

(4-19)

The two strain-displacement matrices given in equations 4-17 and 4-19 can be combined in the following manner to obtain the discretized form of the Green-Lagrange strain tensor as

$$\epsilon_G = \left(\boldsymbol{B}_0 + \tfrac{1}{2}\, \boldsymbol{B}_L \right) \boldsymbol{u} \tag{4-20}$$

The variation of the Green-Lagrange strain tensor may be expressed as

$$\delta\epsilon_G = \left(\boldsymbol{B}_0 + \boldsymbol{B}_L \right) \boldsymbol{u} \tag{4-21}$$

4.5 The Internal Force Vector

The above matrices can be used in the finite element equilibrium equation given in the B-notation as

$$\delta \boldsymbol{u}^T \int\int\int_{{}^0V} \boldsymbol{B}^T \sigma \; \mathrm{d}^0V \; - \; \delta \boldsymbol{u}^T \boldsymbol{R} = 0 \tag{4-22}$$

where

$$B = \left(B_0 + B_L\right) \qquad (4\text{-}23)$$

and the vector R is a vector of equivalent nodal forces corresponding to externally applied tractions and body forces. Equation 4-22 may be further expanded to obtain the finite element equilibrium equation in the form of

$$\delta u^T \iiint_{{}^0V} B_0{}^T D\, B_0 + B_L{}^T D\, B_0 + \tfrac{1}{2} B_0{}^T D\, B_L$$

$$+ \tfrac{1}{2} B_L{}^T \mathrm{D}\, B_L \; \mathrm{d}^0V \;\; u \;\; - \;\; \delta u^T R = 0 \qquad (4\text{-}24)$$

The integrand of equation 4-24 produces the unsymmetric form of the secant stiffness matrix as given by Wood and Schrefler (1978). This secant stiffness matrix, when multiplied by the nodal displacement vector, yields the element internal force vector. The internal force vector plays the key role in the solution of the nonlinear finite element equations. Also figuring prominently in the nonlinear solution process are the tangent stiffness matrices.

4.6 The Tangent Stiffness Matrix

In the case of the nonlinear plane frame element formulation given here, only the linear portion and the initial stress portion of the incremental stiffness matrix are included in the tangent stiffness matrix. The linear portion of the tangent stiffness matrix, often referred to as K_0, is the stiffness matrix which is commonly used in linear analysis. The familiar form of the matrix can be seen below:

$$K_0 = \alpha \begin{bmatrix} \beta & 0 & 0 & -\beta & 0 & 0 \\ 0 & 12 & 6L & 0 & -12 & 6L \\ 0 & 6L & 4L^2 & 0 & -6L & 2L^2 \\ -\beta & 0 & 0 & \beta & 0 & 0 \\ 0 & -12 & -6L & 0 & 12 & -6L \\ 0 & 6L & 2L^2 & 0 & -6L & 4L^2 \end{bmatrix} \qquad (4\text{-}25)$$

where

$$\alpha = \frac{E\,I}{L^3} \quad , \quad \beta = \frac{A\,L^2}{I}$$

with E being the modulus of elasticity, A being the cross-sectional area, and I being the moment of inertia of the section. Similar to K_0, the initial stress matrix K_σ can be given as

$$K_\sigma = \frac{P}{L}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{6}{5} & \frac{L}{10} & 0 & -\frac{6}{5} & \frac{L}{10} \\ 0 & \frac{L}{10} & \frac{2L^2}{15} & 0 & -\frac{L}{10} & -\frac{L^2}{30} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{6}{5} & -\frac{L}{10} & 0 & \frac{6}{5} & -\frac{L}{10} \\ 0 & \frac{L}{10} & -\frac{L^2}{30} & 0 & -\frac{L}{10} & \frac{2L^2}{15} \end{bmatrix} \qquad (4\text{-}26)$$

where P is the axial force in the frame element given by

$$P = \sigma A \tag{4-27}$$

Chapter 5
The Computer Programs

5.1 Introduction

The programs described in this chapter are the ones used for the implementation and comparison of the two nonlinear plane truss formulations presented in previous chapters. Both programs are written in the Fortran-77 programming language using double precision.

5.2 Program Development

The two programs given here are modifications to the programs given by Sunku (1991) and Ahmed (1988). The tree charts of the program structure, given in Fig. 5-1 and 5-2, denote the modified subroutines by cross-hatching. Similarly N-S diagrams and verbal descriptions are also given for the affected subroutines.

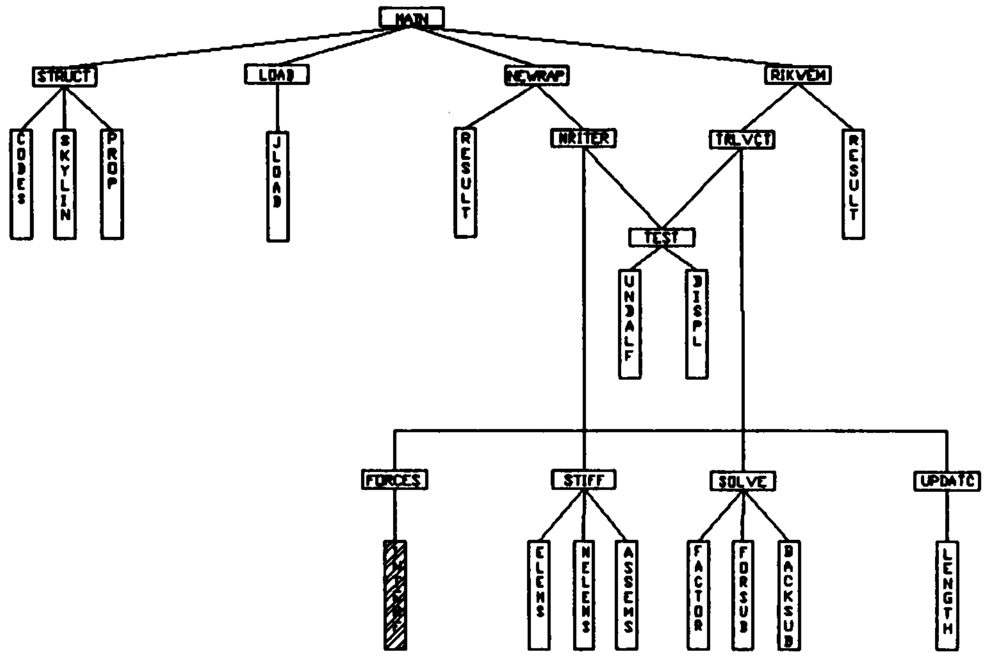


Figure (5-1) Tree Chart for Program Using the B-notation

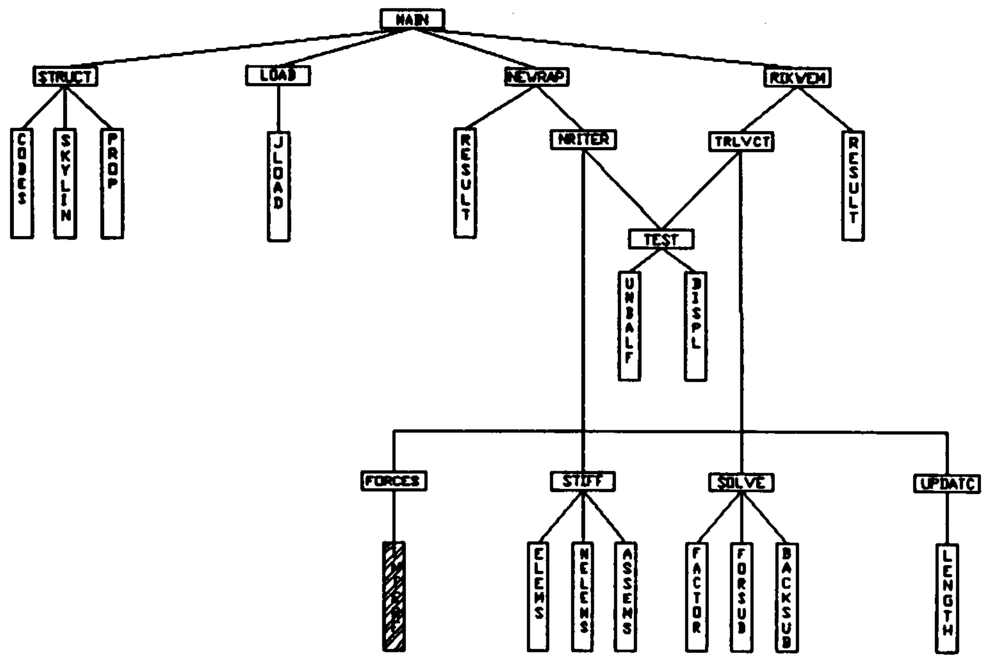


Figure (5-2) Tree Chart for Program Using Bathe's Notation

5.3 Subroutines

The subroutine INTERF computes the internal forces for the unconstrained global degrees of freedom of each element in the new configuration. This routine was modified in order that the comparison of the two truss formulations could be made. The NS-diagrams of Figures 5-3 and 5-4 give an overview of the differences in implementation while Appendices A and B present the actual Fortran code.

```
DO FOR III = 1,4
    ZZZ(III) = 0.D0
    AAA(III) = 0.D0
DO FOR II = 1,4
    J = MCODE(I,II)
    IF (J .NE. 0)
        THEN
            ZZZ(II) = D(J)
        ELSE
ZZZ1 = ZZZ(1)
ZZZ2 = ZZZ(2)
ZZZ3 = ZZZ(3)
ZZZ4 = ZZZ(4)
AAA(1) = C1S(I)*ZZZ1+C1S(I)*ZZZ2
AAA(2) = -C2S(I)*ZZZ1+C1S(I)*ZZZ2
AAA(3) = C1S(I)*ZZZ3+C2S(I)*ZZZ4
AAA(4) = -C2S(I)*ZZZ3+C1S(I)*ZZZ4
AAA1 = AAA(1)
AAA2 = AAA(2)
AAA3 = AAA(3)
AAA4 = AAA(4)
FL1 = -5.0D-1*((2.0D0*AAA3-2.0D0*AAA1)*AREA(I)*ELENG
      (I)**2+(AAA4**2-2.0D0*AAA2*AAA4+3.0D0*AAA3**
      2-6.0D0*AAA1*AAA3+AAA2**2+3.0D0*AAA1**2)*
      AREA(I)*ELENG(I)+((AAA3-1.0D0*AAA1)*AAA4**2+
      (2.0D0*AAA1*AAA2-2.0D0*AAA2*AAA3)*AAA4+AAA3
      **3-3.0D0*AAA1*AAA3**2+(AAA2**2+3.0D0*AAA1
      **2)*AAA3-1.0D0*AAA1*AAA2**2-1.0D0*AAA1**3)
      *AREA(I)*EMOD(I)/ELENG(I)**3
FL2 = (-5.0D-1*(((2.0D0*AAA3-2.0D0*AAA1)*AAA4-2.0D0
      *AAA2*AAA3+2.0D0*AAA1*AAA2)*AREA(I)*ELENG(I)
      +(AAA4**3-3.0D0*AAA2*AAA4**2+(AAA3**2-2.0D0
      *AAA1*AAA3+3.0D0*AAA2**2+AAA1**2)*AAA4-1.0D0
      *AAA2*AAA3**2+2.0D0*AAA1*AAA2*AAA3-1.0D0*
      AAA2**3-1.0D0*AAA1**2*AAA2)*AREA(I))*EMOD(I)
      /ELENG(I)**3)*-1.0D0
FL3 = 5.0D-1*((2.0D0*AAA3-2.0D0*AAA1)*AREA(I)*ELENG
      (I)**2+(AAA4**2-2.0D0*AAA2*AAA4+3.0D0*AAA3
      **2-6.0D0*AAA1*AAA3+AAA2**2+3.0D0*AAA1**2)
      *AREA(I)*ELENG(I)+((AAA3-1.0D0*AAA1)*AAA4**2
      +(2.0D0*AAA1*AAA2-2.0D0*AAA2*AAA3)*AAA4+
      AAA3**3-3.0D0*AAA1*AAA3**2+(AAA2**2+3.0D0
      *AAA1**2)*AAA3-1.0D0*AAA1*AAA2**2-1.0D0*AAA1
      **3)*AREA(I))*EMOD(I)/ELENG(I)**3
FL4 = (5.0D-1*(((2.0D0*AAA3-2.0D0*AAA1)*AAA4-2.0D0*
      AAA2*AAA3+2.0D0*AAA1*AAA2)*AREA(I)*ELENG(I)
      +(AAA4**3-3.0D0*AAA2*AAA4**2+(AAA3**2-2.0D0
      *AAA1*AAA3+3.0D0*AAA2**2+AAA1**2)*AAA4-1.0D0
      *AAA2*AAA3**2+2.0D0*AAA1*AAA2*AAA3-1.0D0*
      AAA2**3-1.0D0*AAA2**3-1.0D0*AAA1**2*AAA2)*
      AREA(I)*EMOD(I)/ELENG(I)**3)*-1.0D0
DO FOR L = 1,4
    K = MCODE(L,I)
    IF (K .NE. 0)
        THEN
            IF L =
                1   F(K) = C1S(I)*FL1-C2S(I)*FL2+F(K)
                2   F(K) = C2S(I)*FL1+C1S(I)*FL2+F(K)
                3   F(K) = C1S(I)*FL3-C2S(I)*FL4+F(K)
                4   F(K) = C2S(I)*FL3+C1S(I)*FL4+F(K)
        ELSE
```

Figure (5-3) NS-Diagram of INTERF Subroutine Using the B-notation

```
GAM=DASIN (C2S(I)*C1(I)-C1S(I)*C2(I))
DO FOR III = 1,4
    ZZZ(III) = 0.D0
    AAA(III) = 0.D0
DO FOR II = 1,4
    J = MCODE(II,I)
    IF (J .NE. 0)
        THEN
            ZZZ(II) = D(J)
            DDD(II) = DD(J)
        ELSE
DD1 = C1S(I)*DDD(1)+C2S(I)*DDD(2)
DD3 = C1S(I)*DDD(3)+C2S(I)*DDD(4)
DD4 = -C2S(I)*DDD(3)+C1S(I)*DDD(4)
DD2 = -C2S(I)*DDD(1)+C1S(I)*DDD(2)
STRES(I) = ((-1.D0*DCOS(GAM)*DD1-DSIN(GAM)*DD2+DCOS
            (GAM)*DD3+DSIN(GAM)*DD4)/ELENG(I)+0.5D0*
            (((DD3-DD1)/ELENG(I))**2+((DD4-DD2)/
            ELENG(I))**2))*EMOD(I)
STRESS(I) = STRESS(I)+STRES(I)
FL1 = -AREA(I)*STRESS(I)*DCOS(GAM)
FL2 = -AREA(I)*STRESS(I)*DSIN(GAM)
FL3 =  AREA(I)*STRESS(I)*DCOS(GAM)
FL4 =  AREA(I)*STRESS(I)*DSIN(GAM)
DO FOR L = 1,4
    K = MCODE(L,I)
    IF (K .NE. 0)
        THEN
            IF L =
                1   F(K) = C1S(I)*FL1-C2S(I)*FL2+F(K)
                2   F(K) = C2S(I)*FL1+C1S(I)*FL2+F(K)
                3   F(K) = C1S(I)*FL3-C2S(I)*FL4+F(K)
                4   F(K) = C2S(I)*FL3+C1S(I)*FL4+F(K)
        ELSE
```

Figure (5-4) . NS-Diagram of INTERF Subroutine using Bathe's Notati

Chapter 6
Test Problems and Results

6.1 Introduction

Five test cases were evaluated to compare the numerical solutions of the two nonlinear plane truss formulations presented earlier. As a standard for comparison, the test problems were also analyzed with ABAQUS. The Newton/Raphson method was used by ABAQUS and the two programs as the nonlinear solution algorithm. The same load steps and force tolerance were also used for comparing each test case. For the first three test cases exact solutions were obtained and plotted with the results from ABAQUS and the two programs. The results of the exact solutions are plotted in two curves. One giving the solution with the element cross-sectional area assumed constant,

$$P = EA_0 \ln\left(\frac{\sqrt{(w-v)^2 + L^2}}{\sqrt{w^2 + L^2}}\right) \frac{(w - v)}{\sqrt{(w-v)^2 + L^2}} \tag{6-1}$$

and the other giving the solution when the element volume is held constant.

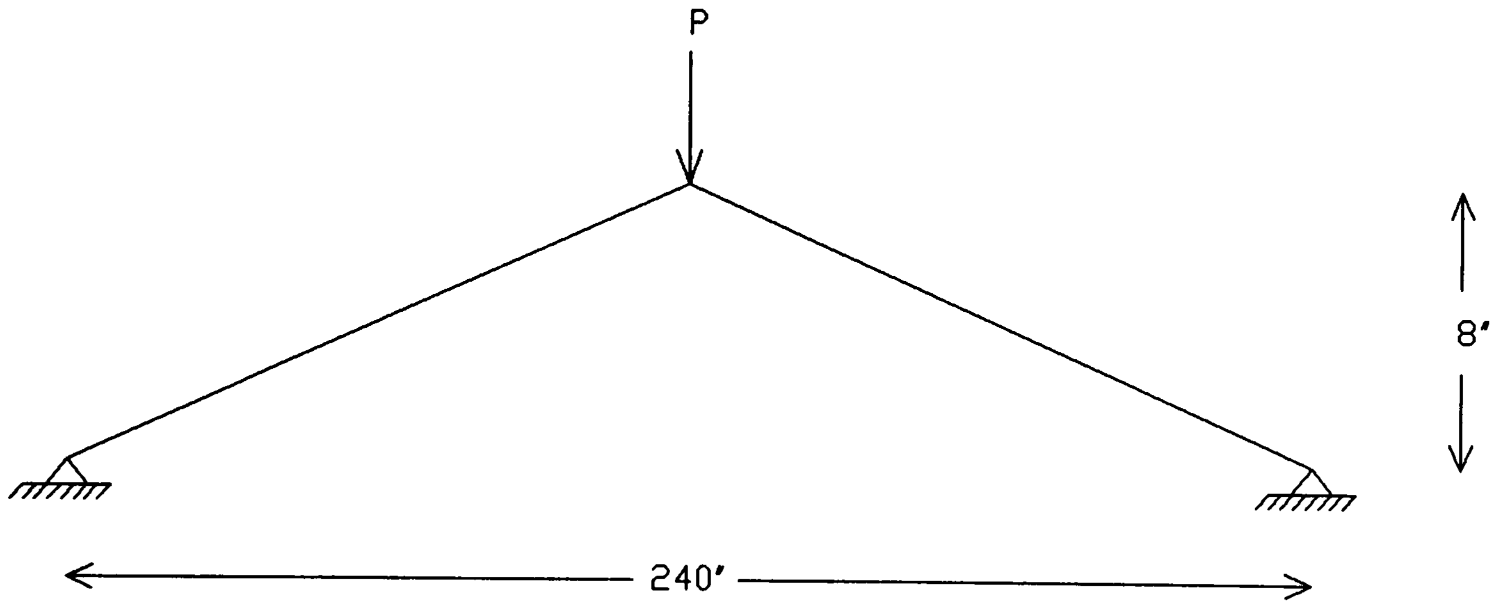
$$P = \frac{EA_0\,(w-v)\left(\sqrt{w^2+L^2} - \sqrt{(w-v)^2 + L^2}\right)}{(w-v)^2 + L^2} \tag{6-2}$$

where P is the load applied to the central node, E is 29500 ksi, A_0 is the original cross-sectional area, w is the height of the truss arch, v is the vertical downward deflection of the central node, and L is the span of the truss arch.

6.2 Test Case 1

The first test structure evaluated is the truss arch given in Fig. 6-1. The arch consists of two elements and three nodes. The base nodes are pinned. A single load, designated as P, of 16 kips is incrementally applied to the central node of the structure. The critical load for this structure was determined by ABAQUS to be 16.78 kips. The height of the structure, as seen in Fig 6-1, is 8 inches and the span is 240 inches. Each member has a cross-sectional area of 5 square inches. The material used for each member is steel, thus a modulus of elasticity of 29,500 ksi was used.

The test case was run on ABAQUS and both programs. Similarly, two exact solutions are also given in this plot. One of these solutions assumes the cross-sectional area to remain unchanged during the deformation while the other solution assumes constant volume of the element but allows for the cross-sectional area to change. The results of these runs are plotted in Fig. 6-2. Increasing load increments are plotted against vertical downward displacement of the central node in this figure. The results are that no significant difference can be seen in the solution accuracy of the three runs of this test case.



Cross-Sectional Area of Each Element = 5 in.^2
Modulus of Elasticity = 29500 ksi

Figure (6-1) Test Case 1 - Truss Arch Test Problem

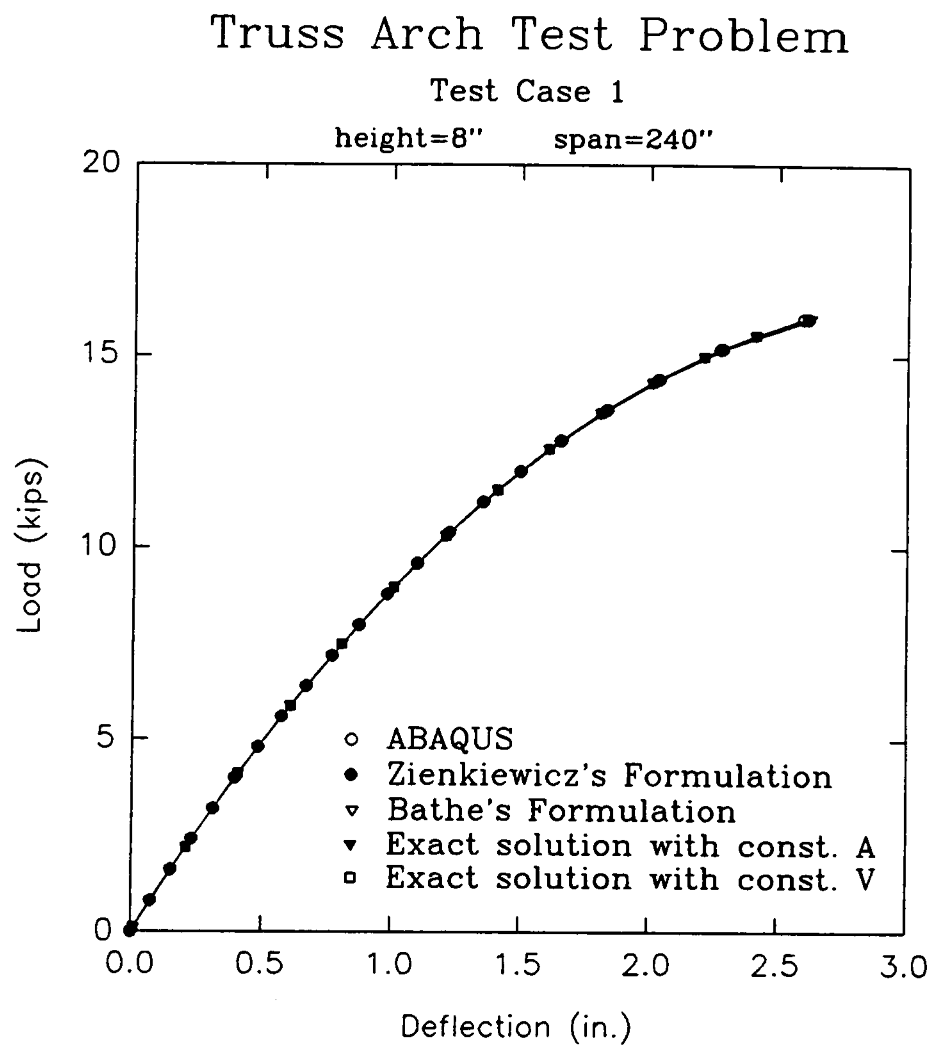
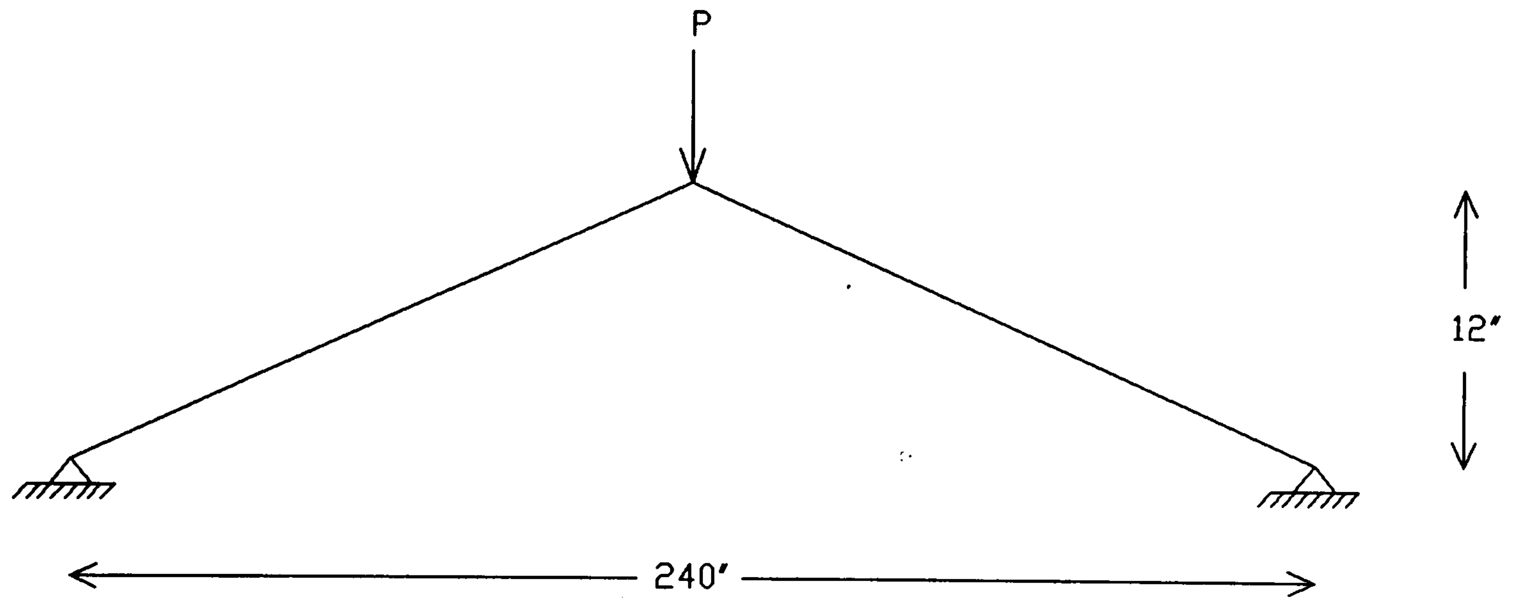


Figure (6–2)   Test Case 1 Equilibrium Paths

6.3 Test Case 2

The same structure described in Test Case 1 is modified for this case. The height was increased to 12 inches but the span remained unchanged (see Fig. 6-3). The load P, incrementally imposed on the central node, is increased so as to ultimately attain 55 kips. The critical load of this structure was determined by ABAQUS to be 55.56 kips.

The test case was run on ABAQUS and both programs. The results of these runs and the results of the exact solutions are plotted in Fig. 6-4. Increasing load increments are plotted against vertical downward displacement of the central node in this figure. The results are that no significant difference can be seen in the solution accuracy of the three runs of this test case.



Cross-Sectional Area of Each Element = 5 in.^{2}
Modulus of Elasticity = 29500 ksi

Figure (6-3) Test Case 2 - Truss Arch Test Problem

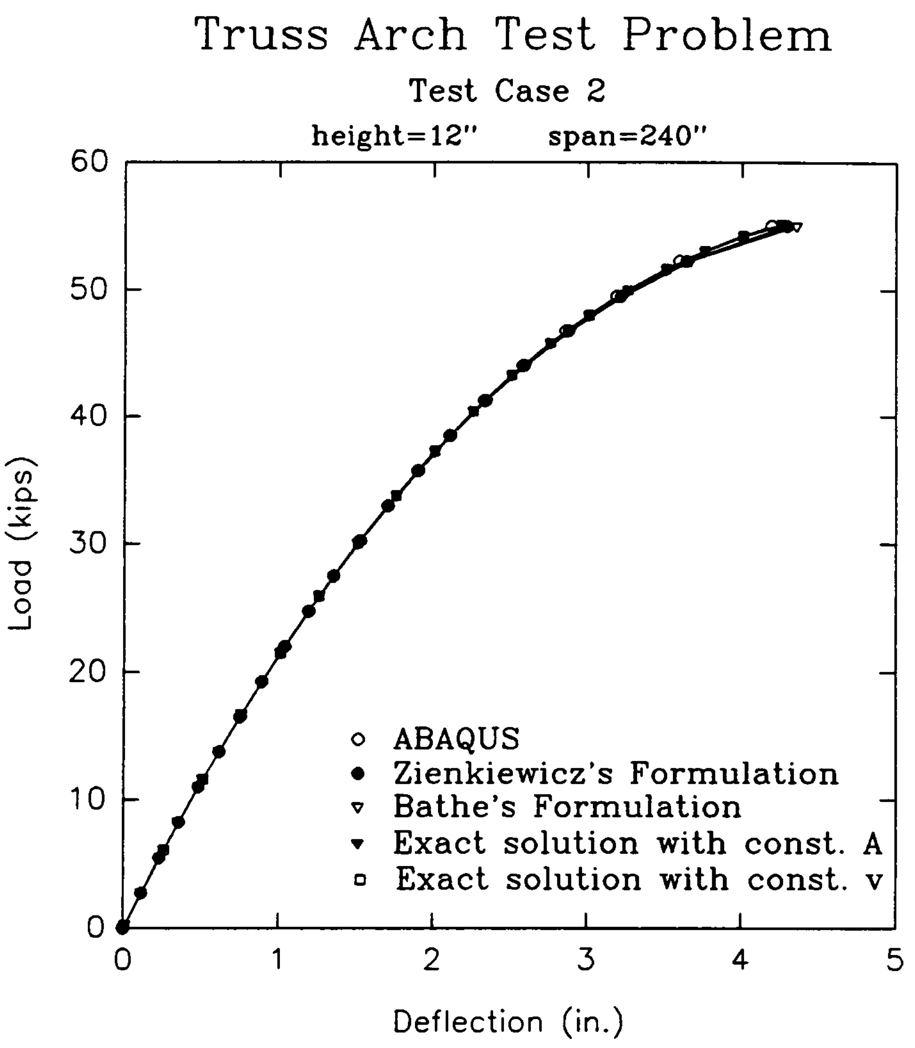
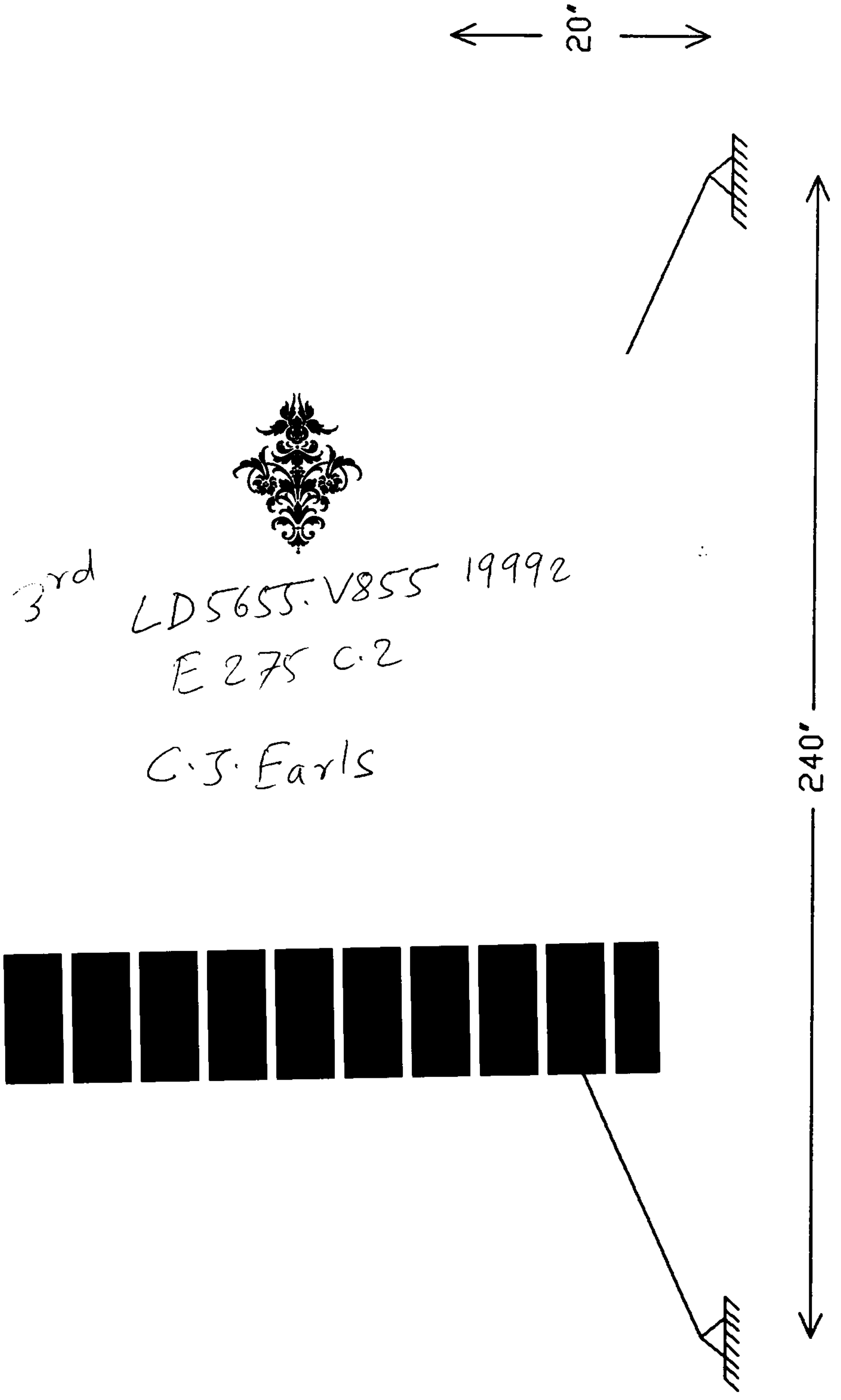


Figure (6-4) Test Case 2 Equilibrium Paths

6.4 Test Case 3

Once again the structure in Test Case 1 is modified. All parameters remained unchanged from Test Case 1 except the height. This is increased to 20 inches (see Fig. 6-5). The incremental load P, applied to the central node, was changed so as to attain a final 245 kip value. The critical load of this structure was determined by ABAQUS to be 247 kips.

The test case was run on ABAQUS and both programs. The results of these runs and the results of the exact solutions are plotted in Fig. 6-6. Increasing load increments are plotted against vertical downward displacement of the central node in this figure. The results are that minute differences between the results of ABAQUS and the two programs can be seen. This discrepancy between ABAQUS and the two programs seems to manifest itself more in the upper region of the equilibrium paths near the limit point. In a structure this steep a large amount of strain will be present in the equilibrium configurations of the structure corresponding to this region in the plots. This point is significant since ABAQUS uses the log strain while both programs use the Green-Lagrange strain. The two strain measures agree in cases of small strain, but tend to disagree as the strains become large this point was previously addressed in chapter 2. The results from the two programs compare well with each other.



Cross-Sectional Area of Each Element = 5 in.^{2}
Modulus of Elasticity = 29500 ksi

Figure (6-5) Test Case 3 - Truss Arch Test Problem

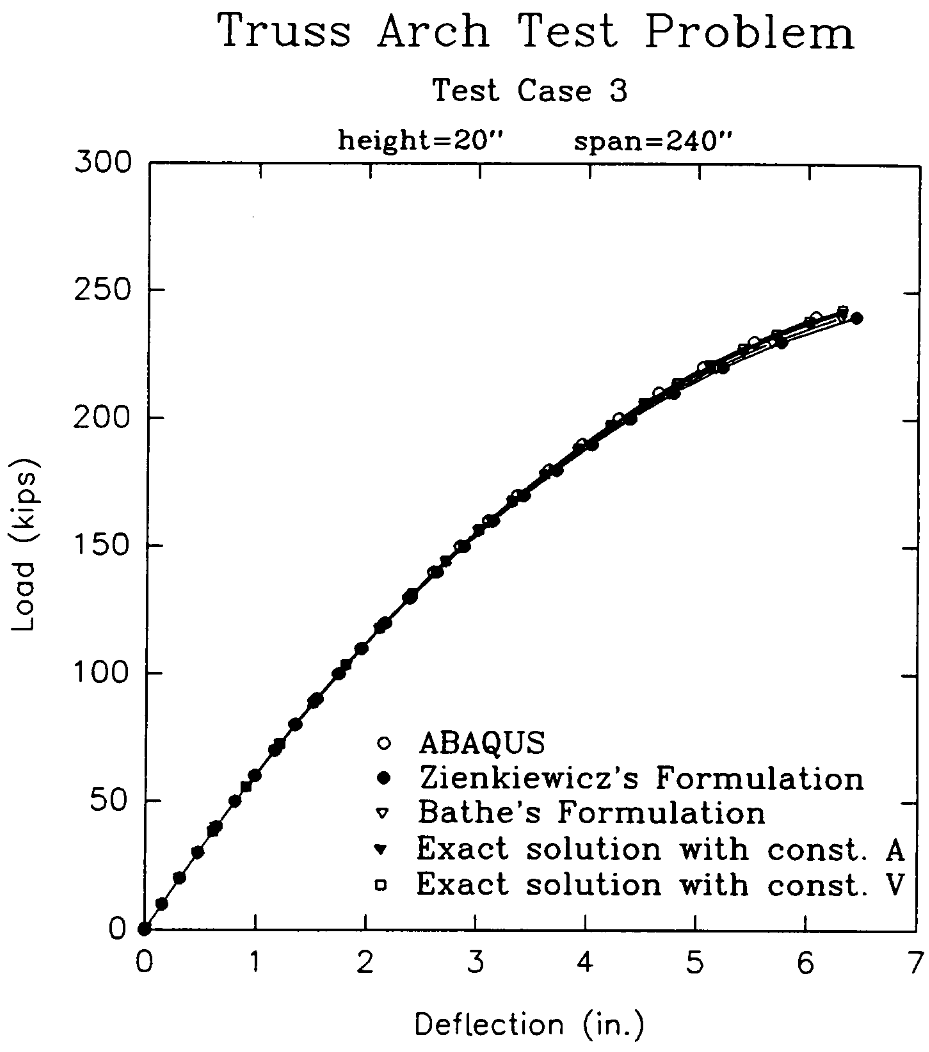


Figure (6–6) Test Case 3 Equilibrium Paths

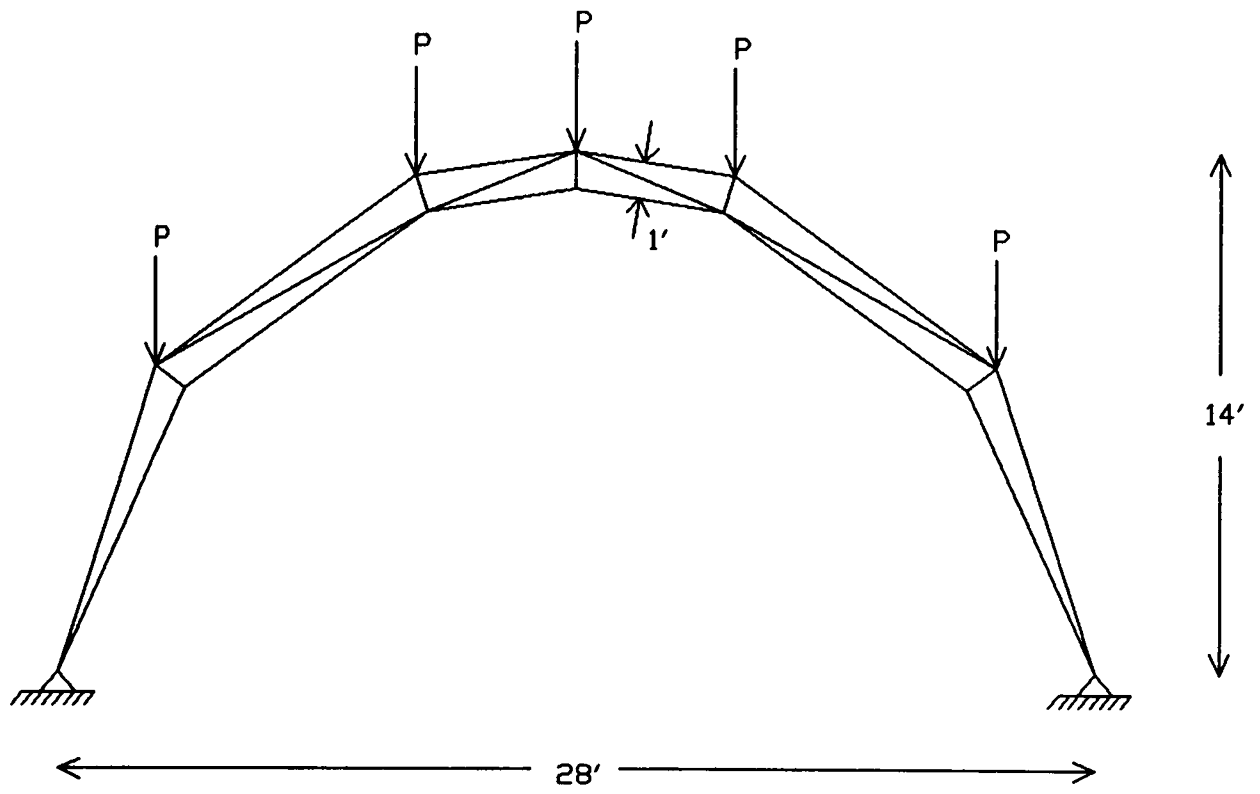
6.5 Test Case 4

The fourth test case evaluated is a plane circular lattice arch. It consists of 20 elements and 12 nodes. The two base nodes are pinned. Identical vertical downward loads, P, of 200 kips are incrementally imposed at each unconstrained top chord node. A critical load of 204 kips was given by ABAQUS for this load distribution. The height of the structure, as given in Fig. 6-7, is 14 feet and it has a span of 28 feet. Each element has a cross-sectional area of 5 square inches. Each element is constructed out of steel, thus a modulus of elasticity of 29,500 ksi is used.

The results of the evaluation of this test problem with ABAQUS and the two programs are given in Fig. 6-8. This figure shows a plot of increasing load increments versus vertical downward displacement of the central node. No significant differences between the three curves can be discerned.

6.6 Test Case 5

The fifth test case evaluated is a shallow plane circular lattice arch. It consists of 37 elements and 20 nodes. Identical vertical downward loads, P , of 230 kips are imposed at each unconstrained top chord node. A critical load of 232 kips was given by ABAQUS for this load distribution. The height of the structure, as displayed in Fig. 6-9, is 4 feet and it has a span of 28 feet. Each



Cross-Sectional Area of Each Element = 5 in.^{2}
Modulus of Elasticity = 29500 ksi

Figure (6-7) Test Case 4 - Circular Lattice Arch Test Problem

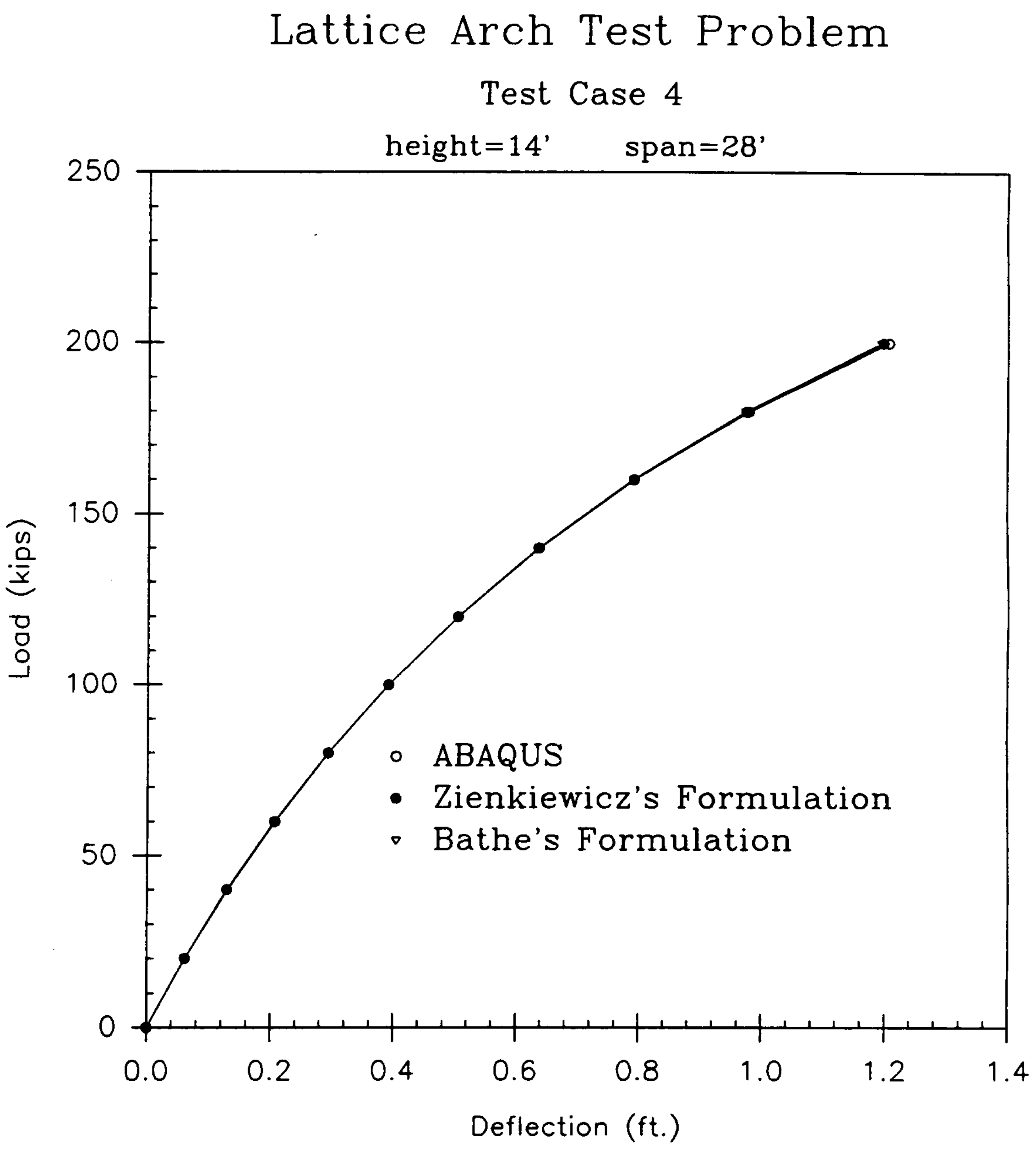
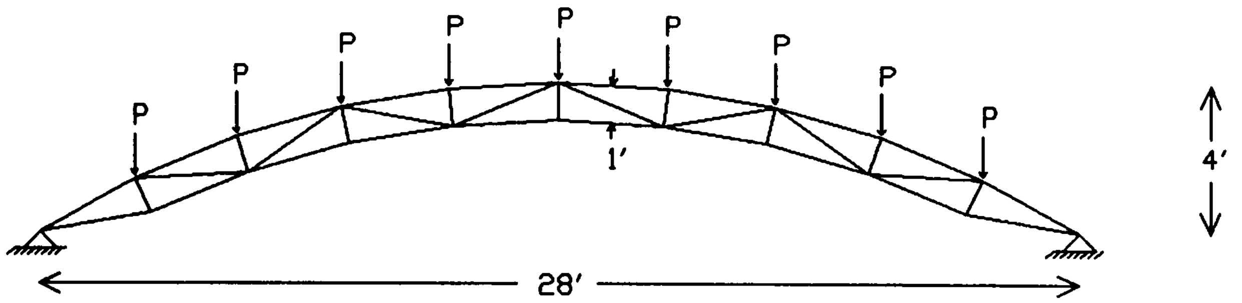


Figure (6–8) Test Case 4 Equilibrium Paths



Cross-Sectional Area of Each Element = 5 in.
Modulus of Elasticity = 29500 ksi

Figure (6-9) Test Case 5 - Circular Lattice Arch Test Problem

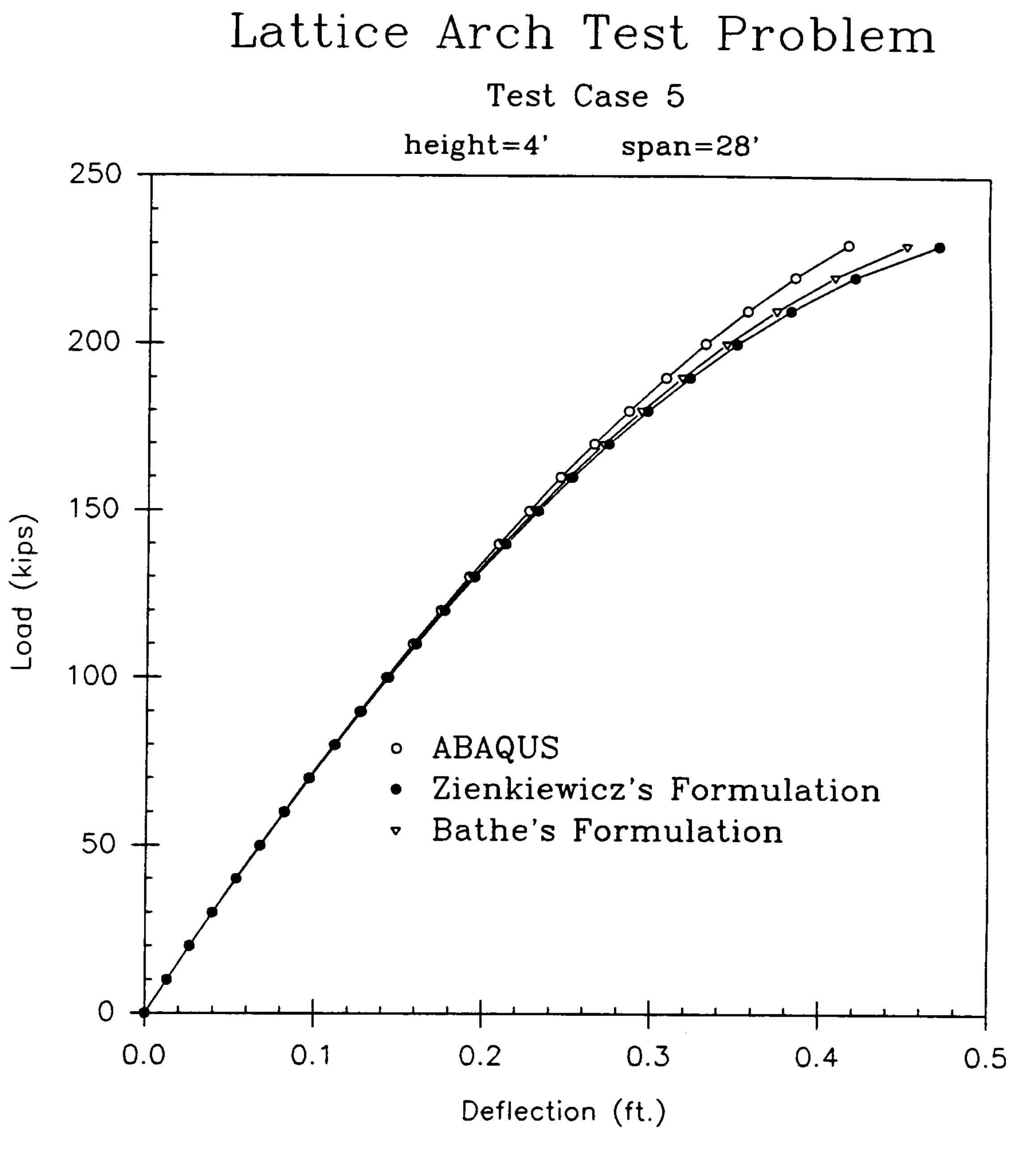


Figure (6–10) Test Case 5 Equilibrium Paths

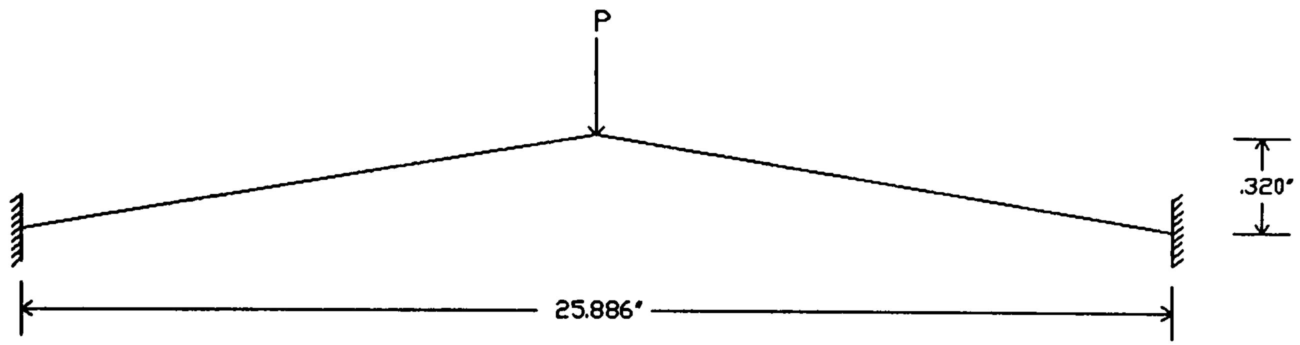
structural member has a cross-sectional area of 5 square inches. The truss is constructed out of steel, thus a modulus of elasticity of 29,500 ksi is used.

The results of the analysis of this test problem with ABAQUS and the two programs are given in Fig. 6-10. This figure displays a plot of increasing load increments versus vertical downward displacement of the central node. A discrepancy between the results of ABAQUS and the two programs is noted in the region of the equilibrium paths near the limit point.

6.7 Test Case 6

The following two test cases are not given to compare Zienkiewicz's formulation to that of Bathe's. The purpose of these test cases is to compare the results of a geometrically nonlinear finite element program using the formulation of the nonlinear frame element given in chapter 4 with the B23 element of ABAQUS. Test case 6 is a toggle frame of span 25.886" and height .320" (as shown in Fig. 6-11). A concentrated load, P, of 80 pounds is applied incrementally to the central node. The end nodes of the structure are clamped. The dimensions of the elements are .753" by .243" thus the cross-sectional area is .182979 in.^{2} and the moment of inertia is .000900394 in.^{4} The modulus of elasticity is 10.3x10^6 p.s.i.

The results of the evaluation of this test problem with ABAQUS and the program is given in Fig. 6-12. This figure shows a plot, based on the convergent



Cross-Sectional Area of Each Element = .182979 in.^{2}
Modulus of Elasticity = 10.3×10^6 p.s.i.

Figure (6-11) Test Case 6 - Toggle Frame Test Problem

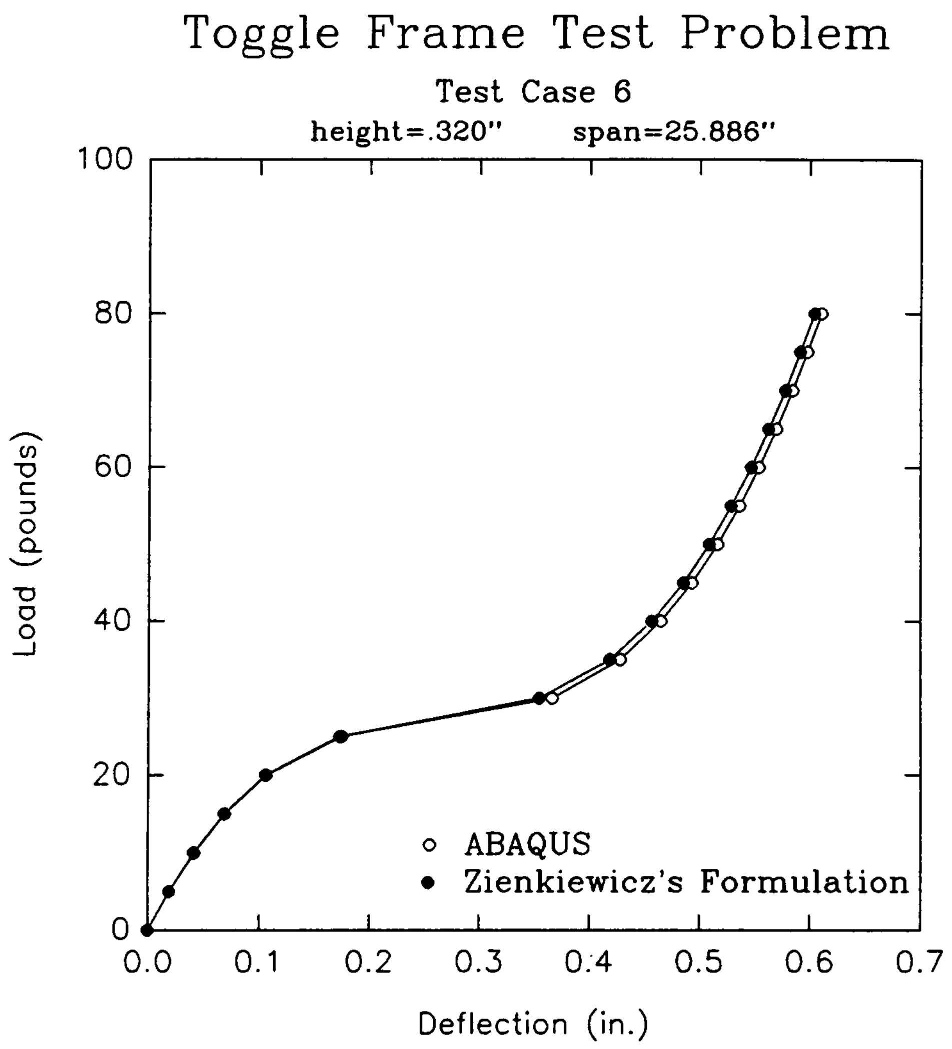
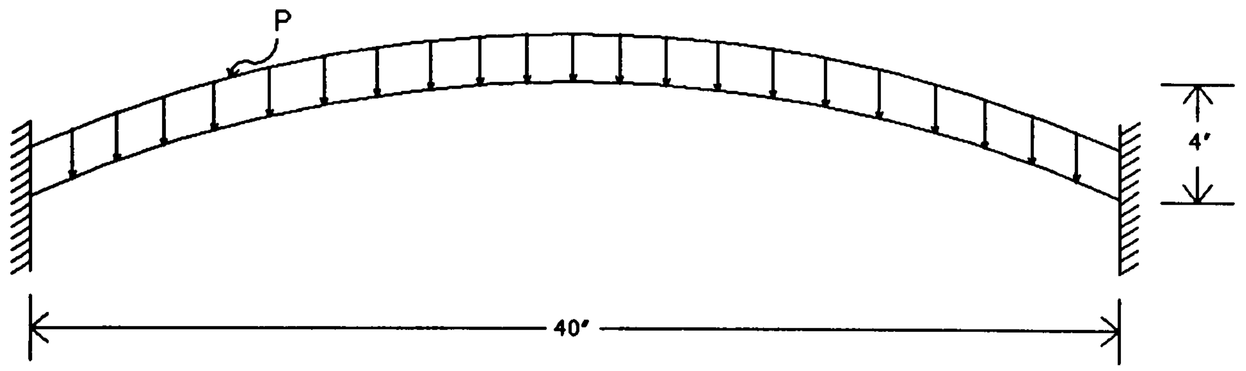


Figure (6−12) Test Case 6 Equilibrium Paths solutions of ABAQUS and the program, of increasing load increments versus vertical downward displacement of the central node. The results compare well.

6.8 Test Case 7

Test case 7 is a continuous arch of span 40" and height 4" (as defined in Fig. 6-13). A distributed load, P, of 700 pounds per inch is incrementally applied to the structure. The end nodes are clamped. The dimensions of the element cross sections are 1 by 1. The modulus of elasticity used is $1\text{x}10^7$ p.s.i.

The results of the evaluation of this test problem with ABAQUS and the program are given in Fig. 6-14. This figure shows a plot of increasing load increments versus vertical downward displacement of the central node. The results show a good comparison between the formulation of chapter 4 and ABAQUS.



Cross-Sectional Area of Each Element = 1 in.^{2}
Modulus of Elasticity = 1 x 10^7 p.s.i.

Figure (6-13) Test Case 7 - Continuous Arch Test Problem

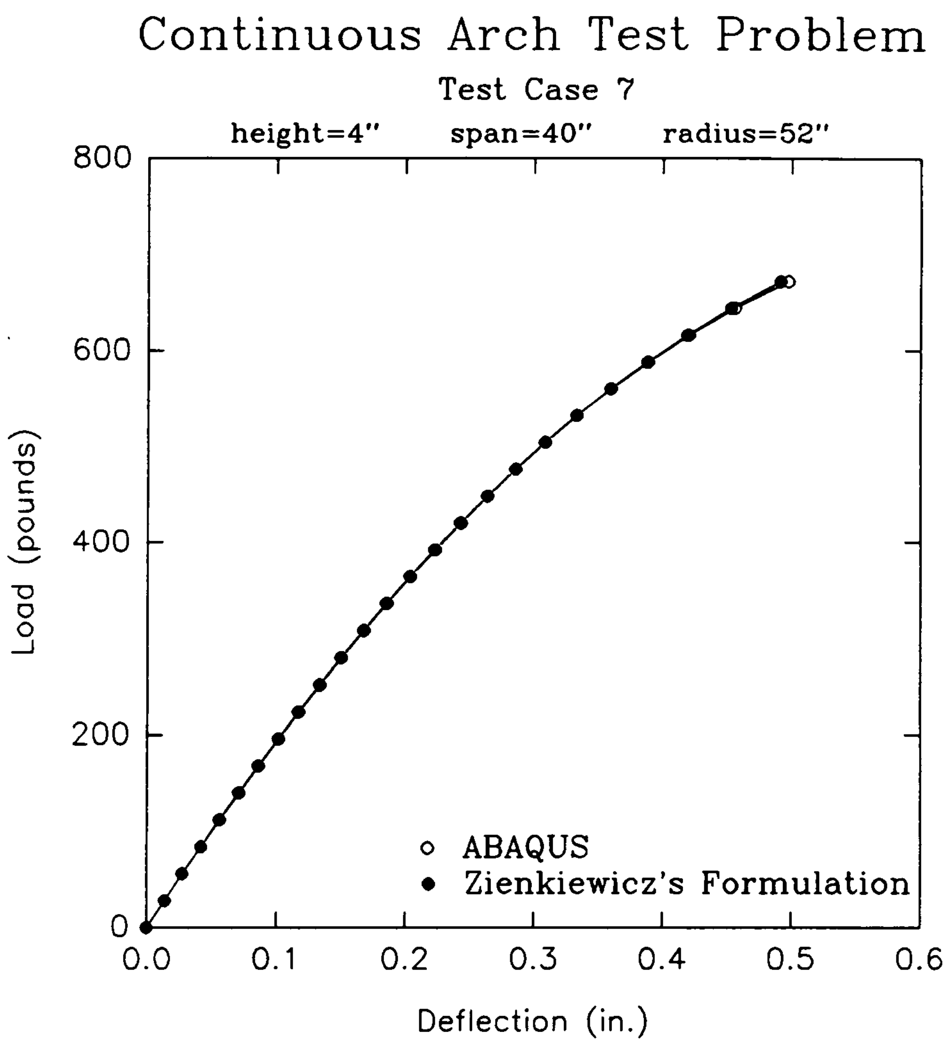


Figure (6–14) Test Case 7 Equilibrium Paths

Chapter 7
Conclusions and Recommendations

7.1 Conclusions

In this thesis the continuum mechanics based incremental nonlinear finite element formulations of Zienkiewicz and Bathe were presented and applied to the nonlinear plane truss formulation. The two subsequent formulations of this element were implemented into two geometrically nonlinear finite element programs.

Five test problems were then analyzed with the two programs and ABAQUS. The outcome of these test cases was that no significant difference in solution could be discerned between ABAQUS and the two programs except in instances of very large strain. In such cases ABAQUS provided a stiffer response due to its use of log strain as opposed to the Green-Lagrange strain which was used in the programs.

Upon examining the formulations it was seen that Bathe's formulation contains an approximation in the internal force vector whereas Zienkiewicz's formulation does not. Bathe's approximation is acceptable as long as the displacement increments remain small. This conclusion is supported by the results obtained from the analysis of the test cases presented in chapter 6

7.2 Recommendations

For further study on this topic the following recommendations are submitted.

1. Use the Riks/Wempner algorithm to trace the equilibrium paths of the test structures beyond the first limit point to check solution agreement between the two formulations and ABAQUS.

2. Use the B-notation and the notation of Bathe to formulate the Bernoulli-Euler 2-D beam element. Implement these two formulations into programs and compare analysis results against similar tests done on ABAQUS using the B23 element.

References

1. Ahmed, Z., (1988) "Nonlinear Finite Element Creep Analysis of Plane Trusses," Independent Study Report, Virginia Polytechnic Institute and State University, Blacksburg, Virginia.

2. Bathe, K. J., (1982) *Finite Element Procedures in Engineering Analysis*, Prentice-Hall Inc., Englewood-Cliffs, New Jersey.

3. Bathe, K. J., Ramm, E., and Wilson E., (1975) "Finite Element Formulations For Large Deformation Dynamic Analysis," *International Journal For Numerical Methods In Engineering*, Vol. 9, 353-386.

4. Crisfield, M. A., (1991) *Non-linear Finite Element Analysis of Solids and Structures*, John Wiley & Sons, Chichester, West Sussex. ✓

5. Holzer, S. M., (1990) "Class Notes on Finite Element Analysis of Structures," Virginia Polytechnic Institute and State University, Blacksburg, Virginia, Spring Semester.

6. Holzer, S. M., (1985) *Computer Analysis of Structures*, Elsevier Science Publishing Co., Inc., New York, New York.

7. Martin, H. C., and Carey, G. F., (1973) *Introduction to Finite Element Analysis*, McGraw-Hill Book Company, New York, New York.

8. Oden, J. T., (1972) *Finite Elements of Nonlinear Continua*, McGraw-Hill Book Company, New York, New York.

9. Osterrieder, P., and Ramm, E., (1987) "Nonlinear Frame Analysis By Finite Element Method (response to)," *Journal of the Structural Division - A.S.C.E.,* Vol. 114, 497-498. ✓

10. Rajasekaran, S., and Murray, D. W., (1973) "Incremental Finite Element Matrices," *Journal of the Structural Division - A.S.C.E.,* Vol. 99, 2423 - 2437.

11. Stippes, M., Wempner, G., Stern, M., and Beckett, R., (1961) *The Mechanics of Deformable Bodies*, Charles E. Merrill Publishing Co., Columbus, Ohio.

12. Sunku, B. S., (1991) "Comparison of Modified Riks/Wempner and Homotopy Methods," M. S. Thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, September.

13. Wood, R. D., and Schrefler, B., (1978) "Geometrically Non-Linear Analysis--A Correlation of Finite Element Notations," *International Journal For Numerical Methods In Engineering*, Vol. 12, 635-642.

14. Wunderlich, W., and Obrecht, H., (1981) "Large Spatial Deformations of Rods Using Generalized Variational Principles," *Nonlinear Finite Element Analysis in Structural Mechanics*, Springer-Verlag, Berlin.

15. Zienkiewicz, O. C., and Nayak, G. C., (1973) "A General Approach to Problems of Plasticity and Large Deformation Using Isoparametric Elements," *Proceedings of the Third Conference on Matrix Methods in Structural Mechanics*, Air Force Flight Dynamics Laboratory Air Force Systems Command and Air Force Institute of Technology Air University.

16. Zienkiewicz, O. C., and Wood, R. D. (1976) "Geometrically Nonlinear Finite Element Analysis of Beams, Frames, Arches and Axisymmetric Shells, *Computers & Structures*, Vol. 7, 725-735. ✓

17. Zienkiewicz, O. C., and Taylor, R. L., (1991) *The Finite Element Method*, Fourth Edition, Volume 2, McGraw-Hill Book Company, London.

Bibliography

1. Argyris, J. H., and Symeonidis, Sp., (1981) "Nonlinear Finite Element Analysis of Elastic Systems Under Nonconservative Loading-Natural Formulation. Part I. Quasistatic Problems," *Computer Methods in Applied Mechanics and Engineering,* Vol. 26, 75-123.

2. Bathe, K. J., and Almeida, C. A., (1980) "A Simple and Effective Pipe Elbow Element-Linear Analysis," *Journal of Applied Mechanics - Transactions of the A.S.M.E.,* Vol. 47, 93-100.

3. Bathe, K. J., and Bolourchi, S., (1979) "Large Displacement Analysis of Three-Dimensional Beam Structures," *International Journal for Numerical Methods in Engineering,* Vol. 14, 961-986.

4. Bathe, K. J., and Cimento, A. P., (1980) "Some Practical Procedures for the Solution of Nonlinear Finite Element Equations," *Computer Methods in Applied Mechanics and Engineering,* Vol. 22, 59-85.

5. Bathe, K. J., and Ramaswamy S., (1979) "On Three-Dimensional Nonlinear Analysis of Concrete Structures," *Nuclear Engineering and Design,* Vol. 52, 385-409. √

6. Batra R. C., (1980) "Finite Plane Strain Deformations of Rubberlike Materials," *International Journal For Numerical Methods in Engineering,* Vol. 15, 145-160.

7. Belytschko, T., Lin, J. I., and Tsay, C., (1984) "Explicit Algorithms for the Nonlinear Dynamics of Shells," *Computer Methods in Applied Mechanics and Engineering,* Vol. 42, 225-251.

8. Belytschko, T., (1977) "Methods and Programs for Analysis of Fluid-Structure Systems," *Nuclear Engineering and Design,* Vol. 42, 41-52.

9. Bufler, H., (1983) "On the Work Theorems for Finite and Incremental Elastic Deformations with Discontinuous Fields: A Unified Treatment of Different Versions," *Computer Methods in Applied Mechanics and Engineering,* Vol. 36, 95-124.

10. Cescotto, S., and Fonder, G., (1979) "A Finite Element Approach for Large Strains of Nearly Incompressible Rubber-Like Materials," *International Journal of Solids and Structures,* Vol. 15, 589-605.

11. Chao, W. C., and Reddy, J. N., (1984) "Analysis of Laminated Composite Shells Using a Degenerated 3-D Element," *International Journal for Numerical Methods in Engineering,* Vol. 20, 1991-2007.

12. Cheng, J., and Kikuchi, N., (1986) "A Mesh Re-zoning Technique for Finite Element Simulations of Metal Forming Processes," *International Journal for Numerical Methods in Engineering,* Vol. 23, 219-228.

13. Cheng, J., and Kikuchi, N., (1985) "An Analysis of Metal Forming Processes Using Large Deformation Elastic-Plastic Formulations," *Computer Methods in Applied Mechanics and Engineering,* Vol. 49, 71-108.

14. Choi, K. K., and Santos, J. L. T., (1987) "Design Sensitivity Analysis of Non-Linear Structural Systems Part I: Theory," *International Journal for Numerical Methods in Engineering,* Vol. 24, 2039-2055.

15. Desai, C. S., Phan, H. V., and Perumpral, J. V., (1982) "Mechanics of Three-Dimensional Soil-Structure Interaction," *Journal of Engineering Mechanics - A.S.C.E.,* Vol. 108, 731-747.

16. Gadala, M. S., Dokainish, M. A., and Oravas, G. AE., (1984) "Formulation Methods of Geometric and Material Nonlinearity Problems," *International Journal for Numerical Methods in Engineering,* Vol. 20, 887-914.

17. Gross, J. L., and McGuire, W., (1983) "Progressive Collapse Resistant Design," *Journal of Structural Engineering - A.S.C.E.,* Vol. 109, 1-15.

18. Haber, R. B., (1984) "A Mixed Eulerian-Lagrangian Displacement Model For Large-Deformation Analysis in Solid Mechanics," *Computer Methods in Applied Mechanics and Engineering,* Vol. 43, 277-292.

19. Haber, R. B., and Abel, J. F., (1982) "Initial Equilibrium Solution Methods for Cable Reinforced Membranes Part I - Formulations," *Computer Methods in Applied Mechanics and Engineering,* Vol. 30, 263-284.

20. Haber, R. B., and Abel, J. F., (1983) "Contact-Slip Analysis Using Mixed Displacements," *Journal of Engineering Mechanics - A.S.C.E.,* Vol. 109, 411-429.

21. Hino, J., Yoshimura, T., and Ananthanarayana, N., (1985) "Vibration Analysis of Non-Linear Beams Subjected to a Moving Load Using the Finite Element Method," *Journal of Sound and Vibration,* Vol. 100, 477-491.

22. Johnson, G. R., (1977) "High Velocity Impact Calculations in Three Dimensions," *Journal of Applied Mechanics - Transactions of the A.S.M.E.,* Vol. 44, 95-100.

23. Jones, N., and Reis, H. L. M., (1980) "On the Dynamic Buckling of a Simple Elastic-Plastic Model," *International Journal of Solids and Structures,* Vol. 16, 969-989.

24. Kamat, M. P., and Hayduk R. J., (1982) "Recent Developments in Quasi-Newton Methods for Structural Analysis and Synthesis," *A.I.A.A. Journal,* Vol. 20, 672-679.

25. Kim, S. J., and Oden, J. T., (1985) "Finite Element Analysis of a Class of Problems in Finite Elastoplasticity Based on the Thermodynamical Theory of Materials of Type N," *Computer Methods in Applied Mechanics and Engineering,* Vol. 53, 277-302.

26. Kiousis, P. D., Voyiadjis, G. Z., and Tumay, M. T., (1986) "A Large Strain Theory for the Two Dimensional Problems in Geomechanics," *International Journal for Numerical and Analytical Methods in Geomechanics,* Vol. 10, 17-39.

27. Kroplin, B. H., (1981) "A Viscous Approach to Post Buckling Analysis," *Engineering Structures,* Vol. 3, 187-189.

28. Lee, G. C., Tseng, N. T., and Yuan, Y. M., (1983) "Finite Element Modeling of Lungs Including Interlobar Fissures and the Heart Cavity," *Journal of Biomechanics,* Vol. 16, 679-690.

29. Leonard, J. W., and Young, R. A., (1985) "Coupled Response of Compliant Offshore Platforms," *Engineering Structures,* Vol. 7, 74-84.

30. Milford, R. V., and Schnobrich, W. C., (1986) "Degenerated Isoparametric Finite Elements Using Explicit Integration," *International Journal for Numerical Methods in Engineering,* Vol. 23, 133-154.

31. Nimmer, R. P., and Miller, L. C., (1984) "Neck Propagation in Tensile Tests: A Study Using Rate-Independent, Strain Hardening Plasticity," *Journal of Applied Mechanics - Transactions of the A.S.M.E.,* Vol. 51, 759-765.

32. Nimmer, R. P., (1983) "Analysis of the Puncture of a Bisphenol-A Polycarbonate Disc," *Polymer Engineering and Science,* Vol. 23, 155-164.

33. Noor, A. K., and Peters, J. M., (1985) "Penalty Finite Element Models for Nonlinear Dynamic Analysis," *A.I.A.A. Journal,* Vol. 24, 312-320.

34. Noor, A. K., (1984) "Hybrid Analytical Technique for Nonlinear Analysis of Structures," *A.I.A.A. Journal,* Vol. 23, 938-946.

35. Noor, A. K., and Knight, N. F., (1980) "Nonlinear Dynamic Analysis of Curved Beams," *Computer Methods in Applied Mechanics and Engineering,* Vol. 23, 225-251.

36. Peterson, A., and Petersson, H., (1985) "On Finite Element Analysis of Geometrically Nonlinear Problems," *Computer Methods in Applied Mechanics and Engineering,* Vol. 51, 277-286.

37. Rebora, B., Zimmermann, Th., and Wolf, J. P., (1976) "Dynamic Rupture Analysis of Reinforced Concrete Shells," *Nuclear Engineering and Design,* Vol. 37, 269-297.

38. Sheinman, I., (1982) "Large Deflection of Curved Beam with Shear Deformation," *Journal of Engineering Mechanics - A.S.C.E.,* Vol. 108, 636-647.

39. Sheinman, I., (1980) "Dynamic Large-Displacement Analysis of Curved Beams involving Shear Deformation," *International Journal of Solids and Structures,* Vol. 16, 1037-1049.

40. Stein, E., and Wriggers, P., (1982) "Calculation of Impact-Contact Problems of Thin Elastic Shells Taking into Account Geometrical Nonlinearities within the Contact Region," *Computer Methods in Applied Mechanics and Engineering,* Vol. 34, 861-880.

41. Saigal, S., and Yang, T. Y., (1985) "Nonlinear Dynamic Analysis with a 48 D.O.F. Curved Thin Shell Element," *International Journal for Numerical Methods in Engineering,* Vol. 21, 1115-1128.

42. Tottenham, H., and Barony, S. Y., (1978) "Mixed Finite Element Formulation for Geometrically Non-Linear Analysis of Shells of Revolution," *International Journal for Numerical Methods in Engineering,* Vol. 12, 195-201.

43. Tseng, N. T., and Lee, G. C., (1985) "Inelastic Finite Strain Analysis of Structures Subjected to Nonproportional Loading," *International Journal for Numerical Methods in Engineering,* Vol. 21, 941-957.

44. Yang, Y. B., and McGuire, W., (1986) "Stiffness Matrix for Geometric Nonlinear Analysis," *Journal of Structural Engineering - A.S.C.E.,* Vol. 112, 853-877.

45. Yang, T. Y., and Saigal, S., (1984) "A Simple Element for Static and Dynamic Response of Beams with Material and Geometric Nonlinearities," *International Journal for Numerical Methods in Engineering,* Vol. 20, 851-867.

Appendix A

Geometrically Nonlinear Plane Truss Finite Element Fortran Program Using Bathe's Notation

```
C***********************************************************************
C           NONLINEAR ANALYSIS OF SPACE TRUSSES              *
C                                                  *
C     NEWTON-RAPHSON METHOD    AND    RIKS-WEMPNER METHOD          *
C***********************************************************************
C*                    INPUT DATA                            *
C***********************************************************************
C    LIST-DIRECTED INPUT:
C    INPUT UNITS: KIP, INCH, RADIAN, FAHRENHEIT
C
C    1. ENTER DATE IN THE FORM  01/14/91  (IN MAIN)
C         DATE
C
C    2. ENTER THE METHOD TO BE USED (IN MAIN)
C       NEWTON-RAPHSON: 1    RIKS-WEMPNER: 2
C         METHOD
C
C    3. ENTER NUMBER OF ELEMENTS AND NUMBER OF JOINTS (IN MAIN)
C         NE, NJ
C
C    4. ENTER MEMBER INCIDENCES (IN STRUCT)
C         MINC(1,I), MINC(2,I)  I=1,NE
C
C    5. ENTER FOR EACH JOINT CONSTRAINT (IN STRUCT)
C         JNUM, JDIR
C         AFTER LAST JOINT CONSTRAINT ENTER
C         0, 0
C
C    6. ENTER JOINT COORDINATES FOR EACH JOINT(IN PROP)
C         X(1,J), X(2,J), X(3,J)   J=1,NJ
C
C    7. ENTER MEMBER PROPERTIES CROSS SECTIONAL AREA
C         AND ELASTIC MODULUS FOR EACH MEMBER (IN PROP)
C         AREA(I), EMOD(I)    I=1,NE
C
C
C    8. IF THERE ARE JOINT LOADS ENTER (IN JLOAD)
C           JNUM, JDIR, FORCE
C           AFTER LAST JOINT LOAD ENTER
C           0, 0, 0
C         ELSE ENTER
C           0, 0, 0
C         END IF
C
C    9. ENTER THE MAXIMUM VALUE OF LAMBDA, INITIAL VALUE OF LAMBDA
C       AND THE INCREMENT IN LAMBDA   (IN MAIN)
C         QIMAX, QI, DQI
C
C   10. ENTER THE NUMBER OF ITERATIONS FOR UPDATING STIFFNESS MATRIX,
C       MAXIMUM AND DESIRED NUMBER OF ITERATIONS (IN MAIN)
C         ITENUM, ITEMAX, ITEDES
```

```
C
C   11. ENTER THE TOLERANCES IN DISPLACEMENT, AND FORCE (IN MAIN)       )
C          TOLDIS, TOLFOR
C
C   12. ENTER THE DOF FOR WHICH DISPLACEMENT AND LAMBDA VALUES ARE
TO
C        BE PRINTED AFTER EACH LOAD INCREMENT (IN RESULT)
C        END WITH 0
C
C***********************************************************************
C*                        MAIN PROGRAM                         *
C***********************************************************************
C
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DOUBLE PRECISION TIMEA, TIMEB, DTIME
      CHARACTER*(*) TITLE, UNITS, DATE*8, FNAME*12
      PARAMETER (LIM = 100000, TITLE = 'PLANE TRUSS ANALYSIS',
     $          UNITS = 'UNITS: KIP, INCH, RADIAN, FAHRENHEIT')
      DIMENSION A(LIM),IA(LIM)
C
C    RESERVE MEMORY; READ AND ECHO NE, NJ; SET POINTERS FOR DATA
C    ARRAYS; IF MEMORY IS ADEQUATE CALL STRUCT, ELSE SEND MESSAGE AND
C    STOP; SET POINTERS FOR SOLUTION ARRAYS; IF MEMORY IS ADEQUATE,
C    CALL LOAD. THEN CALL NEWRAP OR RIKWEM
C-----------------------------------------
C    OPEN DATA FILES: ********* FOR PC ONLY*******
C    WRITE(6,*) 'INPUT DATA FILE: '
C    READ(*,'(A)') FNAME
C***  OPEN(5,FILE = FNAME)
C***  OPEN(7,FILE = 'RW.OUT', STATUS = 'UNKNOWN')
C-----------------------------------------
C
C
      READ(5,'(A)') DATE
      WRITE(6,5) TITLE,'(EARLS, 1992)',
     $'DATE: ',DATE,UNITS
 5    FORMAT('1',T5,73('*')/T5,'*',T32,A,T77,'*'/T5,'*',T35,A,T77,'*'/
     $          T5,73('*')//T64,2(A)///T5,A)
      READ(5,*)METHOD
      READ(5,*) NE, NJ
      WRITE(6,15) 'CONTROL VARIABLES',
     $          'NUMBER OF ELEMENTS',NE,'NUMBER OF JOINTS',NJ
 15   FORMAT(///T5,A/2(T5,A,T30,I4/))
C
C--- SET POINTERS FOR DATA ARRAYS (IN STRUCT):
C
      NP=1
      NAREA=NP+3*NJ
      NEMOD=NAREA+NE
      NELENG=NEMOD+NE
      NC1=NELENG+NE
     NC2=NC1+NE
     NC3=NC2+NE
     NMCODE=NC3+NE
     NJCODE=NMCODE+6*NE
     NMINC=NJCODE+3*NJ
     NMAXA=NMINC+2*NE
C     TEMPORARILY LET NEQ=3*NJ TO SET POINTERS FOR NKHT AND MAX
     NEQ=3*NJ
     NKHT=NMAXA+(NEQ+1)
     MAX=NKHT+NEQ-1
C
C--- IF MEMORY IS ADEQUATE CALL STRUCT, ELSE SEND MESSAGE AND STOP.
C
     IF(MAX .LE. LIM) THEN
       CALL STRUCT(A(NP),A(NAREA),A(NEMOD),A(NELENG),
    $            A(NC1),A(NC2),A(NC3),IA(NMAXA),IA(NKHT),
    $            IA(NMCODE),IA(NJCODE),IA(NMINC),NE,NJ,NEQ,LSS)
C
C--- SET POINTERS FOR SOLUTION ARRAYS (IN LOAD AND NEWRAP OR RIKWEM)
C
C--- TEMPORARILY SET LSS=NEQ
C       LSS = NEQ
C
       NF=NKHT+NEQ
       NSS=NF+NEQ
       NQ=NSS+LSS
       NQT=NQ+NEQ
       NR=NQT+NEQ
       NFP=NR+NEQ
       ND=NFP+NEQ
       NDD=ND+NEQ
       NDDO=NDD+NEQ
       NDD01=NDDO+NEQ
       NDD1=NDD01+NEQ
       NDD2=NDD1+NEQ
       NDDP=NDD2+NEQ
       NFPI=NDDP+NEQ
       NDEFLN=NFPI+NEQ
       NELONG=NDEFLN+NE
       NTT=NELONG+NE
       MAX=NTT+NEQ-1
C
C--- IF MEMORY IS ADEQUATE CALL FOR EACH LOAD CONDITION LOAD,
C    NEWRAP OF RIKWEM.
C
       IF(MAX .LE. LIM) THEN
           CALL LOAD(A(NQ),IA(NJCODE),NEQ)
C
C--- READ QIMAX, QI ,DQI
         READ(5,*)QIMAX,QI,DQI
C
```

```
        WRITE(6,25) 'QIMAX = ', QIMAX,'  QI = ',QI,'  DQI = ',DQI
 25      FORMAT(/3(T10,A,F12.6/))
C
C--- READ ITEUPD,ITEMAX,ITEDES
        READ(5,*)ITENUM, ITEMAX, ITEDES
C
        WRITE(6,*)'NUMBER OF ITERATIONS FOR UPDATING STIFFNESS',
   $    ' MATRIX = ',ITEUPD
        WRITE(6,*)'MAXIMUM NUMBER OF ITERATIONS IN EACH LOAD S',
   $    'TEP     = ',ITEMAX
        WRITE(6,*)'DESIRED NUMBER OF ITERATIONS IN EACH LOAD S',
   $    'TEP     = ',ITEDES
C
C--- READ TOLDIS,TOLFOR
        READ(5,*)TOLDIS,TOLFOR
C
        WRITE(6,35) 'TOLERANCE IN DISPLACEMENT  = ',  TOLDIS,
   $           'TOLERANCE IN FORCE          = ',  TOLFOR
 35      FORMAT(/2(T10,A,T45,F12.6/))
C
        LSTEP = 1
C
C
C--- INSERT CALL TO INITIALIZE SYSTEM TIMER HERE (FOR VM1)
        CALL CPUTIME(TIMEA, IRCD_A)
C
        IF(METHOD .EQ. 1)THEN
          WRITE(6,40) 'ITERATIVE METHOD: NEWTON-RAPHSON '
 40        FORMAT(T10,A)
C
          CALL NEWRAP(A(ND),A(NDD),A(NQ),A(NQT),A(NTT),A(NSS),
   $             A(NAREA),A(NEMOD),A(NELENG),A(NC1),
   $             A(NC2),A(NC3),A(NELONG),A(NDEFLN),A(NF),
   $             A(NFP),A(NR),A(NP),A(NDDO),A(NFPI),
   $             IA(NMAXA),IA(NMCODE),IA(NJCODE),IA(NMINC),
   $             INCONV,ITECNT,ITENUM,ITEUPD,ITEMAX,NE,NEQ,NJ,
   $             LSS,QIMAX,QI,DQI,TOLDIS,TOLFOR,LSTEP,NFE)
C
        ELSEIF(METHOD .EQ. 2)THEN
          WRITE(6,50) 'ITERATIVE METHOD: MODIFIED RIKS/WEMPNER '
 50        FORMAT(T10,A)
C
          CALL RIKWEM(A(ND),A(NDD),A(NDDO),A(NDD01),A(NDD1),A(NDD2),
   $             A(NDDP),A(NAREA),A(NEMOD),A(NELENG),
   $             A(NC1),A(NC2),A(NC3),A(NQ),A(NQT),
   $             A(NSS),A(NTT),A(NP),A(NELONG),A(NDEFLN),
   $             A(NF),A(NFP),A(NFPI),A(NR),
   $             IA(NJCODE),IA(NMAXA),IA(NMCODE),IA(NMINC),
   $             IT,INCONV,ITECNT,ITENUM,ITEMAX,ITEUPD,ITEDES,
   $             NE,NEQ,NJ,LSS,QIMAX,QI,DQI,
   $             TOLDIS,TOLFOR,LSTEP,NFE)
```

```
C
          ENDIF
C
        ELSE
          WRITE(6,'(T10,A/)') 'ERROR MESSAGE: INCREASE MEMORY'
        END IF
      ELSE
        WRITE(6,'(T10,A/)') 'ERROR MESSAGE: INCREASE MEMORY'
      ENDIF
C
C
C INSERT CALL TO RETURN EXECUTION TIME IN MICROSECONDS IN DTIME.
C
       CALL CPUTIME(TIMEB, IRCD_B)
       IF(IRCD_A .NE. 8 .AND. IRCD_B .EQ. 0)THEN
           DTIME = (TIMEB - TIMEA)
       ENDIF
C
        WRITE(7,*) ' CPUTIME (MICROSECONDS)  = ', DTIME
        WRITE(7,*) '                    NFE   = ', NFE
C
      STOP
      END
C**********************************************************************
C*                        STRUCT                         *
C**********************************************************************
      SUBROUTINE STRUCT(X,AREA,EMOD,ELENG,C1,C2,C3,
     $             MAXA,KHT,MCODE,JCODE,MINC,NE,NJ,NEQ,LSS)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION X(3,*),AREA(*),EMOD(*),ELENG(*),C1(*),C2(*)
     $        ,C3(*),MAXA(*),KHT(*),MCODE(6,*),JCODE(3,*),MINC(2,*)
C
C     READ AND ECHO THE MEMBER INCIDENCES, MINC(L,I); INITIALIZE THE
C     ELEMENTS OF THE JOINT CODE MATRIX, JCODE, TO UNITY, READ AND
C     ECHO FOR EACH JOINT CONSTRAINT THE JOINT NUMBER, JNUM, AND
C     JOINT DIRECTION, JDIR, AND STORE A ZERO IN THE CORRESPONDING
C     LOCATION OF JCODE (END OF DATA MARKER JNUM=0); CALL CODES,
C     SKYLIN, AND PROP.
C
      WRITE(6,10) 'MEMBER INCIDENCES','MEMBER','A-END','B-END'
   10 FORMAT(///T10,A//T10,A,T17,A,T23,A/)
      DO 30 I=1,NE
         READ(5,*) MINC(1,I),MINC(2,I)
         WRITE(6,20) I,MINC(1,I),MINC(2,I)
   20    FORMAT(T11,I3,T18,I3,T24,I3)
   30 CONTINUE
C
      DO 60 J=1,NJ
         DO 50 L=1,2
            JCODE(L,J)=1
   50    CONTINUE
```

```
  60 CONTINUE
C
    WRITE(6,70) 'JOINT CONSTRAINTS','JOINT','DIRECTION'
  70 FORMAT(///T10,A//T10,A,T17,A/)
    READ(5,*) JNUM,JDIR
  80 IF (JNUM.NE.0) THEN
       WRITE(6,90) JNUM,JDIR
  90    FORMAT(T9,I3,T19,I3)
       JCODE(JDIR,JNUM)=0
       READ(5,*) JNUM,JDIR
       GO TO 80
    END IF
C
    CALL CODES(MCODE,JCODE,MINC,NE,NJ,NEQ)
C
    CALL SKYLIN(KHT,MAXA,MCODE,NE,NEQ,LSS)
C
    CALL PROP(X,AREA,EMOD,ELENG,C1,C2,C3,MINC,NE,NJ)
C
    RETURN
    END
C**********************************************************************
C*                          CODES                          *
C**********************************************************************
    SUBROUTINE CODES(MCODE,JCODE,MINC,NE,NJ,NEQ)
    IMPLICIT DOUBLE PRECISION (A-H, O-Z)
    DIMENSION MCODE(6,*),JCODE(3,*),MINC(2,*)
C
C
C    GENERATE JOINT CODE, JCODE, BY ASSIGNING INTEGERS IN SEQUENCE,
C    BY COLUMNS, TO ALL NONZERO ELEMENTS OF JCODE FROM 1 T0 NEQ;
C      GENERATE THE MEMBER CODE, MCODE, BY TRANSFERRING VIA MINC
COLUMNS
C    OF JCODE INTO COLUMNS OF MCODE.
C
C    GENERATE JCODE
C
    NEQ=0
    DO 20 J=1,NJ
      DO 10 L=1,2
        IF(JCODE(L,J).NE.0) THEN
          NEQ=NEQ+1
          JCODE(L,J)=NEQ
        END IF
  10   CONTINUE
  20 CONTINUE
C
    WRITE(6,*) 'NEQ = ', NEQ
C   GENERATE MCODE
C
    DO 40 I=1,NE
```

```
        J=MINC(1,I)
        K=MINC(2,I)
        DO 30 L=1,2
          MCODE(L,I)=JCODE(L,J)
          MCODE(L+2,I)=JCODE(L,K)
   30   CONTINUE
   40 CONTINUE
C
C     WRITE(6,50)'MCODE(TRANSPOSED)'
C  50 FORMAT(/T10,A/)
C     DO 60 I=1,NE
C       PRINT*,(MCODE(L,I),L=1,4)
C  60 CONTINUE
C
C
      RETURN
      END
C**********************************************************************
C*                         SKYLIN                                *
C**********************************************************************
C
C     SKYLIN DETERMINES KHT USING MCODE, AND DETERMINES MAXA FROM
KHT.
*
*    I HAVE MODIFIED SKYLIN SUCH THAT DOF IN EACH COLUMN OF MCODE
*      NEED NOT BE ARRANGED IN INCREASING ORDER FROM TOP TO BOTTOM;
I.E.,
*        ELEMENTS NEED NOT BE DIRECTED FROM A LOWER TO A HIGHER
NUMBERED
*    JOINT(HOLZER 2/8/91).
C
      SUBROUTINE SKYLIN(KHT,MAXA,MCODE,NE,NEQ,LSS)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION KHT(*),MAXA(*),MCODE(6,*)
C
C
      DO 10 I=1,NEQ
        KHT(I)=0
   10 CONTINUE
C
C     GENERATE KHT
C
      DO 30 I=1,NE
        MIN=NEQ
        DO 15 L=1,4
          IF (MCODE(L,I) .GT. 0 .AND. MCODE(L,I) .LT. MIN) THEN
            MIN=MCODE(L,I)
          END IF
   15   CONTINUE
C
        DO 20 L=1,4
```

```
           K=MCODE(L,I)
           IF(K.NE.0) THEN
             KHT(K)=MAX0(KHT(K),(K-MIN))
           ENDIF
   20    CONTINUE
   30 CONTINUE
*      WRITE(6,100)
*  100 FORMAT(//11X,'I',10X,'KHT(I)',10X,'MAXA(I)')
C
C     GENERATE MAXA
C
      MAXA(1)=1
      DO 40 I=1,NEQ
*         WRITE(6,200) I,KHT(I),MAXA(I)
*  200    FORMAT(7X,I5,9X,I5,11X,I5)
         MAXA(I+1)=MAXA(I)+KHT(I)+1
   40 CONTINUE
      LSS=MAXA(NEQ+1)-1
      I=NEQ+1
*      WRITE(6,300) I,MAXA(I),LSS
*  300 FORMAT(7X,I5,25X,I5//7X,'LSS = ',I5)
C
       WRITE(6,*)' LSS = ', LSS
C
      RETURN
      END
C***********************************************************************
C*                              PROP                          *
C***********************************************************************
      SUBROUTINE PROP(X,AREA,EMOD,ELENG,C1,C2,C3,MINC,NE,NJ)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION X(3,*),AREA(*),EMOD(*),ELENG(*),C1(*),C2(*)
     $         ,C3(*),MINC(2,*)
C
C
C     READ AND ECHO JOINT COORDINATES, X(L,J); COMPUTE FOR EACH
C     ELEMENT BY THE LENGTH, ELENG(I), AND THE DIRECTION COSINES
C     C1(I), C2(I) & C3(I); READ FOR EACH ELEMENT THE CROSS SECTIONAL
C     AREA, AREA(I), THE MODULUS OF ELASTICITY, EMOD(I)
C     PRINT ELEMENT PROPERTIES.
C
      WRITE(6,10) 'JOINT COORDINATES','JOINT','DIRECTION-1',
     $          'DIRECTION-2','DIRECTION-3'
   10 FORMAT(///T10,A//T10,A,T17,A,T30,A,T43,A)
      DO 30 J=1,NJ
         READ(5,*) X(1,J),X(2,J),X(3,J)
         WRITE(6,20) J,X(1,J),X(2,J),X(3,J)
   20    FORMAT(T9,I3,T18,F8.2,T31,F8.2,T44,F8.2)
   30 CONTINUE
C
      WRITE(6,40) 'ELEMENT PROPERTIES','ELASTIC',
     $       'ELEMENT','AREA','MODULUS','LENGTH'
   40 FORMAT(///T10,A//T39,A/T10,A,T19,A,T39,A,T62,A)
C
      DO 60 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)
         EL3=X(3,K)-X(3,J)
         ELENG(I)=DSQRT(EL1**2+EL2**2+EL3**2)
         C1(I)=EL1/ELENG(I)
         C2(I)=EL2/ELENG(I)
         C3(I)=EL3/ELENG(I)
C
         READ(5,*) AREA(I),EMOD(I)
         WRITE(6,50) I,AREA(I),EMOD(I),ELENG(I)
         WRITE(7,*) I, ELENG(I)
   50    FORMAT(T12,I3,T17,E9.3,T39,E9.3,T60,F8.3)
   60 CONTINUE
C
      RETURN
      END
C**********************************************************************
C*                         LOAD                          *
C**********************************************************************
      SUBROUTINE LOAD(Q,JCODE,NEQ)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION Q(*),JCODE(3,*)
C
C     INITIALIZE TO ZERO THE JOINT LOAD VECTOR, Q.
C     CALL JLOAD.
C
C
      DO 20 K=1,NEQ
         Q(K)=0.0
   20 CONTINUE
C
      CALL JLOAD(Q,JCODE)
C
      RETURN
      END
C**********************************************************************
C*                         JLOAD                         *
C**********************************************************************
      SUBROUTINE JLOAD(Q,JCODE)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION Q(*),JCODE(3,*)
C
C     READ THE JOINT NUMBER, JNUM, THE JOINT DIRECTION, JDIR, AND THE
C     APPLIED FORCE, FORCE; WHILE JNUM IS NOT EQUAL TO ZERO, PRINT JNUM,
C     JDIR, FORCE,STORE FORCE IN Q, AND READ JNUM, JDIR, FORCE.
```

```
C
     READ(5,*) JNUM,JDIR,FORCE
     IF(JNUM.NE.0) THEN
*        WRITE(6,10) 'JOINT LOADS','GLOBAL','JOINT','DIRECTION','FORCE'
*  10    FORMAT(///T10,A/T10,11('-')/T18,A/T10,A,T17,A,T35,A)
   20    IF (JNUM.NE.0) THEN
*          WRITE(6,30) JNUM,JDIR,FORCE
*  30      FORMAT(T11,I3,T21,I1,T28,F15.8)
           K=JCODE(JDIR,JNUM)
           Q(K)=FORCE
           READ(5,*) JNUM,JDIR,FORCE
           GO TO 20
         END IF
     ELSE
       WRITE(6,40) 'NO JOINT FORCES'
   40  FORMAT(///T10,A)
     END IF
C
     RETURN
     END
C***********************************************************************
C*                          NEWRAP                             *
C***********************************************************************
     SUBROUTINE NEWRAP(D,DD,Q,QT,TT,SS,AREA,EMOD,ELENG,C1,C2,C3,
    $             ELONG,DEFLEN,F,FP,R,X,DDO,FPI,
    $             MAXA,MCODE,JCODE,MINC,
    $             INCONV,ITECNT,ITENUM,ITEUPD,ITEMAX,NE,NEQ,NJ,
    $             LSS,QIMAX,QI,DQI,TOLDIS,TOLFOR,LSTEP,NFE)
     IMPLICIT DOUBLE PRECISION (A-H, O-Z)
     DIMENSION D(*),DD(*),Q(*),QT(*),TT(*),SS(*),AREA(*),EMOD(*),
    $       ELENG(*),C1(*),C2(*),C3(*),ELONG(*),DEFLEN(*),F(*),
    $       FP(*),R(*),X(3,*),DDO(*),FPI(*),
    $       MAXA(*),MCODE(6,*),JCODE(3,*),MINC(2,*),C1S(40),C2S(40)
C
     DO 5 I = 1,NE
       C1S(I)=C1(I)
       C2S(I)=C2(I)
 5   CONTINUE
C
     DO 10 I = 1, NEQ
       D(I) = 0.D0
       DD(I) = 0.D0
       F(I) = 0.D0
       FP(I) = 0.D0
       FPI(I) = 0.D0
 10  CONTINUE
C
     DO 15 I = 1, NE
      DEFLEN(I) = ELENG(I)
      ELONG(I)  = 0.D0
 15  CONTINUE
```

```
C
 30  IF (QI .LE. QIMAX) THEN
C
       DO 40 I = 1, NEQ
         QT(I) = Q(I) * QI
 40     CONTINUE
C
       CALL NRITER(D,DD,Q,QT,TT,SS,AREA,EMOD,ELENG,C1,C2,C3,
    $            ELONG,DEFLEN,F,FP,R,X,DDO,FPI,
    $            MAXA,MCODE,JCODE,MINC,
    $            INCONV,ITECNT,ITENUM,ITEUPD,ITEMAX,NE,NEQ,NJ,
    $            LSS,QIMAX,QI,DQI,TOLDIS,TOLFOR,NFE,C1S,C2S)
C
       DO 60 I = 1, NEQ
          FP(I) = F(I)
 60     CONTINUE
C
       IF (INCONV .NE. 0) THEN
         WRITE(6,*)'*****ERROR *****SOLUTION FAILS TO CONVERGE IN',
    $             ' GIVEN NUMBER OF ITERATIONS'
         STOP
       ELSE
C         CALL RESULT(D,MCODE,JCODE,MINC,NE,NJ,NEQ,LSTEP,QI)
       WRITE(7,*) D(6),',',QI
C       WRITE(*,*) D(4),',',QI
       ENDIF
C
       QI = QI + DQI
C
       GO TO 30
     ENDIF
C
     RETURN
     END
C**********************************************************************
C*                        NRITER                              *
C**********************************************************************
     SUBROUTINE NRITER(D,DD,Q,QT,TT,SS,AREA,EMOD,ELENG,C1,C2,C3,
    $            ELONG,DEFLEN,F,FP,R,X,DDO,FPI,
    $            MAXA,MCODE,JCODE,MINC,
    $            INCONV,ITECNT,ITENUM,ITEUPD,ITEMAX,NE,NEQ,NJ,
    $            LSS,QIMAX,QI,DQI,TOLDIS,TOLFOR,NFE,C1S,C2S)
     IMPLICIT DOUBLE PRECISION (A-H, O-Z)
     DIMENSION D(*),DD(*),Q(*),QT(*),TT(*),SS(*),AREA(*),EMOD(*),
    $      ELENG(*),C1(*),C2(*),C3(*),ELONG(*),DEFLEN(*),F(*),
    $      FP(*),R(*),X(3,*),DDO(*),FPI(*),
    $      MAXA(*),MCODE(6,*),JCODE(3,*),MINC(2,*),C1S(40),C2S(40)
C
C
     CALL FORCES(F,QT,AREA,EMOD,ELENG,C1,C2,C3,ELONG,R,MCODE,MINC,
    $            NE,NEQ,D,C1S,C2S,DD)
```

```
C
      ITENUM = ITEUPD
      ITECNT = 1
      INCONV = 1
C
 10   IF (INCONV .NE. 0 .AND. ITECNT .LE. ITEMAX) THEN
C
      DO 15 I = 1, NEQ
         R(I) = QT(I) - F(I)
C      WRITE(*,*)'RES = ',R(I)
 15    CONTINUE
C      WRITE(*,*)'  '
C
      IF (ITENUM .GE. ITEUPD) THEN
        CALL STIFF(SS,AREA,EMOD,ELENG,C1,C2,C3,ELONG,DEFLEN,
     $             MAXA,MCODE,NE,LSS)
        ITENUM = 0
      ENDIF
C
      DO 20 I = 1, NEQ
         TT(I) = 0.D0
         TT(I) = R(I)
 20    CONTINUE
C
      CALL SOLVE(SS,TT,MAXA,NEQ,1)
C      WRITE(*,*)'-----*',TT(1)
C
      DO 40 I = 1, NEQ
         DD(I) = TT(I)
         TT(I) = 0.D0
 40    CONTINUE
C
      IF (ITECNT .EQ. 1) THEN
       DO 50 I = 1, NEQ
         DDO(I) = DD(I)
 50     CONTINUE
      ENDIF
C
      DO 60 I = 1, NEQ
         D(I) = D(I) + DD(I)
C      WRITE(*,*)'DISPL*********',D(I)
 60    CONTINUE
C      WRITE(*,*)'  '
C
      CALL UPDATC(X,DD,C1,C2,C3,ELONG,ELENG,DEFLEN,MINC,JCODE,
     $          NE,NJ,NEQ)
C
      CALL FORCES(F,QT,AREA,EMOD,ELENG,C1,C2,C3,ELONG,R,MCODE,MINC,
     $          NE,NEQ,D,C1S,C2S,DD)
C
      CALL TEST(D,DD,DDO,F,FP,FPI,QT,INCONV,NEQ,
     $          TOLFOR,TOLDIS)
C
        DO 80 I = 1, NEQ
           FPI(I) = F(I)
 80     CONTINUE
C
        ITECNT = ITECNT + 1
        ITENUM = ITENUM + 1
C
        GO TO 10
      ENDIF
C
      RETURN
      END
C************************************************************************
C*                      RIKWEM                          *
C************************************************************************
      SUBROUTINE RIKWEM(D,DD,DDO,DD01,DD1,DD2,DDP,AREA,EMOD,ELENG,
     $           C1,C2,C3,Q,QT,SS,TT,X,ELONG,DEFLEN,F,FP,FPI,R,
     $           JCODE,MAXA,MCODE,MINC,
     $           IT,INCONV,ITECNT,ITENUM,ITEMAX,ITEUPD,ITEDES,
     $           NE,NEQ,NJ,LSS,QIMAX,QI,DQI,
     $           TOLDIS,TOLFOR,LSTEP,NFE)
C
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
C
      DIMENSION D(*),DD(*),DDO(*),DD01(*),DD1(*),DD2(*),DDP(*),AREA(*),
     $       EMOD(*),ELENG(*),C1(*),C2(*),C3(*),Q(*),QT(*),
     $       SS(*),TT(*),X(3,*),ELONG(*),DEFLEN(*),
     $       F(*),FP(*),FPI(*),R(*),
     $       JCODE(3,*),MAXA(*),MCODE(6,*),MINC(2,*)
C
C---- INITIALIZE THE VARIABLES D,DD,F,FP,FPI TO ZERO
C
      DO 10 I = 1, NEQ
        D(I) = 0.D0
        DD(I) = 0.D0
        F(I) = 0.D0
        FP(I) = 0.D0
        FPI(I) = 0.D0
 10   CONTINUE
C
C---- INITIALIZE THE VARIABLES DEFLEN, ELONG
C
      DO 15 I = 1, NE
       DEFLEN(I) = ELENG(I)
       ELONG(I)  = 0.D0
 15   CONTINUE
C
      NFE = 0
      ITECNT = 1
```

```
C
 20  IF (QI .LE. QIMAX .AND. ITECNT .LE. 50)THEN
C     WRITE(6,'(T10,A,3X,F16.10)') 'QI = ',QI
C
C--- COMPUTE THE TANGENT STIFFNESS MATRIX
C
      CALL STIFF(SS,AREA,EMOD,ELENG,C1,C2,C3,ELONG,DEFLEN,
     $           MAXA,MCODE,NE,LSS)
C
C--- COMPUTE THE FIRST TRIAL SOLUTION
C
      DO 40 I = 1, NEQ
        TT(I) = 0.D0
        TT(I) = Q(I)
 40   CONTINUE
C
      CALL SOLVE(SS,TT,MAXA,NEQ,1)
C
      DO 50 I = 1, NEQ
        DD01(I) = TT(I)
        TT(I) = 0.D0
 50   CONTINUE
C
C--- COMPUTE THE ARC LENGTH FOR THE FIRST ITERATION
C    FOR SUBSEQUENT ITERATIONS COMPUTE THE LOAD INCREMENT
C
      IF(ITECNT .EQ. 1)THEN
        DS=DQI*DSQRT(DOTPRD(DD01,DD01,NEQ)+1.D0)
        DSMAX=DS*1.0D0
      ELSE
        DQI=DS/DSQRT(DOTPRD(DD01,DD01,NEQ)+1.D0)
        TEMP=DQI*(DOTPRD(DD01,DDP,NEQ)+DQII)
        IF(TEMP .GT. 0)THEN
          SGN=1
        ELSE
          SGN=-1
        ENDIF
        DQI = SGN*DQI
      ENDIF
C
      DQII = 0.0
      DQI1 = DQI
      DQII = DQII + DQI
C
C--- COMPUTE THE INCREMENT IN DISPLACEMENT AND TOTAL DISPLACEMENT
C
      DO 60 I = 1, NEQ
        DDO(I) = DQI*DD01(I)
        D(I) = D(I) + DDO(I)
        DDP(I) = DDO(I)
 60   CONTINUE
```

```
C
C--- INCREMENT THE LOAD PARAMETER
C
          QI = QI + DQI
C      WRITE(6,*) ' QI ',QI
C--- UPDATE THE COORDINATES
        CALL UPDATC(X,DDO,C1,C2,C3,ELONG,ELENG,DEFLEN,MINC,JCODE,
     $            NE,NJ,NEQ)
C
        CALL FORCES(F,QT,AREA,EMOD,ELENG,C1,C2,C3,ELONG,R,MCODE,MINC,
     $            NE,NEQ)
C
C
C--- CALL SUBROUTINE TRLVCT
C
        CALL TRLVCT(D,DD,DDO,DD1,DD2,DDP,AREA,EMOD,ELENG,
     $        C1,C2,C3,Q,QT,SS,TT,X,ELONG,DEFLEN,F,FP,FPI,R,
     $        JCODE,MAXA,MCODE,MINC,
     $        IT,INCONV,ITECNT,ITENUM,ITEMAX,ITEUPD,NE,NEQ,
     $        NJ,LSS,QIMAX,QI,DQI,TOLDIS,TOLFOR,DQI1,DQII,NFE)
C
C
        DO 80 I = 1, NEQ
            FP(I) = F(I)
 80      CONTINUE
C
C
        IF(INCONV .NE. 0)THEN
          WRITE(6,*) '**** ERROR **** SOLUTION FAILS TO CONVERGE',
     $    ' IN GIVEN NUMBER OF ITERATIONS'
          STOP
        ELSE
          CALL RESULT(D,MCODE,JCODE,MINC,NE,NJ,NEQ,LSTEP,QI)
C
        ENDIF
C
           ITECNT = ITECNT + 1
C
C---COMPUTE THE SCALED ARC LENGTH
C
         DS=DS*DSQRT((1.D0*ITEDES)/(1.D0*IT))
         IF(DABS(DS) .GT. DSMAX)THEN
             IF(DS .LT. 0)THEN
               DS = -DSMAX
             ELSE
               DS = DSMAX
             ENDIF
         ENDIF
C
         GO TO 20
      ENDIF
```

```
C
        RETURN
        END
C************************************************************************
C*                      TRLVCT                          *
C************************************************************************
      SUBROUTINE TRLVCT(D,DD,DDO,DD1,DD2,DDP,AREA,EMOD,ELENG,
     $             C1,C2,C3,Q,QT,SS,TT,X,ELONG,DEFLEN,F,FP,FPI,R,
     $             JCODE,MAXA,MCODE,MINC,
     $             IT,INCONV,ITECNT,ITENUM,ITEMAX,ITEUPD,NE,NEQ,
     $             NJ,LSS,QIMAX,QI,DQI,TOLDIS,TOLFOR,
     $             DQI1,DQII,NFE)
C
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
C
      DIMENSION D(*),DD(*),DDO(*),DD1(*),DD2(*),DDP(*),AREA(*),
     $        EMOD(*),ELENG(*),C1(*),C2(*),C3(*),Q(*),QT(*),
     $        SS(*),TT(*),X(3,*),ELONG(*),DEFLEN(*),
     $        F(*),FP(*),FPI(*),R(*),
     $        JCODE(3,*),MAXA(*),MCODE(6,*),MINC(2,*)
C
C
      ITENUM = ITEUPD
      IT = 1
      INCONV = 1
C
 10   IF(INCONV .NE. 0 .AND. IT .LE. ITEMAX)THEN
C
C        WRITE(6,'(T8,A,3X,I3)') 'ITERATION ', IT
C
         DO 15 I = 1, NEQ
            FPI(I) = F(I)
 15      CONTINUE
C--- UPDATE THE LOAD VECTOR
        DO 20 I = 1, NEQ
          QT(I) = Q(I) * QI
 20      CONTINUE
C
C--- COMPUTE  UNBALANCED FORCES
C
        DO 25 I = 1, NEQ
           R(I) = QT(I) - F(I)
 25      CONTINUE
C
C--- COMPUTE THE TANGENT STIFFNESS MATRIX
        IF (ITENUM .GE. ITEUPD) THEN
          CALL STIFF(SS,AREA,EMOD,ELENG,C1,C2,C3,ELONG,DEFLEN,
     $             MAXA,MCODE,NE,LSS)
          ITENUM = 0
        ENDIF
C
```

```
C--- SOLVE FOR DD1
      DO 30 I = 1, NEQ
         TT(I) = 0.D0
         TT(I) = Q(I)
 30   CONTINUE
C
      CALL SOLVE(SS,TT,MAXA,NEQ,1)
C
      DO 40 I = 1, NEQ
         DD1(I) = TT(I)
         TT(I) = 0.D0
 40   CONTINUE
C
C--- SOLVE FOR DD2
      DO 50 I = 1, NEQ
         TT(I) = 0.D0
         TT(I) = R(I)
 50   CONTINUE
C
      CALL SOLVE(SS,TT,MAXA,NEQ,2)
C
      DO 60 I = 1, NEQ
         DD2(I) = TT(I)
         TT(I) = 0.D0
 60   CONTINUE
C
C--- COMPUTE THE INCREMENT IN LOAD PARAMETER
      DQI = -(DOTPRD(DDO,DD2,NEQ))/(DOTPRD(DDO,DD1,NEQ)+DQI1)
      DQII = DQII + DQI
      QI = QI + DQI
C
      DO 70 I = 1, NEQ
        DD(I) = DQI*DD1(I) + DD2(I)
        D(I) = D(I) + DD(I)
        DDP(I) = DDP(I) + DD(I)
 70   CONTINUE
C
C       WRITE(6,'(T10,A,3X,F20.15)') 'QI = ',QI
C       WRITE(6,'(T10,A,2(5X,F20.15))') 'D1, D2', D(1),  D(2)
C
C--- UPDATE THE JOINT COORDINATES
       CALL UPDATC(X,DD,C1,C2,C3,ELONG,ELENG,DEFLEN,MINC,JCODE,
     $             NE,NJ,NEQ)
C
C--- COMPUTE THE INTERNAL FORCES
       CALL FORCES(F,QT,AREA,EMOD,ELENG,C1,C2,C3,ELONG,R,MCODE,MINC,
     $             NE,NEQ)
C
       DO 80 I = 1, NEQ
C         WRITE(6,*) 'FORCE  =  ' , F(I)
 80     CONTINUE
```

```
C
C--- CHECK FOR CONVERGENCE
       CALL TEST(D,DD,DDO,F,FP,FPI,QT,INCONV,NEQ,
   $            TOLFOR,TOLDIS)
C
       IT = IT + 1
       ITENUM = ITENUM + 1
C
       GO TO 10
    ENDIF
      NFE = NFE+(IT-1)
      WRITE(6,*) ' NFE = ', NFE
C
    RETURN
    END
C***********************************************************************
C*                         DOTPRD                             *
C***********************************************************************
    FUNCTION DOTPRD(DOT1,DOT2,N)
    IMPLICIT DOUBLE PRECISION (A-H, O-Z)
    DIMENSION DOT1(*),DOT2(*)
C
C--- DOTPRD COMPUTES THE DOT PRODUCT OF DOT1 AND DOT2.
C
    DOTPRD=0.D00
    DO 10 I=1,N
      DOTPRD=DOTPRD+DOT1(I)*DOT2(I)
 10  CONTINUE
C
    RETURN
    END
C***********************************************************************
C*                         STIFF                             *
C***********************************************************************
    SUBROUTINE STIFF(SS,AREA,EMOD,ELENG,C1,C2,C3,ELONG,DEFLEN,
   $            MAXA,MCODE,NE,LSS)
    IMPLICIT DOUBLE PRECISION (A-H, O-Z)
    DIMENSION SS(*),AREA(*),EMOD(*),ELENG(*),C1(*),C2(*),C3(*),
   $      ELONG(*),DEFLEN(*),G(6),H(6),MAXA(*),MCODE(6,*)
C
C   INITIALIZE THE SYSTEM STIFFNESS MATRIX, SS, TO ZERO; FOR
C   EACH ELEMENT CALL ELEMS AND ASSEMS.
C
    DO 10 L = 1,LSS
      SS(L) = 0.D0
  10 CONTINUE
C
    DO 20 N = 1,NE
      CALL ELEMS(AREA,EMOD,ELENG,C1,C2,C3,G,N)
C      CALL NELEMS(AREA,EMOD,ELENG,C1,C2,C3,H,ELONG,DEFLEN,N)
      CALL ASSEMS(SS,G,H,MCODE,MAXA,N,LSS)
  20 CONTINUE
C
    DO 30 I = 1, LSS
C      WRITE(6,*) 'SS = ',SS(I)
  30 CONTINUE
C
    RETURN
    END
C*********************************************************************
C*                    ELEMS                         *
C*********************************************************************
    SUBROUTINE ELEMS(AREA,EMOD,ELENG,C1,C2,C3,G,N)
    IMPLICIT DOUBLE PRECISION (A-H, O-Z)
    DIMENSION AREA(*),EMOD(*),ELENG(*),C1(*),C2(*),C3(*),G(6)
C
C    FOR ELEMENT N, COMPUTE THE GLOBAL STIFFNESS COEFFICIENTS, G(6),
C    DEFINED IN EQS. 5.15 (HOLZER)
C
    GAMMA = AREA(N)*EMOD(N)/ELENG(N)
C
    G(1) = (GAMMA)*(C1(N)**2)
    G(2) = (GAMMA)*(C2(N)**2)
    G(3) =  GAMMA*C1(N)*C2(N)
C
    RETURN
    END
C*********************************************************************
C*                   NELEMS                         *
C*********************************************************************
    SUBROUTINE NELEMS(AREA,EMOD,ELENG,C1,C2,C3,H,ELONG,DEFLEN,N)
    IMPLICIT DOUBLE PRECISION (A-H, O-Z)
    DIMENSION AREA(*),EMOD(*),ELENG(*),C1(*),C2(*),C3(*),H(6),
   $        ELONG(*),DEFLEN(*)
C
C        FOR ELEMENT N, COMPUTE THE NON-LINEAR GLOBAL STIFFNESS
COEFFICIENTS,
C    H(6)
C
C
    GAMMA = AREA(N)*EMOD(N)/ELENG(N)
    P = ELONG(N)/DEFLEN(N)
C
    H(1) = GAMMA*P*(1-(C1(N)*C1(N)))
    H(2) = GAMMA*P*(1-(C2(N)*C2(N)))
    H(3) = -GAMMA*C1(N)*C2(N)*P
C
    RETURN
    END
C*********************************************************************
C*                   ASSEMS                         *
C*********************************************************************
```

```
      SUBROUTINE ASSEMS(SS,G,H,MCODE,MAXA,N,LSS)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION SS(*),G(*),H(*),MCODE(6,*),MAXA(*),INDEX(4,4)
      DATA INDEX/1,3,-1,-3, 3,2,-3,-2, -1,-3,1,3, -3,-2,3,2/
C
C
*     MODIFIED TO ALLOW DOF IN ANY COLUMN OF MCODE TO BE IN
*     ANY ORDER; SEE SKYLIN
*
C     INITIALIZE INDEX. ; ASSIGN STIFFNESS COEFFICIENTS, G(L),
C     OF ELEMENT N TO THE SYSTEM STIFFNESS MATRIX, SS, BY INDEX, MCODE,
C     AND MAXA.
C
      DO 20 JE = 1, 4
        J = MCODE(JE,N)
        IF (J .NE. 0) THEN
          DO 10 IE = 1, JE
            I = MCODE(IE,N)
            IF (I .NE. 0) THEN
              IF (I .GT. J) THEN
                K = MAXA(I) + I - J
              ELSE
                K = MAXA(J) + J - I
              END IF
              L = INDEX(IE,JE)
              IF (L .GT. 0) THEN
                SS(K) = SS(K) + G(L) + H(L)
              ELSE
                SS(K) = SS(K) - G(-L) - H(-L)
              END IF
            END IF
   10     CONTINUE
        END IF
   20 CONTINUE
C
      RETURN
      END
C**********************************************************************
C*                         SOLVE                         *
C**********************************************************************
      SUBROUTINE SOLVE(SS,Q,MAXA,NEQ,LC)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION SS(*),Q(*),MAXA(*)
C
C        SOLVE DETERMINES THE SOLUTION TO THE SYSTEM EQUATIONS BY
COMPACT
C     GAUSSIAN ELIMINATION (HOLZER, PP. 290, 296, 307) BASED ON THE
C     SUBROUTINE COLSOL (BATHE P. 721) AND THE MODIFICATION BY MICHAEL
C     BUTLER (MS 1984): IF LC = 1, CALL FACTOR,FORSUB,AND BACSUB
C     IF LC > 1, CALL FORSUB AND BACSUB.
C
```

```
C
    IF (LC.EQ.1) THEN
       CALL FACTOR(SS,MAXA,NEQ)
    END IF
       CALL FORSUB(SS,Q,MAXA,NEQ)
       CALL BACSUB(SS,Q,MAXA,NEQ)
C
    RETURN
    END
C*********************************************************************
C*                        FACTOR                          *
C*********************************************************************
C   FACTOR PERFORMS THE LDU FACTORIZATION OF THE STIFFNESS MATRIX.
C
    SUBROUTINE FACTOR(SS,MAXA,NEQ)
    IMPLICIT DOUBLE PRECISION (A-H, O-Z)
    DIMENSION SS(*),MAXA(*)
C
    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=MIN0(IC,ND)
      C=0.00
      DO 30 L=1,KK
  30   C=C+SS(KI+L)*SS(KLT+L)
      SS(KLT)=SS(KLT)-C
  40  K=K+1
  50  K=N
     B=0.00
     DO 60 KK=KL,KU
      K=K-1
      KI=MAXA(K)
      C=SS(KK)/SS(KI)
      B=B+C*SS(KK)
  60  SS(KK)=C
     SS(KN)=SS(KN)-B
C
C   STOP EXECUTION IF A ZERO PIVOT IS DETECTED
C
```

```
   70  IF(SS(KN).EQ.0.00) THEN
        PRINT 75,N,SS(KN)
   75   FORMAT('-STIFFNESS MATRIX IS NOT POSITIVE DEFINITE'/'0PIVOT IS
    $   ZERO FOR D.O.F. ',I4/'0PIVOT = ',E15.8)
        STOP
      END IF
C
   80 CONTINUE
C
      RETURN
      END
C************************************************************************
C*                       FORSUB                          *
C************************************************************************
C
C    FORSUB PERFORMS THE FORWARD SUBSTITUTION.
C
      SUBROUTINE FORSUB(SS,Q,MAXA,NEQ)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION SS(*),Q(*),MAXA(*)
C
      DO 20 N=1,NEQ
        KL=MAXA(N)+1
        KU=MAXA(N+1)-1
        KH=KU-KL
        IF(KH.GE.0) THEN
          K=N
          C=0.00
          DO 10 KK=KL,KU
            K=K-1
            C=C+SS(KK)*Q(K)
   10     CONTINUE
          Q(N)=Q(N)-C
        END IF
   20 CONTINUE
C
      RETURN
      END
C************************************************************************
C*                       BACSUB                          *
C************************************************************************
C
C    BACSUB PERFORMS BACK-SUBSTITUTION TO OBTAIN THE SOLUTION.
C
      SUBROUTINE BACSUB(SS,Q,MAXA,NEQ)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION SS(*),Q(*),MAXA(*)
C
      DO 10 N=1,NEQ
        K=MAXA(N)
        Q(N)=Q(N)/SS(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
         K=N
         DO 20 KK=KL,KU
           K=K-1
           Q(K)=Q(K)-SS(KK)*Q(N)
  20     CONTINUE
       END IF
       N=N-1
  30 CONTINUE
C
     RETURN
     END
C***********************************************************************
C*                        UPDATC                                       *
C***********************************************************************
     SUBROUTINE UPDATC(X,DD,C1,C2,C3,ELONG,ELENG,DEFLEN,MINC,JCODE,
    $               NE,NJ,NEQ)
     IMPLICIT DOUBLE PRECISION (A-H, O-Z)
     DIMENSION X(3,*),DD(*),C1(*),C2(*),C3(*),ELONG(*),ELENG(*),
    $         DEFLEN(*),MINC(2,*),JCODE(3,*)
C
     DO 30 J = 1, NJ
       DO 20 L = 1, 2
         K = JCODE(L,J)
         IF(K .NE. 0) THEN
            X(L,J) = X(L,J) + DD(K)
         ENDIF
 20      CONTINUE
 30  CONTINUE
C
     CALL LENGTH (X,C1,C2,C3,ELONG,ELENG,DEFLEN,MINC,NE)
C
     RETURN
     END
C***********************************************************************
C*                        LENGTH                                       *
C***********************************************************************
     SUBROUTINE LENGTH (X,C1,C2,C3,ELONG,ELENG,DEFLEN,MINC,NE)
     IMPLICIT DOUBLE PRECISION (A-H, O-Z)
     DIMENSION X(3,*),C1(*),C2(*),C3(*),ELONG(*),ELENG(*),DEFLEN(*),
    $         MINC(2,*)
C
     DO 10 I = 1, NE
C
```

```
       J = MINC(1,I)
       K = MINC(2,I)
C
       EL1 = X(1,K) - X(1,J)
       EL2 = X(2,K) - X(2,J)
       EL3 = X(3,K) - X(3,J)
       DEFLEN(I) = DSQRT((EL1**2)+(EL2**2)+(EL3**2))
       ELONG(I) =  DEFLEN(I) - ELENG(I)
       C1(I) = EL1/DEFLEN(I)
       C2(I) = EL2/DEFLEN(I)
       C3(I) = EL3/DEFLEN(I)
C
C       WRITE(*,*) ' '
C       WRITE(*,*) C1(I)
C       WRITE(*,*) C2(I)
C       WRITE(*,*) C3(I)
 10   CONTINUE
C
     RETURN
     END
C***********************************************************************
C*                        FORCES                         *
C***********************************************************************
     SUBROUTINE FORCES(F,QT,AREA,EMOD,ELENG,C1,C2,C3,ELONG,R,MCODE,
    $              MINC,NE,NEQ,D,C1S,C2S,DD)
     IMPLICIT DOUBLE PRECISION (A-H, O-Z)
     DIMENSION F(*),QT(*),AREA(*),EMOD(*),ELENG(*),C1(*),
    $        C2(*),C3(*),ELONG(*),R(*),MCODE(6,*),MINC(2,*),D(*),
    $        C1S(40),C2S(40),DD(*)
C
C    INITIALIZE THE FORCE VECTOR TO ZERO.
C    CALL INTERF.
C
     DO 10 I = 1,NEQ
       F(I) = 0.D0
 10  CONTINUE
C
     DO 20 I = 1, NE
       CALL INTERF(F,AREA,EMOD,ELENG,C1,C2,C3,ELONG,MCODE,I,D,C1S,C2S
    $,DD)
 20  CONTINUE
C    DO 30 I = 1,NEQ
C    WRITE(*,*)F(I)
C 30  CONTINUE
C
     RETURN
     END
C***********************************************************************
C*                        INTERF                         *
C***********************************************************************
     SUBROUTINE INTERF(F,AREA,EMOD,ELENG,C1,C2,C3,ELONG,MCODE,I,D,C1S,
     $             C2S,DD)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION F(*),AREA(*),EMOD(*),ELENG(*),C1(*),C2(*),C3(*),
     $        ELONG(*),MCODE(6,*),AAA(4),ZZZ(4),ELNG(60),D(*),C1S(40),
     $        C2S(40),DD(*),DDD(40),STRESS(40),STRES(40),DDEFLEN(40)
C
      CI1=C1(I)
      CI2=C2(I)
      CI3=C3(I)
      ELNG(I)=ELENG(I)
C
      GAM = DASIN(C2S(I)*C1(I)-C1S(I)*C2(I))
C
      DO 10 III=1,4
          ZZZ(III) = 0.0D0
          AAA(III) = 0.D0
 10   CONTINUE
C
      DO 26 II=1,4
        J = MCODE(II,I)
        IF (J .NE. 0) THEN
          ZZZ(II) = D(J)
          DDD(II) = DD(J)
        END IF
 26   CONTINUE

      DD1=C1S(I)*DDD(1)+C2S(I)*DDD(2)
      DD2=-C2S(I)*DDD(1)+C1S(I)*DDD(2)
      DD3=C1S(I)*DDD(3)+C2S(I)*DDD(4)
      DD4=-C2S(I)*DDD(3)+C1S(I)*DDD(4)
      ZZZ1=ZZZ(1)
      ZZZ2=ZZZ(2)
      ZZZ3=ZZZ(3)
      ZZZ4=ZZZ(4)
C
         AAA(1) =  C1S(I)*ZZZ1+C2S(I)*ZZZ2
         AAA(2) = -C2S(I)*ZZZ1+C1S(I)*ZZZ2
         AAA(3) =  C1S(I)*ZZZ3+C2S(I)*ZZZ4
         AAA(4) = -C2S(I)*ZZZ3+C1S(I)*ZZZ4
C
         AAA1=AAA(1)
         AAA2=AAA(2)
         AAA3=AAA(3)
         AAA4=AAA(4)
C
      STRES(I)=((DD3-DD1)/ELENG(I)+((DD1-DD3)*DD1+(DD2-DD4)*DD2+(DD3-DD1
     $)*DD3+(DD4-DD2)*DD4)/(ELENG(I)**2)+0.5D0*(((DD3-DD2)/ELENG(I))**2
     $+((DD4-DD2)/ELENG(I))**2))*EMOD(I)
       STRESS(I)=STRESS(I)+STRES(I)
          FL1  =AREA(I)*STRESS(I)*DCOS(GAM)*(-1.D0)
          FL2  =AREA(I)*STRESS(I)*DSIN(GAM)
```

```
           FL3 = -1.D0*FL1
           FL4 = -1.D0*FL2
C
      DO 20 L = 1, 4
        K = MCODE(L,I)
        IF (K .NE. 0) THEN
           IF (L .EQ. 1) THEN
          F(K) = (C1S(I)*FL1-C2S(I)*FL2) + F(K)
           ELSEIF (L .EQ. 2) THEN
          F(K) = (C2S(I)*FL1+C1S(I)*FL2) + F(K)
           ELSEIF (L .EQ. 3) THEN
          F(K) = (C1S(I)*FL3-C2S(I)*FL4) + F(K)
           ELSEIF (L .EQ. 4) THEN
          F(K) = (C2S(I)*FL3+C1S(I)*FL4) + F(K)
           ENDIF
        ENDIF
 20   CONTINUE
C
      RETURN
      END
C***********************************************************************
C*                        TEST                              *
C***********************************************************************
C        PERFORM CONVERGENCE TESTS, ASSUME THAT CONVERGENCE IS
REACHED,
C    ( SETTING INCONV=0 ) UNTIL IT IS PROVED THE CONTRARY. CALL
C    DISPL, UNBALF
C
      SUBROUTINE TEST(D,DD,DDO,F,FP,FPI,QT,INCONV,NEQ,
     $           TOLFOR,TOLDIS)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION D(*),DD(*),DDO(*),F(*),FP(*),FPI(*),QT(*)
C
      INCONV = 0.D0
      DIVER = 0.D0
C
      IF (TOLDIS .LT. 1.D0) THEN
        CALL DISPL(D,DD,INCONV,NEQ,TOLDIS)
      END IF
C
      IF (TOLFOR .LT. 1.D0) THEN
        CALL UNBALF (F,FP,QT,INCONV,NEQ,TOLFOR)
      END IF
C
C         WRITE(6,*) 'INCONV = ',INCONV
C
      RETURN
      END
C***********************************************************************
C*                        DISPL                             *
C***********************************************************************
```

```
C   PERFORM THE DISPLACEMENT CONVERGENCE TEST USING THE EUCLIDEAN
C   VECTOR NORM OF DISPLACEMENTS.
C
      SUBROUTINE DISPL(D,DD,INCONV,NEQ,TOLDIS)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION D(*),DD(*)
C
      DELTAD = 0.D0
      TOTALD = 0.D0
C
C   EUCLIDIAN VECTOR NORM OF DISPLACEMENTS ............................
      DO 10 I=1,NEQ
        DELTAD = DELTAD + (DD(I))**2
        TOTALD = TOTALD + (D(I))**2
   10 CONTINUE
C
C   CHECK WITH TOLERANCES ..........................................
      IF ( TOTALD.NE.0 ) THEN
        C = ( DSQRT(DELTAD) ) / ( DSQRT(TOTALD) )
        IF ( C.GT.TOLDIS ) THEN
          INCONV = INCONV + 10
        END IF
      ELSE
        WRITE(6,*)' ERROR: DISPLACEMENTS ARE ZERO'
        STOP
      END IF
C
      RETURN
      END
C***********************************************************************
C*                        UNBALF                          *
C***********************************************************************
C   PERFORM A CONVERGENCE TEST FOR THE UNBALANCED FORCE.
C
      SUBROUTINE UNBALF(F,FP,QT,INCONV,NEQ,TOLFOR)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION F(*),FP(*),QT(*)
C
      UNBFI = 0.D0
      UNBFP = 0.D0
C
C   COMPUTE THE UNBALANCED FORCE ......................................
C
      DO 10 I=1,NEQ
C       WRITE(*,*) '  '
C       WRITE(*,*) 'QT(I) =',QT(I),'F(I) =',F(I)
        UNBFI = UNBFI + (QT(I)-F(I))**2
C       WRITE(*,*) UNBFI
        UNBFP = UNBFP + (QT(I)-FP(I))**2
   10 CONTINUE
C
```

```
C    CHECK WITH TOLERANCES ........................................
C
      IF ( UNBFP.NE.0.D0 ) THEN
        C = ( DSQRT(UNBFI) ) / ( DSQRT(UNBFP) )
        IF ( C.GT.TOLFOR ) THEN
          INCONV = INCONV + 100
        END IF
      ELSE
        INCONV = INCONV + 100
      END IF

      RETURN
      END
C*********************************************************************
C*                         RESULT                    *
C*********************************************************************
      SUBROUTINE RESULT(D,MCODE,JCODE,MINC,NE,NJ,NEQ,LSTEP,QI)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION D(*),MCODE(6,*),JCODE(3,*),MINC(2,*)
C
      REWIND 3
C
      IF(LSTEP .EQ. 0)THEN
        NNN = NEQ + NJ + NE
        DO 5 I = 1, NNN
          J = 0
          WRITE (3) J
  5     CONTINUE
        LSTEP = LSTEP + 1
        REWIND 3
       ENDIF
C
      IF(LSTEP .EQ. 1) THEN
        WRITE(7,10) 'R E S U L T S'
 10     FORMAT(////T33,A/T33,I3,('-'))
      ENDIF

      WRITE(7,30) 'LOADSTEP',LSTEP,QI
 30   FORMAT(///T5,A,T14,':',T39,I4/
     $      T5,'LOADING PARAMETER:',T25,F20.4/)
        WRITE(7,35) 'DOF','DISPL','LAMBDA'
 35     FORMAT(T11,A,T28,A,T42,A/)
C
C--  READ THE DEGREES OF FREEDOM FOR WHICH THE DISPLACEMENTS ARE TO
BE
C    GIVEN AND TABULATE THE DISPLACEMENTS FOR THE GIVEN EQUILIBRIUM
C    POINT. (TOTAL DISPLACEMENTS)
C
     IF(LSTEP .EQ. 1)THEN
        READ(5,*) J
        WRITE (3) J
      ELSE
         READ (3) J
      ENDIF
C
C
 40   IF(J .NE. 0) THEN
         WRITE(7,50) J, D(J), QI
 50      FORMAT(T10,I3,T20,F15.8,T35,F15.8)
         IF(LSTEP .EQ. 1) THEN
            READ(5,*)  J
            WRITE (3) J
         ELSE
            READ (3) J
         ENDIF
         GO TO 40
      ENDIF
C
      LSTEP = LSTEP + 1
C
C
      RETURN
      END
```

Appendix B

Geometrically Nonlinear Plane Truss Finite Element Fortran Program Using the B-notation

```
C***********************************************************************
C           NONLINEAR ANALYSIS OF SPACE TRUSSES                *
C                                                  *
C     NEWTON-RAPHSON METHOD    AND    RIKS-WEMPNER METHOD          *
C***********************************************************************
C*                      INPUT DATA                         *
C***********************************************************************
C    LIST-DIRECTED INPUT:
C    INPUT UNITS: KIP, INCH, RADIAN, FAHRENHEIT
C
C    1. ENTER DATE IN THE FORM  01/14/91  (IN MAIN)
C         DATE
C
C    2. ENTER THE METHOD TO BE USED (IN MAIN)
C       NEWTON-RAPHSON: 1    RIKS-WEMPNER: 2
C         METHOD
C
C    3. ENTER NUMBER OF ELEMENTS AND NUMBER OF JOINTS (IN MAIN)
C         NE, NJ
C
C    4. ENTER MEMBER INCIDENCES (IN STRUCT)
C         MINC(1,I), MINC(2,I)  I=1,NE
C
C    5. ENTER FOR EACH JOINT CONSTRAINT (IN STRUCT)
C         JNUM, JDIR
C         AFTER LAST JOINT CONSTRAINT ENTER
C         0, 0
C
C    6. ENTER JOINT COORDINATES FOR EACH JOINT(IN PROP)
C         X(1,J), X(2,J), X(3,J)   J=1,NJ
C
C    7. ENTER MEMBER PROPERTIES CROSS SECTIONAL AREA
C         AND ELASTIC MODULUS FOR EACH MEMBER (IN PROP)
C         AREA(I), EMOD(I)    I=1,NE
C
C
C    8. IF THERE ARE JOINT LOADS ENTER (IN JLOAD)
C           JNUM, JDIR, FORCE
C           AFTER LAST JOINT LOAD ENTER
C           0, 0, 0
C         ELSE ENTER
C           0, 0, 0
C         END IF
C
C    9. ENTER THE MAXIMUM VALUE OF LAMBDA, INITIAL VALUE OF LAMBDA
C       AND THE INCREMENT IN LAMBDA   (IN MAIN)
C         QIMAX, QI, DQI
C
C   10. ENTER THE NUMBER OF ITERATIONS FOR UPDATING STIFFNESS MATRIX,
C       MAXIMUM AND DESIRED NUMBER OF ITERATIONS (IN MAIN)
C         ITENUM, ITEMAX, ITEDES
```

```
C
C   11. ENTER THE TOLERANCES IN DISPLACEMENT, AND FORCE (IN MAIN)      )
C         TOLDIS, TOLFOR
C
C   12. ENTER THE DOF FOR WHICH DISPLACEMENT AND LAMBDA VALUES ARE
TO
C       BE PRINTED AFTER EACH LOAD INCREMENT (IN RESULT)
C       END WITH 0
C
C**********************************************************************
C*                    MAIN PROGRAM                          *
C**********************************************************************
C
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DOUBLE PRECISION TIMEA, TIMEB, DTIME
      CHARACTER*(*) TITLE, UNITS, DATE*8, FNAME*12
      PARAMETER (LIM = 100000, TITLE = 'PLANE TRUSS ANALYSIS',
     $       UNITS = 'UNITS: KIP, INCH, RADIAN, FAHRENHEIT')
      DIMENSION A(LIM),IA(LIM)
C
C    RESERVE MEMORY; READ AND ECHO NE, NJ; SET POINTERS FOR DATA
C    ARRAYS; IF MEMORY IS ADEQUATE CALL STRUCT, ELSE SEND MESSAGE AND
C    STOP; SET POINTERS FOR SOLUTION ARRAYS; IF MEMORY IS ADEQUATE,
C    CALL LOAD. THEN CALL NEWRAP OR RIKWEM
C----------------------------------------
C    OPEN DATA FILES: ********* FOR PC ONLY*******
C    WRITE(6,*) 'INPUT DATA FILE: '
C    READ(*,'(A)') FNAME
C***  OPEN(5,FILE = FNAME)
C***  OPEN(7,FILE = 'RW.OUT', STATUS = 'UNKNOWN')
C----------------------------------------
C
C
      READ(5,'(A)') DATE
      WRITE(6,5) TITLE,'(EARLS, 1992)',
     $'DATE: ',DATE,UNITS
 5    FORMAT('1',T5,73('*')/T5,'*',T32,A,T77,'*'/T5,'*',T35,A,T77,'*'/
     $       T5,73('*')//T64,2(A)///T5,A)
      READ(5,*)METHOD
      READ(5,*) NE, NJ
      WRITE(6,15) 'CONTROL VARIABLES',
     $       'NUMBER OF ELEMENTS',NE,'NUMBER OF JOINTS',NJ
 15   FORMAT(///T5,A/2(T5,A,T30,I4/))
C
C--- SET POINTERS FOR DATA ARRAYS (IN STRUCT):
C
      NP=1
      NAREA=NP+3*NJ
      NEMOD=NAREA+NE
      NELENG=NEMOD+NE
      NC1=NELENG+NE
```

```
    NC2=NC1+NE
    NC3=NC2+NE
    NMCODE=NC3+NE
    NJCODE=NMCODE+6*NE
    NMINC=NJCODE+3*NJ
    NMAXA=NMINC+2*NE
C    TEMPORARILY LET NEQ=3*NJ TO SET POINTERS FOR NKHT AND MAX
    NEQ=3*NJ
    NKHT=NMAXA+(NEQ+1)
    MAX=NKHT+NEQ-1
C
C--- IF MEMORY IS ADEQUATE CALL STRUCT, ELSE SEND MESSAGE AND STOP.
C
    IF(MAX .LE. LIM) THEN
      CALL STRUCT(A(NP),A(NAREA),A(NEMOD),A(NELENG),
   $             A(NC1),A(NC2),A(NC3),IA(NMAXA),IA(NKHT),
   $             IA(NMCODE),IA(NJCODE),IA(NMINC),NE,NJ,NEQ,LSS)
C
C--- SET POINTERS FOR SOLUTION ARRAYS (IN LOAD AND NEWRAP OR RIKWEM)
C
C--- TEMPORARILY SET LSS=NEQ
C       LSS = NEQ
C
      NF=NKHT+NEQ
      NSS=NF+NEQ
      NQ=NSS+LSS
      NQT=NQ+NEQ
      NR=NQT+NEQ
      NFP=NR+NEQ
      ND=NFP+NEQ
      NDD=ND+NEQ
      NDDO=NDD+NEQ
      NDD01=NDDO+NEQ
      NDD1=NDD01+NEQ
      NDD2=NDD1+NEQ
      NDDP=NDD2+NEQ
      NFPI=NDDP+NEQ
      NDEFLN=NFPI+NEQ
      NELONG=NDEFLN+NE
      NTT=NELONG+NE
      MAX=NTT+NEQ-1
C
C--- IF MEMORY IS ADEQUATE CALL FOR EACH LOAD CONDITION LOAD,
C    NEWRAP OF RIKWEM.
C
      IF(MAX .LE. LIM) THEN
        CALL LOAD(A(NQ),IA(NJCODE),NEQ)
C
C--- READ QIMAX, QI ,DQI
        READ(5,*)QIMAX,QI,DQI
C
```

```
      WRITE(6,25) 'QIMAX = ', QIMAX,'  QI = ',QI,'  DQI = ',DQI
 25    FORMAT(/3(T10,A,F12.6/))
C
C--- READ ITEUPD,ITEMAX,ITEDES
      READ(5,*)ITENUM, ITEMAX, ITEDES
C
      WRITE(6,*)'NUMBER OF ITERATIONS FOR UPDATING STIFFNESS',
   $  ' MATRIX = ',ITEUPD
      WRITE(6,*)'MAXIMUM NUMBER OF ITERATIONS IN EACH LOAD S',
   $  'TEP    = ',ITEMAX
      WRITE(6,*)'DESIRED NUMBER OF ITERATIONS IN EACH LOAD S',
   $  'TEP    = ',ITEDES
C
C--- READ TOLDIS,TOLFOR
      READ(5,*)TOLDIS,TOLFOR
C
      WRITE(6,35) 'TOLERANCE IN DISPLACEMENT  = ',  TOLDIS,
   $          'TOLERANCE IN FORCE         = ',  TOLFOR
 35    FORMAT(/2(T10,A,T45,F12.6/))
C
      LSTEP = 1
C
C
C--- INSERT CALL TO INITIALIZE SYSTEM TIMER HERE (FOR VM1)
      CALL CPUTIME(TIMEA, IRCD_A)
C
      IF(METHOD .EQ. 1)THEN
        WRITE(6,40) 'ITERATIVE METHOD: NEWTON-RAPHSON '
 40      FORMAT(T10,A)
C
       CALL NEWRAP(A(ND),A(NDD),A(NQ),A(NQT),A(NTT),A(NSS),
   $          A(NAREA),A(NEMOD),A(NELENG),A(NC1),
   $          A(NC2),A(NC3),A(NELONG),A(NDEFLN),A(NF),
   $          A(NFP),A(NR),A(NP),A(NDDO),A(NFPI),
   $          IA(NMAXA),IA(NMCODE),IA(NJCODE),IA(NMINC),
   $          INCONV,ITECNT,ITENUM,ITEUPD,ITEMAX,NE,NEQ,NJ,
   $          LSS,QIMAX,QI,DQI,TOLDIS,TOLFOR,LSTEP,NFE)
C
      ELSEIF(METHOD .EQ. 2)THEN
       WRITE(6,50) 'ITERATIVE METHOD: MODIFIED RIKS/WEMPNER '
 50     FORMAT(T10,A)
C
       CALL RIKWEM(A(ND),A(NDD),A(NDDO),A(NDD01),A(NDD1),A(NDD2),
   $          A(NDDP),A(NAREA),A(NEMOD),A(NELENG),
   $          A(NC1),A(NC2),A(NC3),A(NQ),A(NQT),
   $          A(NSS),A(NTT),A(NP),A(NELONG),A(NDEFLN),
   $          A(NF),A(NFP),A(NFPI),A(NR),
   $          IA(NJCODE),IA(NMAXA),IA(NMCODE),IA(NMINC),
   $          IT,INCONV,ITECNT,ITENUM,ITEMAX,ITEUPD,ITEDES,
   $          NE,NEQ,NJ,LSS,QIMAX,QI,DQI,
   $          TOLDIS,TOLFOR,LSTEP,NFE)
```

```
C
          ENDIF
C
        ELSE
          WRITE(6,'(T10,A/)') 'ERROR MESSAGE: INCREASE MEMORY'
        END IF
      ELSE
        WRITE(6,'(T10,A/)') 'ERROR MESSAGE: INCREASE MEMORY'
      ENDIF
C
C
C INSERT CALL TO RETURN EXECUTION TIME IN MICROSECONDS IN DTIME.
C
       CALL CPUTIME(TIMEB, IRCD_B)
       IF(IRCD_A .NE. 8 .AND. IRCD_B .EQ. 0)THEN
           DTIME = (TIMEB - TIMEA)
       ENDIF
C
        WRITE(7,*) ' CPUTIME (MICROSECONDS)  = ', DTIME
        WRITE(7,*) '                    NFE   = ', NFE
C
      STOP
      END
C***********************************************************************
C*                        STRUCT                           *
C***********************************************************************
      SUBROUTINE STRUCT(X,AREA,EMOD,ELENG,C1,C2,C3,
     $             MAXA,KHT,MCODE,JCODE,MINC,NE,NJ,NEQ,LSS)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION X(3,*),AREA(*),EMOD(*),ELENG(*),C1(*),C2(*)
     $       ,C3(*),MAXA(*),KHT(*),MCODE(6,*),JCODE(3,*),MINC(2,*)
C
C    READ AND ECHO THE MEMBER INCIDENCES, MINC(L,I); INITIALIZE THE
C    ELEMENTS OF THE JOINT CODE MATRIX, JCODE, TO UNITY, READ AND
C    ECHO FOR EACH JOINT CONSTRAINT THE JOINT NUMBER, JNUM, AND
C    JOINT DIRECTION, JDIR, AND STORE A ZERO IN THE CORRESPONDING
C    LOCATION OF JCODE (END OF DATA MARKER JNUM=0); CALL CODES,
C    SKYLIN, AND PROP.
C
      WRITE(6,10) 'MEMBER INCIDENCES','MEMBER','A-END','B-END'
   10 FORMAT(///T10,A//T10,A,T17,A,T23,A/)
      DO 30 I=1,NE
         READ(5,*) MINC(1,I),MINC(2,I)
         WRITE(6,20) I,MINC(1,I),MINC(2,I)
   20    FORMAT(T11,I3,T18,I3,T24,I3)
   30 CONTINUE
C
      DO 60 J=1,NJ
         DO 50 L=1,2
            JCODE(L,J)=1
   50    CONTINUE
   60 CONTINUE
 C
     WRITE(6,70) 'JOINT CONSTRAINTS','JOINT','DIRECTION'
   70 FORMAT(///T10,A//T10,A,T17,A/)
     READ(5,*) JNUM,JDIR
   80 IF (JNUM.NE.0) THEN
        WRITE(6,90) JNUM,JDIR
   90    FORMAT(T9,I3,T19,I3)
        JCODE(JDIR,JNUM)=0
        READ(5,*) JNUM,JDIR
        GO TO 80
     END IF
 C
     CALL CODES(MCODE,JCODE,MINC,NE,NJ,NEQ)
 C
     CALL SKYLIN(KHT,MAXA,MCODE,NE,NEQ,LSS)
 C
     CALL PROP(X,AREA,EMOD,ELENG,C1,C2,C3,MINC,NE,NJ)
 C
     RETURN
     END
 C**********************************************************************
 C*                          CODES                               *
 C**********************************************************************
     SUBROUTINE CODES(MCODE,JCODE,MINC,NE,NJ,NEQ)
     IMPLICIT DOUBLE PRECISION (A-H, O-Z)
     DIMENSION MCODE(6,*),JCODE(3,*),MINC(2,*)
 C
 C
 C    GENERATE JOINT CODE, JCODE, BY ASSIGNING INTEGERS IN SEQUENCE,
 C    BY COLUMNS, TO ALL NONZERO ELEMENTS OF JCODE FROM 1 T0 NEQ;
 C       GENERATE THE MEMBER CODE, MCODE, BY TRANSFERRING VIA MINC
 COLUMNS
 C    OF JCODE INTO COLUMNS OF MCODE.
 C
 C    GENERATE JCODE
 C
     NEQ=0
     DO 20 J=1,NJ
       DO 10 L=1,2
         IF(JCODE(L,J).NE.0) THEN
           NEQ=NEQ+1
           JCODE(L,J)=NEQ
         END IF
  10   CONTINUE
  20 CONTINUE
 C
     WRITE(6,*) 'NEQ = ', NEQ
 C    GENERATE MCODE
 C
     DO 40 I=1,NE
```

```
      J=MINC(1,I)
      K=MINC(2,I)
      DO 30 L=1,2
        MCODE(L,I)=JCODE(L,J)
        MCODE(L+2,I)=JCODE(L,K)
 30   CONTINUE
 40 CONTINUE
C
C    WRITE(6,50)'MCODE(TRANSPOSED)'
C 50 FORMAT(/T10,A/)
C    DO 60 I=1,NE
C       PRINT*,(MCODE(L,I),L=1,4)
C 60 CONTINUE
C
C
     RETURN
     END
C**********************************************************************
C*                      SKYLIN                               *
C**********************************************************************
C
C     SKYLIN DETERMINES KHT USING MCODE, AND DETERMINES MAXA FROM
KHT.
*
*   I HAVE MODIFIED SKYLIN SUCH THAT DOF IN EACH COLUMN OF MCODE
*     NEED NOT BE ARRANGED IN INCREASING ORDER FROM TOP TO BOTTOM;
I.E.,
*        ELEMENTS NEED NOT BE DIRECTED FROM A LOWER TO A HIGHER
NUMBERED
*   JOINT(HOLZER 2/8/91).
C
     SUBROUTINE SKYLIN(KHT,MAXA,MCODE,NE,NEQ,LSS)
     IMPLICIT DOUBLE PRECISION (A-H, O-Z)
     DIMENSION KHT(*),MAXA(*),MCODE(6,*)
C
C
     DO 10 I=1,NEQ
       KHT(I)=0
 10 CONTINUE
C
C    GENERATE KHT
C
     DO 30 I=1,NE
       MIN=NEQ
       DO 15 L=1,4
         IF (MCODE(L,I) .GT. 0 .AND. MCODE(L,I) .LT. MIN) THEN
           MIN=MCODE(L,I)
         END IF
 15    CONTINUE
C
       DO 20 L=1,4
```

```
         K=MCODE(L,I)
         IF(K.NE.0) THEN
            KHT(K)=MAX0(KHT(K),(K-MIN))
         ENDIF
   20    CONTINUE
   30 CONTINUE
*      WRITE(6,100)
*  100 FORMAT(//11X,'I',10X,'KHT(I)',10X,'MAXA(I)')
C
C     GENERATE MAXA
C
      MAXA(1)=1
      DO 40 I=1,NEQ
*        WRITE(6,200) I,KHT(I),MAXA(I)
*  200   FORMAT(7X,I5,9X,I5,11X,I5)
         MAXA(I+1)=MAXA(I)+KHT(I)+1
   40 CONTINUE
      LSS=MAXA(NEQ+1)-1
      I=NEQ+1
*      WRITE(6,300) I,MAXA(I),LSS
*  300 FORMAT(7X,I5,25X,I5//7X,'LSS = ',I5)
C
       WRITE(6,*)' LSS = ', LSS
C
      RETURN
      END
C**********************************************************************
C*                         PROP                            *
C**********************************************************************
      SUBROUTINE PROP(X,AREA,EMOD,ELENG,C1,C2,C3,MINC,NE,NJ)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION X(3,*),AREA(*),EMOD(*),ELENG(*),C1(*),C2(*)
     $        ,C3(*),MINC(2,*)
C
C
C     READ AND ECHO JOINT COORDINATES, X(L,J); COMPUTE FOR EACH
C     ELEMENT BY THE LENGTH, ELENG(I), AND THE DIRECTION COSINES
C     C1(I), C2(I) & C3(I); READ FOR EACH ELEMENT THE CROSS SECTIONAL
C     AREA, AREA(I), THE MODULUS OF ELASTICITY, EMOD(I)
C     PRINT ELEMENT PROPERTIES.
C
      WRITE(6,10) 'JOINT COORDINATES','JOINT','DIRECTION-1',
     $          'DIRECTION-2','DIRECTION-3'
   10 FORMAT(///T10,A//T10,A,T17,A,T30,A,T43,A)
      DO 30 J=1,NJ
         READ(5,*) X(1,J),X(2,J),X(3,J)
         WRITE(6,20) J,X(1,J),X(2,J),X(3,J)
   20    FORMAT(T9,I3,T18,F8.2,T31,F8.2,T44,F8.2)
   30 CONTINUE
C
       WRITE(6,40) 'ELEMENT PROPERTIES','ELASTIC',
     $      'ELEMENT','AREA','MODULUS','LENGTH'
   40 FORMAT(///T10,A//T39,A/T10,A,T19,A,T39,A,T62,A)
C
      DO 60 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)
         EL3=X(3,K)-X(3,J)
         ELENG(I)=DSQRT(EL1**2+EL2**2+EL3**2)
         C1(I)=EL1/ELENG(I)
         C2(I)=EL2/ELENG(I)
         C3(I)=EL3/ELENG(I)
C
         READ(5,*) AREA(I),EMOD(I)
         WRITE(6,50) I,AREA(I),EMOD(I),ELENG(I)
         WRITE(7,*) I, ELENG(I)
   50    FORMAT(T12,I3,T17,E9.3,T39,E9.3,T60,F8.3)
   60 CONTINUE
C
      RETURN
      END
C***********************************************************************
C*                          LOAD                               *
C***********************************************************************
      SUBROUTINE LOAD(Q,JCODE,NEQ)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION Q(*),JCODE(3,*)
C
C     INITIALIZE TO ZERO THE JOINT LOAD VECTOR, Q.
C     CALL JLOAD.
C
C
      DO 20 K=1,NEQ
         Q(K)=0.0
   20 CONTINUE
C
      CALL JLOAD(Q,JCODE)
C
      RETURN
      END
C***********************************************************************
C*                          JLOAD                              *
C***********************************************************************
      SUBROUTINE JLOAD(Q,JCODE)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION Q(*),JCODE(3,*)
C
C     READ THE JOINT NUMBER, JNUM, THE JOINT DIRECTION, JDIR, AND THE
C     APPLIED FORCE, FORCE; WHILE JNUM IS NOT EQUAL TO ZERO, PRINT JNUM,
C     JDIR, FORCE,STORE FORCE IN Q, AND READ JNUM, JDIR, FORCE.
```

```
C
      READ(5,*) JNUM,JDIR,FORCE
      IF(JNUM.NE.0) THEN
*        WRITE(6,10) 'JOINT LOADS','GLOBAL','JOINT','DIRECTION','FORCE'
*  10    FORMAT(///T10,A/T10,11('-')/T18,A/T10,A,T17,A,T35,A)
   20   IF (JNUM.NE.0) THEN
*          WRITE(6,30) JNUM,JDIR,FORCE
*  30      FORMAT(T11,I3,T21,I1,T28,F15.8)
          K=JCODE(JDIR,JNUM)
          Q(K)=FORCE
          READ(5,*) JNUM,JDIR,FORCE
          GO TO 20
        END IF
      ELSE
        WRITE(6,40) 'NO JOINT FORCES'
   40   FORMAT(///T10,A)
      END IF
C
      RETURN
      END
C**********************************************************************
C*                        NEWRAP                          *
C**********************************************************************
      SUBROUTINE NEWRAP(D,DD,Q,QT,TT,SS,AREA,EMOD,ELENG,C1,C2,C3,
     $            ELONG,DEFLEN,F,FP,R,X,DDO,FPI,
     $            MAXA,MCODE,JCODE,MINC,
     $            INCONV,ITECNT,ITENUM,ITEUPD,ITEMAX,NE,NEQ,NJ,
     $            LSS,QIMAX,QI,DQI,TOLDIS,TOLFOR,LSTEP,NFE)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION D(*),DD(*),Q(*),QT(*),TT(*),SS(*),AREA(*),EMOD(*),
     $       ELENG(*),C1(*),C2(*),C3(*),ELONG(*),DEFLEN(*),F(*),
     $       FP(*),R(*),X(3,*),DDO(*),FPI(*),
     $       MAXA(*),MCODE(6,*),JCODE(3,*),MINC(2,*),C1S(40),C2S(40)
C
      DO 5 I = 1,NE
        C1S(I)=C1(I)
        C2S(I)=C2(I)
 5    CONTINUE
C
      DO 10 I = 1, NEQ
        D(I) = 0.D0
        DD(I) = 0.D0
        F(I) = 0.D0
        FP(I) = 0.D0
        FPI(I) = 0.D0
 10   CONTINUE
C
      DO 15 I = 1, NE
       DEFLEN(I) = ELENG(I)
       ELONG(I)  = 0.D0
 15   CONTINUE
```

```
C
 30   IF (QI .LE. QIMAX) THEN
C
        DO 40 I = 1, NEQ
          QT(I) = Q(I) * QI
 40      CONTINUE
C
        CALL NRITER(D,DD,Q,QT,TT,SS,AREA,EMOD,ELENG,C1,C2,C3,
     $          ELONG,DEFLEN,F,FP,R,X,DDO,FPI,
     $          MAXA,MCODE,JCODE,MINC,
     $          INCONV,ITECNT,ITENUM,ITEUPD,ITEMAX,NE,NEQ,NJ,
     $          LSS,QIMAX,QI,DQI,TOLDIS,TOLFOR,NFE,C1S,C2S)
C
        DO 60 I = 1, NEQ
          FP(I) = F(I)
 60      CONTINUE
C
        IF (INCONV .NE. 0) THEN
          WRITE(6,*)'*****ERROR *****SOLUTION FAILS TO CONVERGE IN',
     $          ' GIVEN NUMBER OF ITERATIONS'
          STOP
        ELSE
C           CALL RESULT(D,MCODE,JCODE,MINC,NE,NJ,NEQ,LSTEP,QI)
        WRITE(7,*) D(6),',',QI
C        WRITE(*,*) D(4),',',QI
        ENDIF
C
        QI = QI + DQI
C
        GO TO 30
      ENDIF
C
      RETURN
      END
C***********************************************************************
C*                    NRITER                                 *
C***********************************************************************
      SUBROUTINE NRITER(D,DD,Q,QT,TT,SS,AREA,EMOD,ELENG,C1,C2,C3,
     $          ELONG,DEFLEN,F,FP,R,X,DDO,FPI,
     $          MAXA,MCODE,JCODE,MINC,
     $          INCONV,ITECNT,ITENUM,ITEUPD,ITEMAX,NE,NEQ,NJ,
     $          LSS,QIMAX,QI,DQI,TOLDIS,TOLFOR,NFE,C1S,C2S)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION D(*),DD(*),Q(*),QT(*),TT(*),SS(*),AREA(*),EMOD(*),
     $     ELENG(*),C1(*),C2(*),C3(*),ELONG(*),DEFLEN(*),F(*),
     $     FP(*),R(*),X(3,*),DDO(*),FPI(*),
     $     MAXA(*),MCODE(6,*),JCODE(3,*),MINC(2,*),C1S(40),C2S(40)
C
C
      CALL FORCES(F,QT,AREA,EMOD,ELENG,C1,C2,C3,ELONG,R,MCODE,MINC,
     $          NE,NEQ,D,C1S,C2S)
```

```
C
      ITENUM = ITEUPD
      ITECNT = 1
      INCONV = 1
C
 10  IF (INCONV .NE. 0 .AND. ITECNT .LE. ITEMAX) THEN
C
      DO 15 I = 1, NEQ
         R(I) = QT(I) - F(I)
C      WRITE(*,*)'RES = ',R(I)
 15    CONTINUE
C      WRITE(*,*)' '
C
      IF (ITENUM .GE. ITEUPD) THEN
        CALL STIFF(SS,AREA,EMOD,ELENG,C1,C2,C3,ELONG,DEFLEN,
     $             MAXA,MCODE,NE,LSS)
        ITENUM = 0
      ENDIF
C
      DO 20 I = 1, NEQ
         TT(I) = 0.D0
         TT(I) = R(I)
 20    CONTINUE
C
      CALL SOLVE(SS,TT,MAXA,NEQ,1)
C      WRITE(*,*)'----*',TT(1)
C
      DO 40 I = 1, NEQ
         DD(I) = TT(I)
         TT(I) = 0.D0
 40    CONTINUE
C
      IF (ITECNT .EQ. 1) THEN
       DO 50 I = 1, NEQ
         DDO(I) = DD(I)
 50      CONTINUE
      ENDIF
C
      DO 60 I = 1, NEQ
         D(I) = D(I) + DD(I)
C      WRITE(*,*)'DISPL*********',D(I)
 60    CONTINUE
C      WRITE(*,*)' '
C
       CALL UPDATC(X,DD,C1,C2,C3,ELONG,ELENG,DEFLEN,MINC,JCODE,
     $            NE,NJ,NEQ)
C
       CALL FORCES(F,QT,AREA,EMOD,ELENG,C1,C2,C3,ELONG,R,MCODE,MINC,
     $            NE,NEQ,D,C1S,C2S)
C
       CALL TEST(D,DD,DDO,F,FP,FPI,QT,INCONV,NEQ,
     $           TOLFOR,TOLDIS)
C
        DO 80 I = 1, NEQ
           FPI(I) = F(I)
 80     CONTINUE
C
        ITECNT = ITECNT + 1
        ITENUM = ITENUM + 1
C
        GO TO 10
      ENDIF
C
      RETURN
      END
C************************************************************************
C*                         RIKWEM                          *
C************************************************************************
      SUBROUTINE RIKWEM(D,DD,DDO,DD01,DD1,DD2,DDP,AREA,EMOD,ELENG,
     $           C1,C2,C3,Q,QT,SS,TT,X,ELONG,DEFLEN,F,FP,FPI,R,
     $           JCODE,MAXA,MCODE,MINC,
     $           IT,INCONV,ITECNT,ITENUM,ITEMAX,ITEUPD,ITEDES,
     $           NE,NEQ,NJ,LSS,QIMAX,QI,DQI,
     $           TOLDIS,TOLFOR,LSTEP,NFE)
C
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
C
      DIMENSION D(*),DD(*),DDO(*),DD01(*),DD1(*),DD2(*),DDP(*),AREA(*),
     $        EMOD(*),ELENG(*),C1(*),C2(*),C3(*),Q(*),QT(*),
     $        SS(*),TT(*),X(3,*),ELONG(*),DEFLEN(*),
     $        F(*),FP(*),FPI(*),R(*),
     $        JCODE(3,*),MAXA(*),MCODE(6,*),MINC(2,*)
C
C--- INITIALIZE THE VARIABLES D,DD,F,FP,FPI TO ZERO
C
      DO 10 I = 1, NEQ
        D(I) = 0.D0
        DD(I) = 0.D0
        F(I) = 0.D0
        FP(I) = 0.D0
        FPI(I) = 0.D0
 10   CONTINUE
C
C--- INITIALIZE THE VARIABLES DEFLEN, ELONG
C
      DO 15 I = 1, NE
       DEFLEN(I) = ELENG(I)
       ELONG(I)  = 0.D0
 15   CONTINUE
C
      NFE = 0
      ITECNT = 1
```

```
C
 20  IF (QI .LE. QIMAX .AND. ITECNT .LE. 50)THEN
C      WRITE(6,'(T10,A,3X,F16.10)') 'QI = ',QI
C
C--- COMPUTE THE TANGENT STIFFNESS MATRIX
C
      CALL STIFF(SS,AREA,EMOD,ELENG,C1,C2,C3,ELONG,DEFLEN,
     $            MAXA,MCODE,NE,LSS)
C
C--- COMPUTE THE FIRST TRIAL SOLUTION
C
      DO 40 I = 1, NEQ
         TT(I) = 0.D0
         TT(I) = Q(I)
 40    CONTINUE
C
      CALL SOLVE(SS,TT,MAXA,NEQ,1)
C
      DO 50 I = 1, NEQ
         DD01(I) = TT(I)
         TT(I) = 0.D0
 50    CONTINUE
C
C--- COMPUTE THE ARC LENGTH FOR THE FIRST ITERATION
C    FOR SUBSEQUENT ITERATIONS COMPUTE THE LOAD INCREMENT
C
      IF(ITECNT .EQ. 1)THEN
         DS=DQI*DSQRT(DOTPRD(DD01,DD01,NEQ)+1.D0)
         DSMAX=DS*1.0D0
      ELSE
         DQI=DS/DSQRT(DOTPRD(DD01,DD01,NEQ)+1.D0)
         TEMP=DQI*(DOTPRD(DD01,DDP,NEQ)+DQII)
         IF(TEMP .GT. 0)THEN
           SGN=1
         ELSE
           SGN=-1
         ENDIF
         DQI = SGN*DQI
      ENDIF
C
      DQII = 0.0
      DQI1 = DQI
      DQII = DQII + DQI
C
C--- COMPUTE THE INCREMENT IN DISPLACEMENT AND TOTAL DISPLACEMENT
C
      DO 60 I = 1, NEQ
        DDO(I) = DQI*DD01(I)
        D(I) = D(I) + DDO(I)
        DDP(I) = DDO(I)
 60    CONTINUE
```

```
C
C--- INCREMENT THE LOAD PARAMETER
C
          QI = QI + DQI
C      WRITE(6,*) ' QI ',QI
C--- UPDATE THE COORDINATES
        CALL UPDATC(X,DDO,C1,C2,C3,ELONG,ELENG,DEFLEN,MINC,JCODE,
     $             NE,NJ,NEQ)
C
        CALL FORCES(F,QT,AREA,EMOD,ELENG,C1,C2,C3,ELONG,R,MCODE,MINC,
     $             NE,NEQ)
C
C
C--- CALL SUBROUTINE TRLVCT
C
        CALL TRLVCT(D,DD,DDO,DD1,DD2,DDP,AREA,EMOD,ELENG,
     $        C1,C2,C3,Q,QT,SS,TT,X,ELONG,DEFLEN,F,FP,FPI,R,
     $        JCODE,MAXA,MCODE,MINC,
     $        IT,INCONV,ITECNT,ITENUM,ITEMAX,ITEUPD,NE,NEQ,
     $        NJ,LSS,QIMAX,QI,DQI,TOLDIS,TOLFOR,DQI1,DQII,NFE)
C
C
        DO 80 I = 1, NEQ
            FP(I) = F(I)
 80      CONTINUE
C
C
        IF(INCONV .NE. 0)THEN
          WRITE(6,*) '**** ERROR **** SOLUTION FAILS TO CONVERGE',
     $    ' IN GIVEN NUMBER OF ITERATIONS'
          STOP
        ELSE
          CALL RESULT(D,MCODE,JCODE,MINC,NE,NJ,NEQ,LSTEP,QI)
C
        ENDIF
C
          ITECNT = ITECNT + 1
C
C---COMPUTE THE SCALED ARC LENGTH
C
         DS=DS*DSQRT((1.D0*ITEDES)/(1.D0*IT))
         IF(DABS(DS) .GT. DSMAX)THEN
             IF(DS .LT. 0)THEN
               DS = -DSMAX
             ELSE
               DS = DSMAX
             ENDIF
         ENDIF
C
         GO TO 20
     ENDIF
```

```
C
        RETURN
        END
C**********************************************************************
C*                      TRLVCT                         *
C**********************************************************************
      SUBROUTINE TRLVCT(D,DD,DDO,DD1,DD2,DDP,AREA,EMOD,ELENG,
     $             C1,C2,C3,Q,QT,SS,TT,X,ELONG,DEFLEN,F,FP,FPI,R,
     $             JCODE,MAXA,MCODE,MINC,
     $             IT,INCONV,ITECNT,ITENUM,ITEMAX,ITEUPD,NE,NEQ,
     $             NJ,LSS,QIMAX,QI,DQI,TOLDIS,TOLFOR,
     $             DQI1,DQII,NFE)
C
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
C
      DIMENSION D(*),DD(*),DDO(*),DD1(*),DD2(*),DDP(*),AREA(*),
     $        EMOD(*),ELENG(*),C1(*),C2(*),C3(*),Q(*),QT(*),
     $        SS(*),TT(*),X(3,*),ELONG(*),DEFLEN(*),
     $        F(*),FP(*),FPI(*),R(*),
     $        JCODE(3,*),MAXA(*),MCODE(6,*),MINC(2,*)
C
C
      ITENUM = ITEUPD
      IT = 1
      INCONV = 1
C
 10   IF(INCONV .NE. 0 .AND. IT .LE. ITEMAX)THEN
C
C        WRITE(6,'(T8,A,3X,I3)') 'ITERATION ', IT
C
         DO 15 I = 1, NEQ
            FPI(I) = F(I)
 15      CONTINUE
C--- UPDATE THE LOAD VECTOR
        DO 20 I = 1, NEQ
          QT(I) = Q(I) * QI
 20      CONTINUE
C
C--- COMPUTE  UNBALANCED FORCES
C
        DO 25 I = 1, NEQ
          R(I) = QT(I) - F(I)
 25      CONTINUE
C
C--- COMPUTE THE TANGENT STIFFNESS MATRIX
        IF (ITENUM .GE. ITEUPD) THEN
          CALL STIFF(SS,AREA,EMOD,ELENG,C1,C2,C3,ELONG,DEFLEN,
     $            MAXA,MCODE,NE,LSS)
          ITENUM = 0
        ENDIF
C
```

```
C--- SOLVE FOR DD1
      DO 30 I = 1, NEQ
         TT(I) = 0.D0
         TT(I) = Q(I)
 30    CONTINUE
C
      CALL SOLVE(SS,TT,MAXA,NEQ,1)
C
      DO 40 I = 1, NEQ
         DD1(I) = TT(I)
         TT(I) = 0.D0
 40    CONTINUE
C
C--- SOLVE FOR DD2
      DO 50 I = 1, NEQ
         TT(I) = 0.D0
         TT(I) = R(I)
 50    CONTINUE
C
      CALL SOLVE(SS,TT,MAXA,NEQ,2)
C
      DO 60 I = 1, NEQ
         DD2(I) = TT(I)
         TT(I) = 0.D0
 60    CONTINUE
C
C--- COMPUTE THE INCREMENT IN LOAD PARAMETER
      DQI = -(DOTPRD(DDO,DD2,NEQ))/(DOTPRD(DDO,DD1,NEQ)+DQI1)
      DQII = DQII + DQI
      QI = QI + DQI
C
      DO 70 I = 1, NEQ
        DD(I) = DQI*DD1(I) + DD2(I)
        D(I) = D(I) + DD(I)
        DDP(I) = DDP(I) + DD(I)
 70    CONTINUE
C
C        WRITE(6,'(T10,A,3X,F20.15)') 'QI = ',QI
C        WRITE(6,'(T10,A,2(5X,F20.15))') 'D1, D2', D(1),  D(2)
C
C--- UPDATE THE JOINT COORDINATES
       CALL UPDATC(X,DD,C1,C2,C3,ELONG,ELENG,DEFLEN,MINC,JCODE,
   $             NE,NJ,NEQ)
C
C--- COMPUTE THE INTERNAL FORCES
       CALL FORCES(F,QT,AREA,EMOD,ELENG,C1,C2,C3,ELONG,R,MCODE,MINC,
   $             NE,NEQ)
C
       DO 80 I = 1, NEQ
C         WRITE(6,*) 'FORCE  =  ' , F(I)
 80      CONTINUE
```

```
C
C--- CHECK FOR CONVERGENCE
         CALL TEST(D,DD,DDO,F,FP,FPI,QT,INCONV,NEQ,
     $            TOLFOR,TOLDIS)
C
         IT = IT + 1
         ITENUM = ITENUM + 1
C
         GO TO 10
      ENDIF
        NFE = NFE+(IT-1)
        WRITE(6,*) ' NFE = ', NFE
C
      RETURN
      END
C**********************************************************************
C*                          DOTPRD                          *
C**********************************************************************
      FUNCTION DOTPRD(DOT1,DOT2,N)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION DOT1(*),DOT2(*)
C
C--- DOTPRD COMPUTES THE DOT PRODUCT OF DOT1 AND DOT2.
C
      DOTPRD=0.D00
      DO 10 I=1,N
        DOTPRD=DOTPRD+DOT1(I)*DOT2(I)
 10   CONTINUE
C
      RETURN
      END
C**********************************************************************
C*                          STIFF                          *
C**********************************************************************
      SUBROUTINE STIFF(SS,AREA,EMOD,ELENG,C1,C2,C3,ELONG,DEFLEN,
     $             MAXA,MCODE,NE,LSS)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION SS(*),AREA(*),EMOD(*),ELENG(*),C1(*),C2(*),C3(*),
     $        ELONG(*),DEFLEN(*),G(6),H(6),MAXA(*),MCODE(6,*)
C
C   INITIALIZE THE SYSTEM STIFFNESS MATRIX, SS, TO ZERO; FOR
C   EACH ELEMENT CALL ELEMS AND ASSEMS.
C
      DO 10 L = 1,LSS
        SS(L) = 0.D0
   10 CONTINUE
C
      DO 20 N = 1,NE
        CALL ELEMS(AREA,EMOD,ELENG,C1,C2,C3,G,N)
        CALL NELEMS(AREA,EMOD,ELENG,C1,C2,C3,H,ELONG,DEFLEN,N)
        CALL ASSEMS(SS,G,H,MCODE,MAXA,N,LSS)
```

```
  20 CONTINUE
C
     DO 30 I = 1, LSS
C       WRITE(6,*) 'SS = ',SS(I)
  30 CONTINUE
C
     RETURN
     END
C***********************************************************************
C*                          ELEMS                         *
C***********************************************************************
     SUBROUTINE ELEMS(AREA,EMOD,ELENG,C1,C2,C3,G,N)
     IMPLICIT DOUBLE PRECISION (A-H, O-Z)
     DIMENSION AREA(*),EMOD(*),ELENG(*),C1(*),C2(*),C3(*),G(6)
C
C     FOR ELEMENT N, COMPUTE THE GLOBAL STIFFNESS COEFFICIENTS, G(6),
C     DEFINED IN EQS. 5.15 (HOLZER)
C
     GAMMA = AREA(N)*EMOD(N)/ELENG(N)
C
     G(1) = (GAMMA)*(C1(N)**2)
     G(2) = (GAMMA)*(C2(N)**2)
     G(3) =  GAMMA*C1(N)*C2(N)
C
     RETURN
     END
C***********************************************************************
C*                          NELEMS                        *
C***********************************************************************
     SUBROUTINE NELEMS(AREA,EMOD,ELENG,C1,C2,C3,H,ELONG,DEFLEN,N)
     IMPLICIT DOUBLE PRECISION (A-H, O-Z)
     DIMENSION AREA(*),EMOD(*),ELENG(*),C1(*),C2(*),C3(*),H(6),
    $        ELONG(*),DEFLEN(*)
C
C         FOR ELEMENT N, COMPUTE THE NON-LINEAR GLOBAL STIFFNESS
COEFFICIENTS,
C    H(6)
C
C
     GAMMA = AREA(N)*EMOD(N)/ELENG(N)
     P = ELONG(N)/DEFLEN(N)
C
     H(1) = GAMMA*P*(1-(C1(N)*C1(N)))
     H(2) = GAMMA*P*(1-(C2(N)*C2(N)))
     H(3) = -GAMMA*C1(N)*C2(N)*P
C
     RETURN
     END
C***********************************************************************
C*                          ASSEMS                        *
C***********************************************************************
```

```
      SUBROUTINE ASSEMS(SS,G,H,MCODE,MAXA,N,LSS)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION SS(*),G(*),H(*),MCODE(6,*),MAXA(*),INDEX(4,4)
      DATA INDEX/1,3,-1,-3, 3,2,-3,-2, -1,-3,1,3, -3,-2,3,2/
C
C
*     MODIFIED TO ALLOW DOF IN ANY COLUMN OF MCODE TO BE IN
*     ANY ORDER; SEE SKYLIN
*
C     INITIALIZE INDEX. ; ASSIGN STIFFNESS COEFFICIENTS, G(L),
C     OF ELEMENT N TO THE SYSTEM STIFFNESS MATRIX, SS, BY INDEX, MCODE,
C     AND MAXA.
C
      DO 20 JE = 1, 4
        J = MCODE(JE,N)
        IF (J .NE. 0) THEN
          DO 10 IE = 1, JE
            I = MCODE(IE,N)
            IF (I .NE. 0) THEN
              IF (I .GT. J) THEN
                K = MAXA(I) + I - J
              ELSE
                K = MAXA(J) + J - I
              END IF
              L = INDEX(IE,JE)
              IF (L .GT. 0) THEN
                SS(K) = SS(K) + G(L) + H(L)
              ELSE
                SS(K) = SS(K) - G(-L) - H(-L)
              END IF
            END IF
   10     CONTINUE
        END IF
   20 CONTINUE
C
      RETURN
      END
C***********************************************************************
C*                        SOLVE                                        *
C***********************************************************************
      SUBROUTINE SOLVE(SS,Q,MAXA,NEQ,LC)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION SS(*),Q(*),MAXA(*)
C
C        SOLVE DETERMINES THE SOLUTION TO THE SYSTEM EQUATIONS BY
COMPACT
C     GAUSSIAN ELIMINATION (HOLZER, PP. 290, 296, 307) BASED ON THE
C     SUBROUTINE COLSOL (BATHE P. 721) AND THE MODIFICATION BY MICHAEL
C     BUTLER (MS 1984): IF LC = 1, CALL FACTOR,FORSUB,AND BACSUB
C     IF LC > 1, CALL FORSUB AND BACSUB.
C
```

```
C
      IF (LC.EQ.1) THEN
         CALL FACTOR(SS,MAXA,NEQ)
      END IF
         CALL FORSUB(SS,Q,MAXA,NEQ)
         CALL BACSUB(SS,Q,MAXA,NEQ)
C
      RETURN
      END
C**********************************************************************
C*                         FACTOR                            *
C**********************************************************************
C    FACTOR PERFORMS THE LDU FACTORIZATION OF THE STIFFNESS MATRIX.
C
      SUBROUTINE FACTOR(SS,MAXA,NEQ)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION SS(*),MAXA(*)
C
      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=MIN0(IC,ND)
        C=0.00
        DO 30 L=1,KK
   30   C=C+SS(KI+L)*SS(KLT+L)
        SS(KLT)=SS(KLT)-C
   40  K=K+1
   50  K=N
       B=0.00
       DO 60 KK=KL,KU
        K=K-1
        KI=MAXA(K)
        C=SS(KK)/SS(KI)
        B=B+C*SS(KK)
   60  SS(KK)=C
       SS(KN)=SS(KN)-B
C
C    STOP EXECUTION IF A ZERO PIVOT IS DETECTED
C
```

```
  70  IF(SS(KN).EQ.0.00) THEN
       PRINT 75,N,SS(KN)
  75    FORMAT('-STIFFNESS MATRIX IS NOT POSITIVE DEFINITE'/'0PIVOT IS
   $   ZERO FOR D.O.F. ',I4/'0PIVOT = ',E15.8)
       STOP
     END IF
C
  80 CONTINUE
C
    RETURN
    END
C************************************************************************
C*                      FORSUB                          *
C************************************************************************
C
C    FORSUB PERFORMS THE FORWARD SUBSTITUTION.
C
    SUBROUTINE FORSUB(SS,Q,MAXA,NEQ)
    IMPLICIT DOUBLE PRECISION (A-H, O-Z)
    DIMENSION SS(*),Q(*),MAXA(*)
C
    DO 20 N=1,NEQ
      KL=MAXA(N)+1
      KU=MAXA(N+1)-1
      KH=KU-KL
      IF(KH.GE.0) THEN
        K=N
        C=0.00
        DO 10 KK=KL,KU
          K=K-1
          C=C+SS(KK)*Q(K)
  10      CONTINUE
        Q(N)=Q(N)-C
      END IF
  20 CONTINUE
C
    RETURN
    END
C************************************************************************
C*                      BACSUB                          *
C************************************************************************
C
C    BACSUB PERFORMS BACK-SUBSTITUTION TO OBTAIN THE SOLUTION.
C
    SUBROUTINE BACSUB(SS,Q,MAXA,NEQ)
    IMPLICIT DOUBLE PRECISION (A-H, O-Z)
    DIMENSION SS(*),Q(*),MAXA(*)
C
    DO 10 N=1,NEQ
      K=MAXA(N)
      Q(N)=Q(N)/SS(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
          K=N
          DO 20 KK=KL,KU
            K=K-1
            Q(K)=Q(K)-SS(KK)*Q(N)
   20     CONTINUE
        END IF
        N=N-1
   30 CONTINUE
C
      RETURN
      END
C**********************************************************************
C*                         UPDATC                          *
C**********************************************************************
      SUBROUTINE UPDATC(X,DD,C1,C2,C3,ELONG,ELENG,DEFLEN,MINC,JCODE,
     $              NE,NJ,NEQ)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION X(3,*),DD(*),C1(*),C2(*),C3(*),ELONG(*),ELENG(*),
     $       DEFLEN(*),MINC(2,*),JCODE(3,*)
C
      DO 30 J = 1, NJ
        DO 20 L = 1, 2
          K = JCODE(L,J)
          IF(K .NE. 0) THEN
             X(L,J) = X(L,J) + DD(K)
          ENDIF
 20       CONTINUE
 30   CONTINUE
C
      CALL LENGTH (X,C1,C2,C3,ELONG,ELENG,DEFLEN,MINC,NE)
C
      RETURN
      END
C**********************************************************************
C*                         LENGTH                          *
C**********************************************************************
      SUBROUTINE LENGTH (X,C1,C2,C3,ELONG,ELENG,DEFLEN,MINC,NE)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION X(3,*),C1(*),C2(*),C3(*),ELONG(*),ELENG(*),DEFLEN(*),
     $       MINC(2,*)
C
      DO 10 I = 1, NE
C
```

```
        J = MINC(1,I)
        K = MINC(2,I)
C
        EL1 = X(1,K) - X(1,J)
        EL2 = X(2,K) - X(2,J)
        EL3 = X(3,K) - X(3,J)
        DEFLEN(I) = DSQRT((EL1**2)+(EL2**2)+(EL3**2))
        ELONG(I) =  DEFLEN(I) - ELENG(I)
        C1(I) = EL1/DEFLEN(I)
        C2(I) = EL2/DEFLEN(I)
        C3(I) = EL3/DEFLEN(I)
C
C        WRITE(*,*) ' '
C        WRITE(*,*) C1(I)
C        WRITE(*,*) C2(I)
C        WRITE(*,*) C3(I)
 10   CONTINUE
C
      RETURN
      END
C**********************************************************************
C*                        FORCES                             *
C**********************************************************************
      SUBROUTINE FORCES(F,QT,AREA,EMOD,ELENG,C1,C2,C3,ELONG,R,MCODE,
     $              MINC,NE,NEQ,D,C1S,C2S)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION F(*),QT(*),AREA(*),EMOD(*),ELENG(*),C1(*),
     $        C2(*),C3(*),ELONG(*),R(*),MCODE(6,*),MINC(2,*),D(*),
     $        C1S(40),C2S(40)
C
C     INITIALIZE THE FORCE VECTOR TO ZERO.
C     CALL INTERF.
C
      DO 10 I = 1,NEQ
        F(I) = 0.D0
 10   CONTINUE
C
      DO 20 I = 1, NE
        CALL INTERF(F,AREA,EMOD,ELENG,C1,C2,C3,ELONG,MCODE,I,D,C1S,C2S)
 20   CONTINUE
C     DO 30 I = 1,NEQ
C     WRITE(*,*)F(I)
C 30  CONTINUE
C
      RETURN
      END
C**********************************************************************
C*                        INTERF                             *
C**********************************************************************
      SUBROUTINE INTERF(F,AREA,EMOD,ELENG,C1,C2,C3,ELONG,MCODE,I,D,C1S,
     $              C2S)
```

```
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION F(*),AREA(*),EMOD(*),ELENG(*),C1(*),C2(*),C3(*),
     $        ELONG(*),MCODE(6,*),AAA(4),ZZZ(4),ELNG(60),D(*),C1S(40),
     $        C2S(40)
C
      CI1=C1S(I)
      CI2=C2S(I)
      CI3=C3(I)
      GAMMA = AREA(I)*EMOD(I)/ELENG(I)
      ELNG(I)=ELENG(I)
C
      DO 10 III=1,4
         ZZZ(III) = 0.0D0
         AAA(III) = 0.D0
 10   CONTINUE
C
      DO 26 II=1,4
        J = MCODE(II,I)
        IF (J .NE. 0) THEN
          ZZZ(II) = D(J)
        END IF
 26   CONTINUE
      ZZZ1=ZZZ(1)
      ZZZ2=ZZZ(2)
      ZZZ3=ZZZ(3)
      ZZZ4=ZZZ(4)
C
          AAA(1) =  C1S(I)*ZZZ1+C2S(I)*ZZZ2
          AAA(2) = -C2S(I)*ZZZ1+C1S(I)*ZZZ2
          AAA(3) =  C1S(I)*ZZZ3+C2S(I)*ZZZ4
          AAA(4) = -C2S(I)*ZZZ3+C1S(I)*ZZZ4
C
          AAA1=AAA(1)
          AAA2=AAA(2)
          AAA3=AAA(3)
          AAA4=AAA(4)
C
            FL1  = -5.0D-1*((2.0D0*AAA3-2.0D0*AAA1)*AREA(I)*ELNG(I)**2+
     1    (AAA4**2-2.0D0*AAA2*AAA4+3.0D0*AAA3**2-6.0D0*AAA1*AAA3+AAA2**2+
     2    3.0D0*AAA1**2)*AREA(I)*ELNG(I)+((AAA3-1.0D0*AAA1)*AAA4**2+(2.0D
     3    0*AAA1*AAA2-2.0D0*AAA2*AAA3)*AAA4+AAA3**3-3.0D0*AAA1*AAA3**2+(A
     4    AA2**2+3.0D0*AAA1**2)*AAA3-1.0D0*AAA1*AAA2**2-1.0D0*AAA1**3)*AR
     5    EA(I))*EMOD(I)/ELNG(I)**3
             F2 = -5.0D-1*(((2.0D0*AAA3-2.0D0*AAA1)*AAA4-2.0D0*AAA2*AA
     1    A3+2.0D0*AAA1*AAA2)*AREA(I)*ELNG(I)+(AAA4**3-3.0D0*AAA2*AAA4**2
     2    +(AAA3**2-2.0D0*AAA1*AAA3+3.0D0*AAA2**2+AAA1**2)*AAA4-1.0D0*AAA
     3    2*AAA3**2+2.0D0*AAA1*AAA2*AAA3-1.0D0*AAA2**3-1.0D0*AAA1**2*AAA2
     4    )*AREA(I))*EMOD(I)/ELNG(I)**3
             FL2=1.D0*F2
             FL3 = 5.0D-1*((2.0D0*AAA3-2.0D0*AAA1)*AREA(I)*ELNG(I)**2+(
     1    AAA4**2-2.0D0*AAA2*AAA4+3.0D0*AAA3**2-6.0D0*AAA1*AAA3+AAA2**2+3
```

```
    2  .0D0*AAA1**2)*AREA(I)*ELNG(I)+((AAA3-1.0D0*AAA1)*AAA4**2+(2.0D0
    3  *AAA1*AAA2-2.0D0*AAA2*AAA3)*AAA4+AAA3**3-3.0D0*AAA1*AAA3**2+(AA
    4  A2**2+3.0D0*AAA1**2)*AAA3-1.0D0*AAA1*AAA2**2-1.0D0*AAA1**3)*ARE
    5  A(I))*EMOD(I)/ELNG(I)**3
          F4 = 5.0D-1*(((2.0D0*AAA3-2.0D0*AAA1)*AAA4-2.0D0*AAA2*AAA
    1  3+2.0D0*AAA1*AAA2)*AREA(I)*ELNG(I)+(AAA4**3-3.0D0*AAA2*AAA4**2+
    2  (AAA3**2-2.0D0*AAA1*AAA3+3.0D0*AAA2**2+AAA1**2)*AAA4-1.0D0*AAA2
    3  *AAA3**2+2.0D0*AAA1*AAA2*AAA3-1.0D0*AAA2**3-1.0D0*AAA1**2*AAA2)
    4  *AREA(I))*EMOD(I)/ELNG(I)**3
          FL4=1.D0*F4
C
     DO 20 L = 1, 4
       K = MCODE(L,I)
       IF (K .NE. 0) THEN
          IF (L .EQ. 1) THEN
         F(K) = (C1S(I)*FL1-C2S(I)*FL2) + F(K)
          ELSEIF (L .EQ. 2) THEN
         F(K) = (C2S(I)*FL1+C1S(I)*FL2) + F(K)
          ELSEIF (L .EQ. 3) THEN
         F(K) = (C1S(I)*FL3-C2S(I)*FL4) + F(K)
          ELSEIF (L .EQ. 4) THEN
         F(K) = (+C2S(I)*FL3+C1S(I)*FL4) + F(K)
C         WRITE(*,*)'F4= ',F(K)
          ENDIF
       ENDIF
 20  CONTINUE
C
    RETURN
    END
C***********************************************************************
C*                         TEST                          *
C***********************************************************************
C        PERFORM  CONVERGENCE  TESTS,  ASSUME  THAT  CONVERGENCE  IS
REACHED,
C    ( SETTING INCONV=0 ) UNTIL IT IS PROVED THE CONTRARY. CALL
C    DISPL, UNBALF
C
    SUBROUTINE TEST(D,DD,DDO,F,FP,FPI,QT,INCONV,NEQ,
   $           TOLFOR,TOLDIS)
    IMPLICIT DOUBLE PRECISION (A-H, O-Z)
    DIMENSION D(*),DD(*),DDO(*),F(*),FP(*),FPI(*),QT(*)
C
    INCONV = 0.D0
    DIVER = 0.D0
C
    IF (TOLDIS .LT. 1.D0) THEN
      CALL DISPL(D,DD,INCONV,NEQ,TOLDIS)
    END IF
C
    IF (TOLFOR .LT. 1.D0) THEN
      CALL UNBALF (F,FP,QT,INCONV,NEQ,TOLFOR)
      END IF
C
C           WRITE(6,*) 'INCONV = ',INCONV
C
      RETURN
      END
C**********************************************************************
C*                         DISPL                         *
C**********************************************************************
C     PERFORM THE DISPLACEMENT CONVERGENCE TEST USING THE EUCLIDEAN
C     VECTOR NORM OF DISPLACEMENTS.
C
      SUBROUTINE DISPL(D,DD,INCONV,NEQ,TOLDIS)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION D(*),DD(*)
C
      DELTAD = 0.D0
      TOTALD = 0.D0
C
C     EUCLIDIAN VECTOR NORM OF DISPLACEMENTS ...........................
      DO 10 I=1,NEQ
       DELTAD = DELTAD + (DD(I))**2
       TOTALD = TOTALD + (D(I))**2
   10 CONTINUE
C
C     CHECK WITH TOLERANCES ..........................................
      IF ( TOTALD.NE.0 ) THEN
        C = ( DSQRT(DELTAD) ) / ( DSQRT(TOTALD) )
        IF ( C.GT.TOLDIS ) THEN
          INCONV = INCONV + 10
        END IF
      ELSE
        WRITE(6,*)' ERROR: DISPLACEMENTS ARE ZERO'
        STOP
      END IF
C
      RETURN
      END
C**********************************************************************
C*                         UNBALF                        *
C**********************************************************************
C     PERFORM A CONVERGENCE TEST FOR THE UNBALANCED FORCE.
C
      SUBROUTINE UNBALF(F,FP,QT,INCONV,NEQ,TOLFOR)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION F(*),FP(*),QT(*)
C
      UNBFI = 0.D0
      UNBFP = 0.D0
C
C     COMPUTE THE UNBALANCED FORCE ......................................
```

```
C
     DO 10 I=1,NEQ
C       WRITE(*,*) '  '
C       WRITE(*,*) 'QT(I) =',QT(I),'F(I) =',F(I)
        UNBFI = UNBFI + (QT(I)-F(I))**2
C       WRITE(*,*) UNBFI
        UNBFP = UNBFP + (QT(I)-FP(I))**2
  10 CONTINUE
C
C    CHECK WITH TOLERANCES ..........................................
C
     IF ( UNBFP.NE.0.D0 ) THEN
       C = ( DSQRT(UNBFI) ) / ( DSQRT(UNBFP) )
       IF ( C.GT.TOLFOR ) THEN
         INCONV = INCONV + 100
       END IF
     ELSE
       INCONV = INCONV + 100
     END IF

     RETURN
     END
C*************************************************************************
C*                          RESULT                             *
C*************************************************************************
     SUBROUTINE RESULT(D,MCODE,JCODE,MINC,NE,NJ,NEQ,LSTEP,QI)
     IMPLICIT DOUBLE PRECISION (A-H, O-Z)
     DIMENSION D(*),MCODE(6,*),JCODE(3,*),MINC(2,*)
C
     REWIND 3
C
     IF(LSTEP .EQ. 0)THEN
       NNN = NEQ + NJ + NE
       DO 5 I = 1, NNN
         J = 0
         WRITE (3) J
 5     CONTINUE
       LSTEP = LSTEP + 1
       REWIND 3
      ENDIF
C
     IF(LSTEP .EQ. 1) THEN
       WRITE(7,10) 'R E S U L T S'
 10    FORMAT(////T33,A/T33,I3,('-'))
     ENDIF

     WRITE(7,30) 'LOADSTEP',LSTEP,QI
 30  FORMAT(///T5,A,T14,':',T39,I4/
    $      T5,'LOADING PARAMETER:',T25,F20.4/)
       WRITE(7,35) 'DOF','DISPL','LAMBDA'
 35    FORMAT(T11,A,T28,A,T42,A/)
```

```
C
C-- READ THE DEGREES OF FREEDOM FOR WHICH THE DISPLACEMENTS ARE TO
BE
C   GIVEN AND TABULATE THE DISPLACEMENTS FOR THE GIVEN EQUILIBRIUM
C   POINT. (TOTAL DISPLACEMENTS)
C
      IF(LSTEP .EQ. 1)THEN
        READ(5,*) J
        WRITE (3) J
      ELSE
        READ (3) J
      ENDIF
C
C
 40   IF(J .NE. 0) THEN
        WRITE(7,50) J, D(J), QI
 50     FORMAT(T10,I3,T20,F15.8,T35,F15.8)
        IF(LSTEP .EQ. 1) THEN
          READ(5,*)  J
          WRITE (3) J
        ELSE
          READ (3) J
        ENDIF
        GO TO 40
      ENDIF
C
      LSTEP = LSTEP + 1
C
C
      RETURN
      END
```