GEOMETRICALLY NONLINEAR ANALYSIS OF PLANE TRUSSES AND PLANE FRAMES

by

Nathan Madutujuh

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

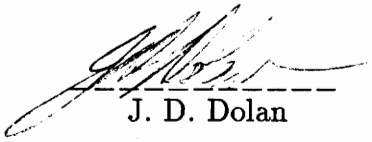
MASTER OF ENGINEERING

in

Civil Engineering

Approved:

S. M. Holzer, Chairman



J. D. Dolan

T. Kuppusamy

December, 1991

Blacksburg, Virginia

ACKNOWLEDGEMENTS

The author sincerely thanks Dr. S. M. Holzer for his wisdom, patience, guidance and encouragement, and for proof-reading this thesis. Much was learned under his guidance. Thanks also go to Dr. J. D. Dolan, and Dr. T. Kupussammy for reviewing this thesis and for serving on the committee.

The discussions with fellow graduate students, Suresh B. Sunku, Niket M. Telang, Budi Ryanto Widjaya, and Chris Earls are very appreciated.

Finally the author gives great thanks to his beloved family, Bapa and Mama, brothers and sister, for their love, support, and prayers given during his college education.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF NOTATION

Chapter 2. Illustrative Example

K,k	Stiffness matrix (global and local)
q,d	Displacement vector
Q,P	Load vector
R	Joint load
e	Elongation
L	Length
Δ	Displacement
F	Element Forces

Chapter 2. General Formulation

W	Total work
We	External work
U,Wi	Internal work
d	Displacement vector
f	Element forces vector
v	Volume
σ	Stress matrix
ϵ	Strain vector
B	Strain-Displacement matrix
$\bar{B}$	Virtual Strain-Displacement matrix
k_s	Secant stiffness matrix
k_t	Tangent stiffness matrix

Chapter 3. Plane Truss Element

ϵ_x	Axial strain
u	Axial deflection
v	Transversal deflection
ϵ_o	Linear axial strain
ϵ_l	Nonlinear axial strain
B_o	Linear B matrix
B_l	Nonlinear B matrix
E	Modulus of elasticity
N	Interpolation matrix
x	Coordinate in axial direction
y	Coordinate in transversal direction
u	Displacement vector of a point at an element
A,Θ	Displacement gradient matrices
G	First derivation of N matrix
S	Stress matrix
S_o	Linear stress matrix
S_l	Nonlinear stress matrix
k_o	Linear tangent stiffness matrix (local)
k_u	Nonlinear displacement tangent stiffness matrix (local)
k_σ	Initial stress tangent stiffness matrix (local)
P	Element axial force
Ag	Section area
Λ,R	Transformation matrix
K_t,k_t	Tangent stiffness matrix (global and local)
c_1,c_2	Direction cosines
λ	Load factor

Chapter 4. Plane Frame Element

ϵ_x	Axial strain
u	Axial deflection
v	Transversal deflection
ϵ_o	Linear axial strain
ϵ_l	Nonlinear axial strain
B_a	Extensional strain B matrix
B_b	Bending strain B matrix
B_v	Nonlinear bending strain B matrix
E	Modulus of elasticity
I	Section Modulus of Inertia
N	Interpolation matrix
x	Coordinate in axial direction
y	Coordinate in transversal direction
u	Displacement vector of a point at an element
A,Θ	Displacement gradient matrices
G	First derivation of N matrix
S	Stress matrix
S_o	Linear stress matrix
S_l	Nonlinear stress matrix
k_a	Axial tangent stiffness matrix (local)
k_b	Bending tangent stiffness matrix (local)
k_σ	Initial stress tangent stiffness matrix (local)
P	Element axial force
Ag	Section area
Λ,R	Transformation matrix
K_t,k_t	Tangent stiffness matrix (global and local)
c_1,c_2	Direction cosines
λ	Load factor

ϵ_x	Axial strain
γ_{xy}	Shear strain
θ	Angle of rotation of a section from the normal plane
u	Axial deflection
v	Transversal deflection
ϵ_o	Linear axial strain
ϵ_l	Nonlinear axial strain
B_a	Extensional strain B matrix
B_θ	Bending strain B matrix
B_v	Nonlinear bending strain B matrix
E	Modulus of elasticity
Gs	Shear modulus
I	Section Modulus of Inertia
k_f	Form factor of a section
N	Interpolation matrix for coordinates and deflection
N_θ	Interpolation matrix for rotation
x	Coordinate in axial direction
y	Coordinate in transversal direction
u	Displacement vector of a point at an element
A,Θ	Displacement gradient matrices
G	First derivation of N matrix
S	Stress matrix
S_o	Linear stress matrix
S_l	Nonlinear stress matrix
k_a	Axial tangent stiffness matrix (local)
k_b	Bending tangent stiffness matrix (local)
k_σ	Initial stress tangent stiffness matrix (local)
k_v	Shear tangent stiffness matrix (local)
P	Element axial force
Ag	Section area
Λ,R	Transformation matrix
K_t,k_t	Tangent stiffness matrix (global and local)

Chapter 1
Introduction

1.1 Introduction

The primary goal of this study is to learn about nonlinear analysis and to implement three finite element models for geometrically nonlinear analysis. Comparisons are made of element stiffness matrix formulations, element force formulations, and solution methods.

The present theory and a standard notation of geometrically nonlinear analysis are reviewed. Three finite elements are used: the 2-node truss element, the 2-node standard plane frame element, and the isoparametric 3-node plane frame element with shear deformation. Two solution schemes are used for this study: the modified Newton-Raphson method and the modified Riks-Wempner method.

The finite element derivations follow the standard notation presented by Wood and Zienkiewicz (1977). The 2-node truss element is based on the work by Wood and Schrefler (1978). The 2-node standard plane frame element is based on Cook et al. (1989). The 3-node plane frame element with shear deformation is based on Hinton and Owen (1979). Mindlin's theory is used to derive this element, so it will be called the 3-node Mindlin frame element. The method for calculation of the element forces for plane frame elements is a modification of an approach by Wood and Zienkiewicz (1977) and Katzenberger (1983).

The descriptions of the solution methods are based on a paper by Holzer et al. (1981). The assembly of element stiffness matrices is done using the MCODE method (Holzer, 1985). The MCODE method uses a matrix to relate the local degree of freedoms of an element to the global degree of fredoms during the assembling process. This matrix is also known as Connectivity Matrix (Bathe, 1982).

1.2 Limitations

Throughout the element formulations, the geometric nonlinearity is limited to a large displacement, large rotation, small strain formulation. Furthermore, it is assumed that only the axial strain contains the nonlinear strain. The shear strain will be assumed to be zero for the standard plane frame element. For the 3-node Mindlin frame element, the shear strain is assumed linear and constant across the element section.

The total Lagrangian formulation is used throughout this thesis. In the total Lagrangian formulation, all variables, differentiations, and integrations are referred to the original reference frame that remains stationery.

The load vector is assumed to be proportional and independent of the deformation of a structure during an analysis.

Chapter 2
Introduction to Nonlinear Analysis

2.1 Introduction

In linear analysis, we assume that the displacements of a finite element model are small, the material is linearly elastic, and the boundary conditions remain unchanged during a load application (Bathe, 1982). The displacement response q, the applied load vector Q, and the stiffness matrix K are related by

$$K \cdot q = Q \tag{2.1}$$

Where K is a constant matrix and the displacement vector q is a linear function of the load vector Q.

In nonlinear analysis, at least one of the three assumptions is not valid. This will cause the matrix K to be dependent on q, thus the displacement vector q is not a linear function of the load vector Q anymore (Bathe, 1982).

Everyday problems are usually analyzed linearly because structural materials are usually used in their linearly elastic range with small deflections. Furthermore, slight nonlinearity does not significantly affect a design based on linear analysis (Cook, 1989).

Although nonlinear analysis usually costs more than linear analysis, there are cases where nonlinear analysis is needed to be able to know the response of a structure during a large deflection. For example, a highrise building enduring a large lateral load, from wind load or earthquake load, will have a large lateral displacement and large axial force at its columns that requires a nonlinear analysis.

2.2 The Sources of nonlinearity

According to Cook et al. (1989), the sources of nonlinearity can be divided into three categories:

1. Geometrically nonlinear
2. Materially nonlinear
3. Contact problems

In geometrically nonlinear problems, the strain-displacement relations are nonlinear and equilibrium is determined for the deformed state of the structures. The strains can be assumed small or large (Bathe, 1982). Most civil engineering structures have small strains even at failure conditions.

A materially nonlinear problem has nonlinear stress-strain relations. To find the strain caused by a load, the equilibrium equation must be written using material properties that depend on the still unknown strain (Cook, et al., 1989).

In a contact problem, the boundary conditions are changing during the application

of a load. Prior to the change in boundary conditions, the response is linear. A free degree-of-freedom may become restrained at a certain load level, and similarly, a restrained degree-of-freedom may become free at a certain load level (Bathe, 1982).

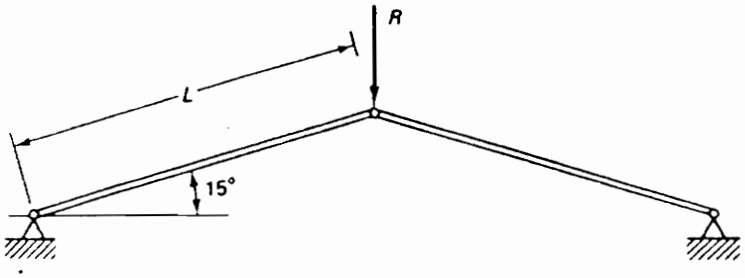
Before performing nonlinear analysis of a problem, it is necessary to determine its significant sources of nonlinearity. The most general large strain formulation will always give better results, but it is not computationally efficient compared to a more restrictive formulation (Bathe, 1982).

2.3 Illustrative example

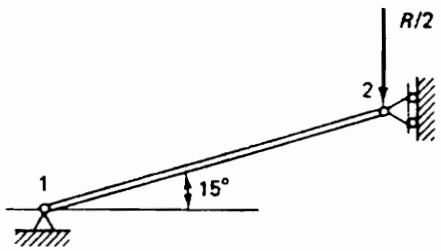
A simple truss arch with two elements and two degrees of freedom will be used to illustrate nonlinear analysis (Bathe, 1982). The truss elements have a length L and make an angle of 15^{o} with the horizontal line. There is a downward load R at the top of the arch (Fig. 2.1a).

To simplify the formulation, we assume that the truss elements only have small strain, thus the axial stiffness k and length L can be assumed to be constant. So, this is a large displacement, small strain problem. Furthermore, by applying the symmetry constraint, the truss arch problem is reduced into a single element and single degree-of-freedom problem (Fig. 2.1b).

To get the nonlinear response of the structure, we must express equilibrium using the deformed configuration. Because the response is nonlinear, we must calculate

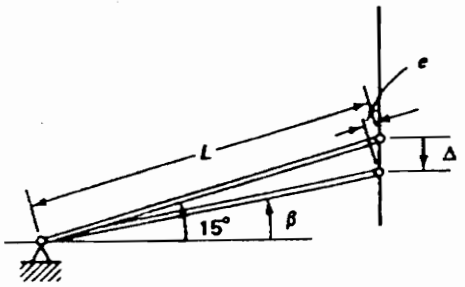


(a) Bar assemblage subjected to apex load

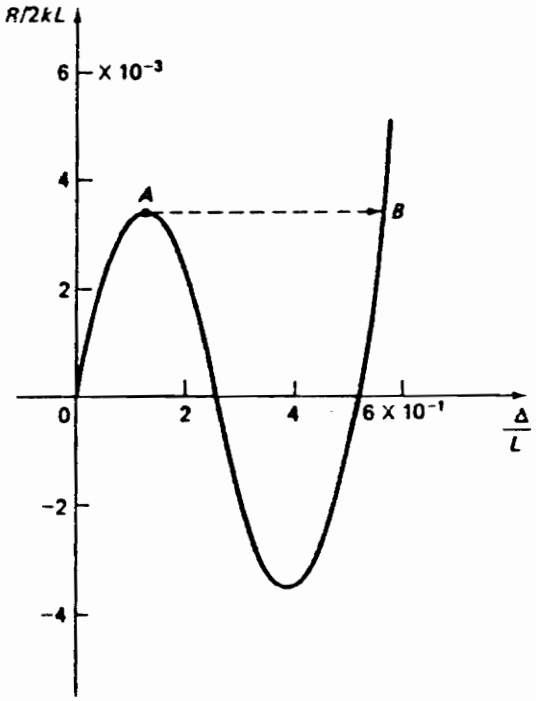


(b) Simple model using one bar (truss) element, nodes 1 and 2

Fig.2.1 Simple Truss Arch Problem
(Bathe, 1982)



(c) Geometric variables in typical configuration



(d) Load-displacement relation

Fig.2.1 Simple Truss Arch Problem
(Bathe, 1982)

the response at several equilibrium points to get the nonlinear response curve, or the equilibrium path.

Because we do not know the characteristic of the curve beforehand, we may jump over a certain region of the curve if we do not specify enough evaluation points. To avoid this, we must iterate along the curve by taking a series of linear steps (Cook et al., 1989). However, in this special case, the explicit equation for the curve can be found easily as follows.

The local axial force of the bar at an iteration step i is given by

$$ {}^{i}F = k \cdot {}^{i}e \tag{2.2} $$

where k is the axial stiffness of the bar and ${}^{i}e$ is the elongation of the bar at load increment i.

The nonlinear response is calculated by examining the equilibrium of the bar in the configuration at increment i (Fig. 2.1c). Using ${}^{i}\Delta$ as downward vertical displacement from the top at increment i, we have

$$ (L - {}^{i}e) \cos {}^{i}\beta = L \cos 15^{o} \tag{2.3} $$

$$ (L - {}^{i}e) \sin {}^{i}\beta = L \sin 15^{o} - {}^{i}\Delta \tag{2.4} $$

hence,

$$ {}^{i}e = L - \sqrt{L^2 - 2L\,{}^{i}\Delta \sin 15^{o} + {}^{i}\Delta^2} \tag{2.5} $$

$$ \sin {}^{i}\beta = \frac{L \sin 15^{o} - {}^{i}\Delta}{L - {}^{i}e} \tag{2.6} $$

Equilibrium at increment i requires that

$$2\,{}^{i}F \sin {}^{i}\beta = {}^{i}R \tag{2.7}$$

Substituting Eqs. (2.2), (2.5), (2.6) into Eq. (2.7) we get the equation that relates the load R and the displacement Δ for each iteration step as

$$\frac{{}^{i}R}{2kL} = \left[-1 + \frac{1}{\sqrt{1 - 2\frac{{}^{i}\Delta}{L}\sin 15^{o} + \left(\frac{{}^{i}\Delta}{L}\right)^{2}}}\right] \cdot \left(\sin 15^{o} - \frac{{}^{i}\Delta}{L}\right) \tag{2.8}$$

Figure 2.1d shows the relation between the load R and the displacement Δ for the simple truss arch problem. It can be seen from Fig. 2.1d that between point A and B the arch has two possible equilibrium configurations. If the load is increased monotonically, the arch will have a snap-through from A to B (Bathe, 1982).

From above example we can see that nonlinear analysis gives more accurate results compared to linear analysis. Moreover, a deeper understanding of the stability and postbuckling behavior of a structure can be learned by examining its equilibrium path.

2.4 General formulation

The general formulation of nonlinear analysis will be presented here. The presentation will be based on two papers by Wood et al. (1977,1978). The nonlinear equilibrium formulation of a finite element will be followed by

linearization of the equilibrium equation to get the incremental equation.

Consider a local finite element defined by the displacement vector d and the element force vector f. According to the principle of virtual work, the total virtual work done by an element in equilibrium is equal to zero. With δW_e equal to the external virtual work done by the element nodal forces, δU equal to the internal virtual work or the first variation of the strain energy of the element, and δW equal to the total virtual work done by the element, we get

$$\delta W = \delta W_e - \delta U = 0 \tag{2.9}$$

where

$$\delta W_e = \delta d^T \cdot f \tag{2.10}$$

and

$$\delta U = \int_v \delta \epsilon^T \sigma \, dv \tag{2.11}$$

where ϵ is the virtual strain vector and σ is the stress vector.

Let

$$\epsilon = B \cdot d \tag{2.12}$$

and

$$\delta \epsilon = \bar{B} \cdot \delta d \tag{2.13}$$

Thus,

$$\delta U = \delta d^T \int_v \bar{B}^T \sigma \, dv \tag{2.14}$$

Substituting Eqs. (2.10) and (2.14) into Eq. (2.9), we get

$$\delta W = \delta d^T [f - \int_v \bar{B}^T \sigma \, dv] = 0 \tag{2.15}$$

or

$$f = \int_v \bar{B}^T \sigma \, dv = k_s \cdot d \tag{2.16}$$

Where k_S is the secant stiffness matrix of the element (Chajes, 1986), a matrix that relates the nodal forces and nodal displacements of the element (Fig. 2.2). The tangent stiffness matrix of the element k_t can be derived by expanding Eq. 2.16 in a first order Taylor's series (Zienkiewicz, 1977). Hence,

$$f(d+\Delta d) = f(d) + \frac{\partial f}{\partial d} \cdot \Delta d + \text{higher order terms} \tag{2.17}$$

By neglecting the higher terms we get

$$\Delta f = f(d+\Delta d) - f(d) = \frac{\partial f}{\partial d} \cdot \Delta d = k_t \cdot \Delta d \tag{2.18}$$

Where Δf is the incremental element force vector, Δd is the incremental nodal displacement vector, and k_t is the incremental stiffness matrix or tangent stiffness matrix of the element (Fig. 2.2). It can be said that Eq. (2.18) is a linearization of Eq. (2.16).

2.5 Solution Techniques

To trace the equilibrium path of a structure, we perform an iterative/incremental analysis. In an incremental analysis, the analysis is done by taking a series of linear steps using the incremental/tangent stiffness matrix. The nonlinear equilibrium equation is linearized at every trial configuration, and the

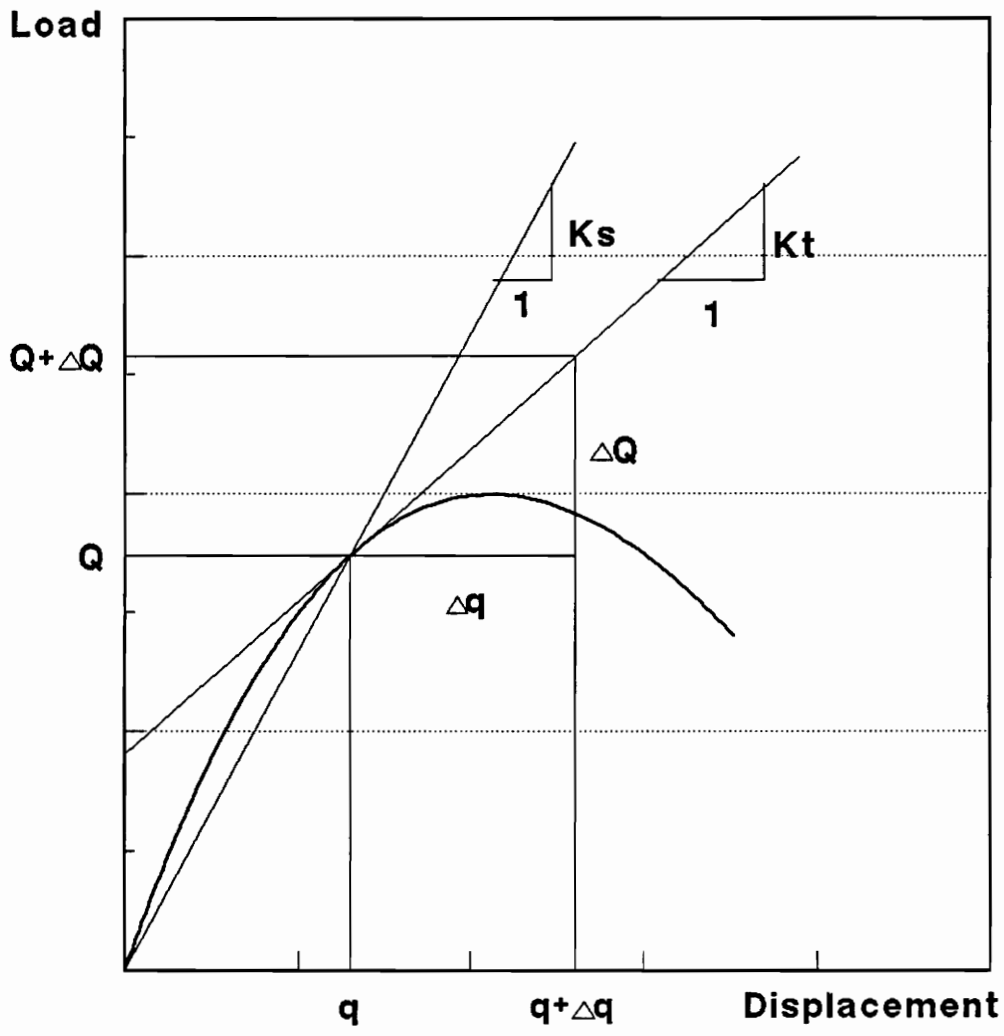


Fig. 2.2 Geometric Interpretation of the Secant and Tangent Stiffness of an SDOF System (Cook, 1989)

displacements calculated from a load step are accumulated during the analysis (Cook, 1989).

Because in every load step we apply the linearized equilibrium equation, an error will be introduced in the solution, and the external load vector and the internal nodal force vector calculated from all element force vectors are not in equilibrium. The difference between the two vectors is the unbalanced load vector or the residual vector. This vector must be used for improving the solution until the solution error is small enough (Fig. 2.3). The accuracy of the solution does not guaranteed the correctness of the results. These depend only on the mesh and the element model.

Because the internal element forces are used for calculating the unbalanced forces that subsequently will be used for correcting the previous result, the accuracy of the final solution depends only on the accuracy of the internal force calculations (Osterrieder and Ramm, 1987). However, the selection of a tangent stiffness matrix formulation method will affect the speed and the stability of a nonlinear solution method.

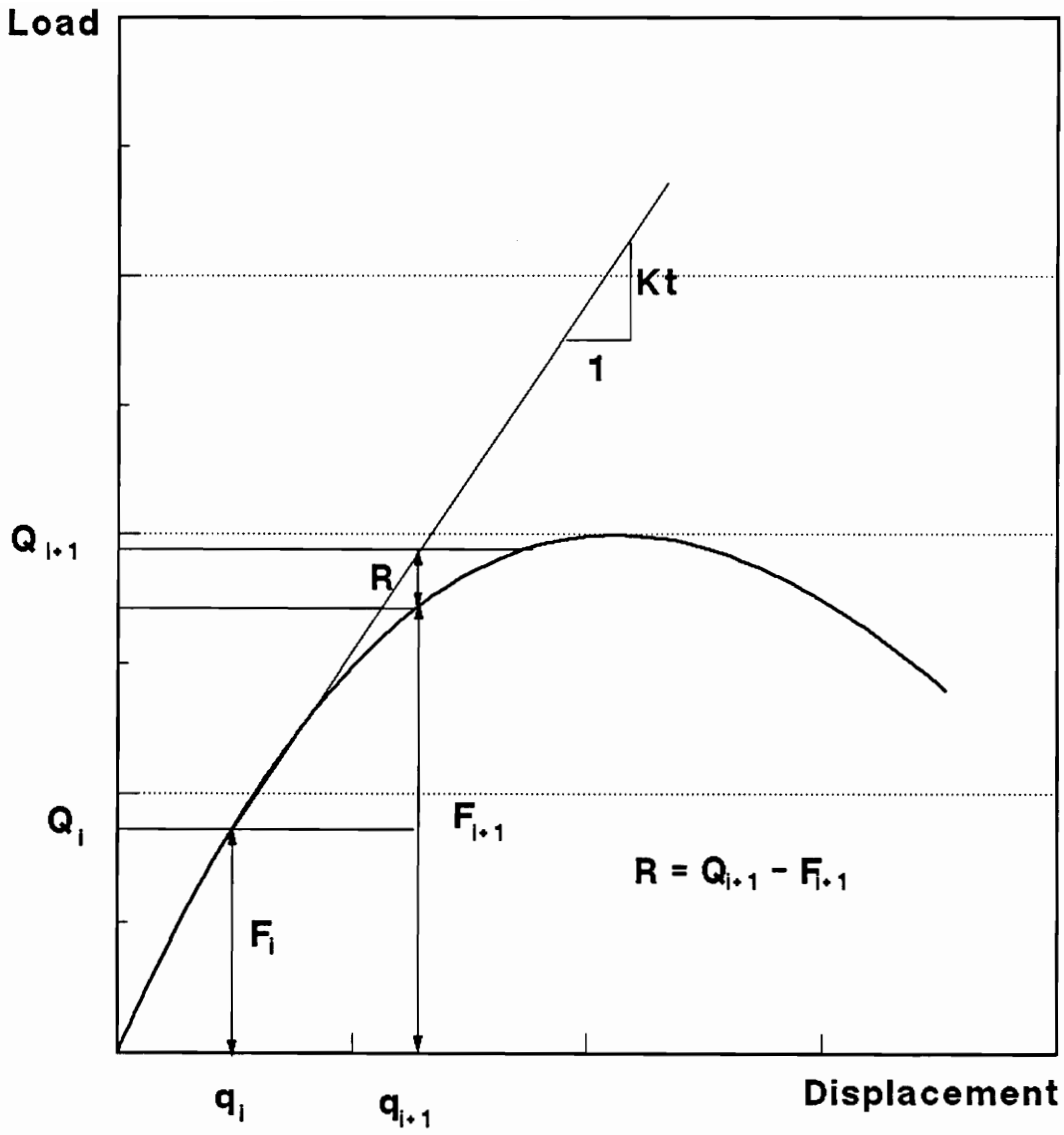


Fig. 2.3 Incremental Analysis
(Cook, 1989)

Chapter 3
Plane Truss Element

3.1 Introduction

Plane truss structures are the simplest models for geometrically nonlinear analysis. Because the formulation is much simpler, the study of this type of structure will provide a good introduction about nonlinear analysis that will be helpful when studying more complex structures.

The 2-node, 4-dof plane truss element in this chapter is based on a paper by Wood et al. (1978). A computer program for geometrically nonlinear analysis of this element is developed using Newton-Raphson and Riks/Wempner methods described by Holzer et al. (1981).

3.2 Strain-displacement relationship

The general strain-displacement relation for a point on a plane truss element is

$$\epsilon_x = u_{,x} + \tfrac{1}{2}\left(u_{,x}^2 + v_{,x}^2 \right) \tag{3.1}$$

where ϵ_x is the axial strain and u and v are the axial and transverse deflections, respectively. Using ϵ_o as the linear axial strain and ϵ_l as the nonlinear axial strain,

we get

$$\epsilon_o = u_{,x} \tag{3.2a}$$

$$\epsilon_1 = \tfrac{1}{2}(u_x^2 + v_{,x}^2) \tag{3.2b}$$

We can rewrite those equations in matrix form using the strain-displacement matrices B_o and B_1 for linear and nonlinear strains as

$$\epsilon_o = B_o \cdot d \tag{3.3a}$$

$$\epsilon_1 = \tfrac{1}{2} \cdot B_1 \cdot d \tag{3.3b}$$

Thus,

$$\epsilon_x = [B_o + \tfrac{1}{2} \cdot B_1] \cdot d = B \cdot d \tag{3.4}$$

Also, the first variation of the axial strain can be expressed by

$$\delta\epsilon_x = [B_o + B_1] \cdot \delta d = \check{B} \cdot \delta d \tag{3.5}$$

3.3 Strain energy

The strain energy of a plane truss element can be expressed as

$$U = \int_{vol} \tfrac{1}{2} \cdot E \cdot \epsilon_x^2 \cdot dv = \int_{vol} \tfrac{1}{2} \cdot E \cdot (u_{,x} + \tfrac{1}{2}(u_x^2 + v_{,x}^2))^2 \cdot dv \tag{3.6}$$

where E is the modulus of elasticity.

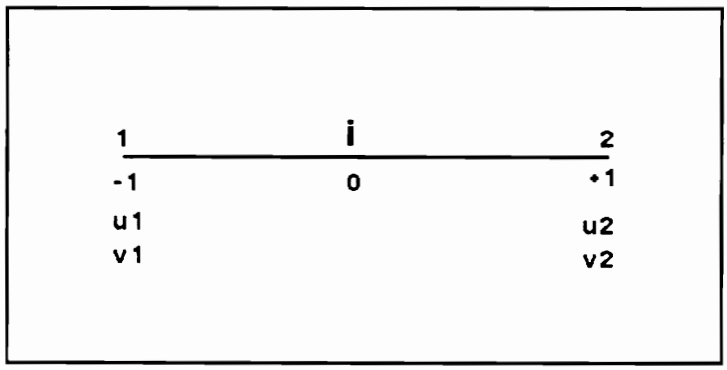


Fig 3.1 Truss Finite Element

Lagrangian 2-node Finite Element

3.4 Interpolation functions

For the 2-node isoparametric truss element, the interpolation function and its first derivatives for coordinates and displacements are:

$$N_1 = \tfrac{1}{2}(1 - \xi) \qquad N_{1,\xi} = -\tfrac{1}{2} \tag{3.7a}$$

$$N_2 = \tfrac{1}{2}(1 + \xi) \qquad N_{2,\xi} = +\tfrac{1}{2} \tag{3.7b}$$

So

$$x = \sum_{i=1}^{2} N_1 \cdot x_i \tag{3.8}$$

and

$$u = \sum_{i=1}^{2} N_1 \cdot u_i \tag{3.9}$$

where x is the coordinate in axial direction, u is the axial displacement of a point on the element, and x_i and u_i are the values of nodal coordinates and displacements, respectively. In matrix notation, displacement vector at any point in the element can be expressed as

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} N_1 & 0 & N_2 & 0 \\ 0 & N_1 & 0 & N_2 \end{bmatrix} \cdot \begin{bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \end{bmatrix} \tag{3.10a}$$

or

$$u = N \cdot d \tag{3.10b}$$

3.5 Displacement gradient matrices

It is convenient to define a displacement gradient matrices as follows (Wood, et

al., 1978).

$$A = \left[\frac{du}{dx} \quad \frac{dv}{dx} \right] \qquad \text{and} \qquad \Theta = A^T = \begin{bmatrix} \frac{du}{dx} \\ \frac{dv}{dx} \end{bmatrix} \tag{3.11}$$

For plane truss with nodal loads only, the displacements vary linearly with x, so the derivation of the displacement in x and y direction can be replaced by

$$\frac{du}{dx} = \frac{\tilde{u}}{L} \tag{3.12a}$$

$$\frac{dv}{dx} = \frac{\tilde{v}}{L} \tag{3.12b}$$

where

$$\tilde{u} = u_2 - u_1 \tag{3.13a}$$

$$\tilde{v} = v_2 - v_1 \tag{3.13b}$$

The G-matrix, as defined in Wood et al. (1978), consists of the first derivation of interpolation functions with respect to x as follows:

$$G = \begin{bmatrix} N_{1,x} & 0 & N_{2,x} & 0 \\ 0 & N_{1,x} & 0 & N_{2,x} \end{bmatrix} = \frac{1}{L} \begin{bmatrix} -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix} \tag{3.14}$$

3.6 Stress matrices

The stress matrix S, a 2x2 matrix, is divided into linear and nonlinear parts as follows:

$$S_o = E \cdot \frac{du}{dx} \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = E \cdot \frac{\tilde{u}}{L} \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.15}$$

$$S_1 = E \cdot \frac{1}{2}\left(u_{,x}^{\,2} + v_{,x}^{\,2}\right) \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \frac{E}{2} \cdot \left[\left(\frac{\bar{u}}{L}\right)^2 + \left(\frac{\bar{v}}{L}\right)^2\right] \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.16}$$

Where s_o is the linear stress matrix, and s_1 is the nonlinear stress matrix.

3.7 Strain-displacement matrices

The component of the strain-displacement matrix in Eq. (3.4) can be expressed as follows:

The linear part,

$$B_o = \begin{bmatrix} N_{1,x} & 0 & N_{2,x} & 0 \end{bmatrix} = \frac{1}{L}\begin{bmatrix} -1 & 0 & 1 & 0 \end{bmatrix} \tag{3.17}$$

while the nonlinear part,

$$B_1 = A \cdot G = \begin{bmatrix} \frac{du}{dx} & \frac{dv}{dx} \end{bmatrix} \cdot \frac{1}{L}\begin{bmatrix} -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix} \tag{3.18a}$$

or

$$B_1 = \frac{1}{L}\begin{bmatrix} \frac{\bar{u}}{L} & \frac{\bar{v}}{L} \end{bmatrix} \cdot \begin{bmatrix} -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix} \tag{3.18b}$$

or

$$B_1 = \frac{1}{L^2}\begin{bmatrix} -\bar{u} & -\bar{v} & \bar{u} & \bar{v} \end{bmatrix} \tag{3.18c}$$

3.8 Tangent stiffness matrix

Using the incremental strain-displacement operator B, the displacement gradient matrix G, and the stress matrix S, the tangent stiffness matrix can be expressed

as (Wood et al., 1978)

$$k_t = \int_v B^t D B \, dv + \int_v G^t S G \, dv \tag{3.19}$$

By expanding the B and S matrices we get

$$k_t = \int_v (B_o + B_l)^t D (B_o + B_l) \, dv + \int_v G^t (S_o + S_l) G \, dv \tag{3.20}$$

or

$$k_t = \int_v B_o{}^t D B_o \, dv + \int_v (B_o{}^t D B_l + B_l{}^t D B_o + B_l{}^t D B_l) \, dv +$$

$$\int_v (G^t S_o G + G^t S_l G) \, dv \tag{3.21}$$

To simplify the derivation, we divide the tangent stiffness matrix k_t into three parts, the linear stiffness matrix k_O, the nonlinear stiffness matrix k_u, and the initial stress stiffness matrix k_σ as follows:

$$k_t = k_o + k_u + k_\sigma \tag{3.22}$$

where

$$k_o = \int_v B_o{}^t D B_o \, dv \tag{3.23}$$

$$k_u = \int_v (B_o{}^t D B_l + B_l{}^t D B_o + B_l{}^t D B_l) \, dv \tag{3.24}$$

$$k_\sigma = \int_v (G^t S_o G + G^t S_l G) \, dv \tag{3.25}$$

The linear stiffness matrix k_O is the stiffness matrix for a linear model. The

nonlinear k_u is a quadratic function of total displacements at the current configuration measured from initial configuration. The initial stress stiffness matrix k_σ is a function of the total axial stress at the current configuration.

3.8.1 Local tangent stiffness matrix

By expanding Eqs. (3.23)-(3.25) and performing numerical integration to find the entries of the matrices, we get explicit expressions for the tangent stiffness matrix in local coordinates as follows.

The linear stiffness matrix,

$$k_o = \int_{vol} B_o^T \cdot D \cdot B_o \cdot dv = \frac{Ag\,L}{2} \int_{-1}^{-1} B_o^T \cdot D \cdot B_o \cdot d\xi \tag{3.26}$$

$$k_o = \frac{EAg\,L}{2L^2} \int_{-1}^{-1} \begin{bmatrix} -1 \\ 0 \\ 1 \\ 0 \end{bmatrix} [-1 \;\; 0 \;\; 1 \;\; 0] \cdot d\xi \tag{3.27}$$

So,

$$K_o = \frac{EAg}{L} \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.28}$$

The nonlinear stiffness matrix k_u consists of linear and quadratic displacements terms. The k_u matrix can be separated into three parts as follows:

$$k_u = \int_{vol} B_o^T \cdot D \cdot B_1 \cdot dv + \int_{vol} B_1^T \cdot D \cdot B_o \cdot dv + \int_{vol} B_1^T \cdot D \cdot B_1 \cdot dv \tag{3.29}$$

or

$$k_u = k_{u1} + k_{u2} + k_{u3} \tag{3.30}$$

where

$$k_{u1} = \frac{Ag\,L}{2} \int_{-1}^{1} B_o^T \cdot D \cdot B_1 \cdot d\xi = \frac{EAg\,L}{2L^3} \int_{-1}^{1} \begin{bmatrix} -1 \\ 0 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} -\bar{u} & -\bar{v} & \bar{u} & \bar{v} \end{bmatrix} d\xi \tag{3.31}$$

$$k_{u2} = \frac{Ag\,L}{2} \int_{-1}^{1} B_1^T \cdot D \cdot B_o \cdot d\xi = \frac{EAg\,L}{2L^3} \int_{-1}^{1} \begin{bmatrix} -\bar{u} \\ -\bar{v} \\ \bar{u} \\ \bar{v} \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 & 0 \end{bmatrix} d\xi \tag{3.32}$$

$$k_{u3} = \frac{Ag\,L}{2} \int_{-1}^{1} B_1^T \cdot D \cdot B_1 \cdot d\xi = \frac{EAg\,L}{2L^4} \int_{-1}^{1} \begin{bmatrix} -\bar{u} \\ -\bar{v} \\ \bar{u} \\ \bar{v} \end{bmatrix} \begin{bmatrix} -\bar{u} & -\bar{v} & \bar{u} & \bar{v} \end{bmatrix} d\xi \tag{3.33}$$

Thus,

$$k_{u1} = \frac{EAg}{L^2} \begin{bmatrix} \bar{u} & \bar{v} & -\bar{u} & \bar{v} \\ 0 & 0 & 0 & 0 \\ -\bar{u} & -\bar{v} & \bar{u} & \bar{v} \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.34}$$

$$k_{u2} = \frac{EAg}{L^2} \begin{bmatrix} \bar{u} & 0 & -\bar{u} & 0 \\ \bar{v} & 0 & -\bar{v} & 0 \\ -\bar{u} & 0 & \bar{u} & 0 \\ -\bar{v} & 0 & \bar{v} & 0 \end{bmatrix} \tag{3.35}$$

$$k_{u3} = \frac{EAg}{L^3} \begin{bmatrix} \bar{u}^2 & \bar{u}\bar{v} & -\bar{u}^2 & -\bar{u}\bar{v} \\ \bar{u}\bar{v} & \bar{v}^2 & -\bar{u}\bar{v} & -\bar{v}^2 \\ -\bar{u}^2 & -\bar{u}\bar{v} & \bar{u}^2 & \bar{u}\bar{v} \\ -\bar{u}\bar{v} & -\bar{v}^2 & \bar{u}\bar{v} & \bar{v}^2 \end{bmatrix} \tag{3.36}$$

The initial stress stiffness matrix is calculated as follows:

$$k_\sigma = \int_{vol} G^T \cdot S_o \cdot G \cdot dv + \int_{vol} G^T \cdot S_1 \cdot G \cdot dv = k_{\sigma 1} + k_{\sigma 2} \tag{3.37}$$

$$k_{\sigma 1} = \frac{Ag\,L}{2} \int_{-1}^{-1} G^T \cdot S_o \cdot G \cdot d\xi \tag{3.38}$$

$$= \frac{EAg\,L}{2L^3} \int_{-1}^{-1} \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \tilde{u} & 0 \\ 0 & \tilde{u} \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix} d\xi \tag{3.39}$$

$$k_{\sigma 2} = \frac{AL}{2} \int_{-1}^{-1} G^T \cdot S_1 \cdot G \cdot d\xi \tag{3.40}$$

$$= \frac{EAg\,L}{2L^2} \int_{-1}^{-1} \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} dn & 0 \\ 0 & dn \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix} d\xi \tag{3.41}$$

where

$$dn = \frac{1}{2} \left[\left(\frac{\tilde{u}}{L}\right)^2 + \left(\frac{\tilde{v}}{L}\right)^2 \right] \tag{3.42}$$

So,

$$k_{\sigma 1} = \frac{EAg}{L^2} \begin{bmatrix} \tilde{u} & 0 & -\tilde{u} & 0 \\ 0 & \tilde{u} & 0 & -\tilde{u} \\ -\tilde{u} & 0 & \tilde{u} & 0 \\ 0 & -\tilde{u} & 0 & \tilde{u} \end{bmatrix} \tag{3.43}$$

$$k_{\sigma 2} = \frac{EAg}{L^2} \begin{bmatrix} dn & 0 & -dn & 0 \\ 0 & dn & 0 & -dn \\ -dn & 0 & dn & 0 \\ 0 & -dn & 0 & dn \end{bmatrix} \tag{3.44}$$

Using

$$\epsilon_x = \frac{\bar{u}}{L} + \frac{1}{2}\left[\left(\frac{\bar{u}}{L}\right)^2 + \left(\frac{\bar{v}}{L}\right)^2\right] \tag{3.45}$$

and

$$P = \frac{EAg}{L} \cdot (\epsilon_x \cdot L) = \frac{EAg}{L} \cdot \Delta L \tag{3.46}$$

we get the initial stress stiffness matrix as follows:

$$k_\sigma = k_{\sigma 1} + k_{\sigma 2} = \frac{P}{L}\begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix} \tag{3.47}$$

Using

$$a = \frac{\bar{u}}{L} \qquad \text{and} \qquad b = \frac{\bar{v}}{L} \tag{3.48}$$

the local tangent stiffness matrix k_t can be written as

$$k_t = \frac{EAg}{L}\begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + \frac{EAg}{L}\begin{bmatrix} a^2+2a & ab+b & -a^2-2a & -ab-b \\ ab+b & b^2 & -ab-b & -b^2 \\ -a^2-2a & -ab-b & a^2+2a & ab+b \\ -ab-b & -b^2 & ab+b & b^2 \end{bmatrix} + \frac{P}{L}\begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix} \tag{3.49}$$

3.8.2 Global tangent stiffness matrix

The global tangent stiffness matrix is calculated by transforming the local tangent stiffness matrix from local axis to global axis (Holzer, 1985) as follows:

$$K_t = \Lambda^T k_t \Lambda \tag{3.50}$$

where

$$\mathbf{\Lambda} = \begin{bmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{0} & \mathbf{R} \end{bmatrix} \quad \text{and} \quad \mathbf{R} = \begin{bmatrix} c1 & c2 \\ -c2 & c1 \end{bmatrix} \tag{3.51}$$

c1 = direction cosine of the element

c2 = direction sine of the element

3.9 Internal forces

There are several methods for calculating the internal forces. The accuracy of the solution depends on what method is used for the internal forces calculation (Ramm et al., 1987). For the full nonlinear analysis, the internal forces can be calculated as follows (Zienkiewicz,1977).

$$F = \int_{vol} \tilde{B}^t \cdot \sigma \cdot dv \tag{3.52}$$

where we can expressed the B matrix and the stress, σ, as function of the total displacement vector, d, measured from the initial configuration as follows:

$$\tilde{B} = B_o + B_l \tag{3.53}$$

$$\sigma = E \cdot \epsilon_x = E \cdot (B_o + \tfrac{1}{2} B_l)\ d \tag{3.54}$$

In matrix form,

$$F = Ag \int_{-1}^{1} \frac{1}{L} \begin{bmatrix} -1-a \\ -b \\ 1+a \\ b \end{bmatrix} \cdot E \cdot \frac{1}{L} \begin{bmatrix} -1-\frac{a}{2} & -\frac{b}{2} & 1+\frac{a}{2} & \frac{b}{2} \end{bmatrix} \cdot \begin{bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \end{bmatrix} d\xi \tag{3.55}$$

or

$$F = \frac{EAg}{L}\int_{-1}^{} \begin{bmatrix} -1-a \\ -b \\ 1+a \\ b \end{bmatrix} \frac{1}{L} \begin{bmatrix} -1-\frac{a}{2} & -\frac{b}{2} & 1+\frac{a}{2} & \frac{b}{2} \end{bmatrix} \begin{bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \end{bmatrix} d\xi \tag{3.56}$$

Performing a matrix multiplication and substituting Eq. (3.48) into Eq. (3.56) we get

$$F = \frac{EAg}{L}\int_{-1}^{} \begin{bmatrix} -1-a \\ -b \\ 1+a \\ b \end{bmatrix} \left\{ \frac{\bar{u}}{L} + \frac{1}{2}\left[\left(\frac{\bar{u}}{L}\right)^2 + \left(\frac{\bar{v}}{L}\right)^2 \right\} \right. d\xi \tag{3.57}$$

or

$$F = \frac{EAg}{L}\int_{-1}^{} \begin{bmatrix} -1-a \\ -b \\ 1+a \\ b \end{bmatrix} \cdot \epsilon_x \cdot d\xi \tag{3.58}$$

Using Eq. (3.58) we will get the local element forces at each iteration.

3.10 Program development

Based on the above derivations, a nonlinear frame analysis program using 2-node truss elements has been developed. The tree chart of the program can be seen at Fig. 3.2.

The program developed here is based on the linear truss analysis program by Holzer (1985). The nonlinear version of the program is created by adding the

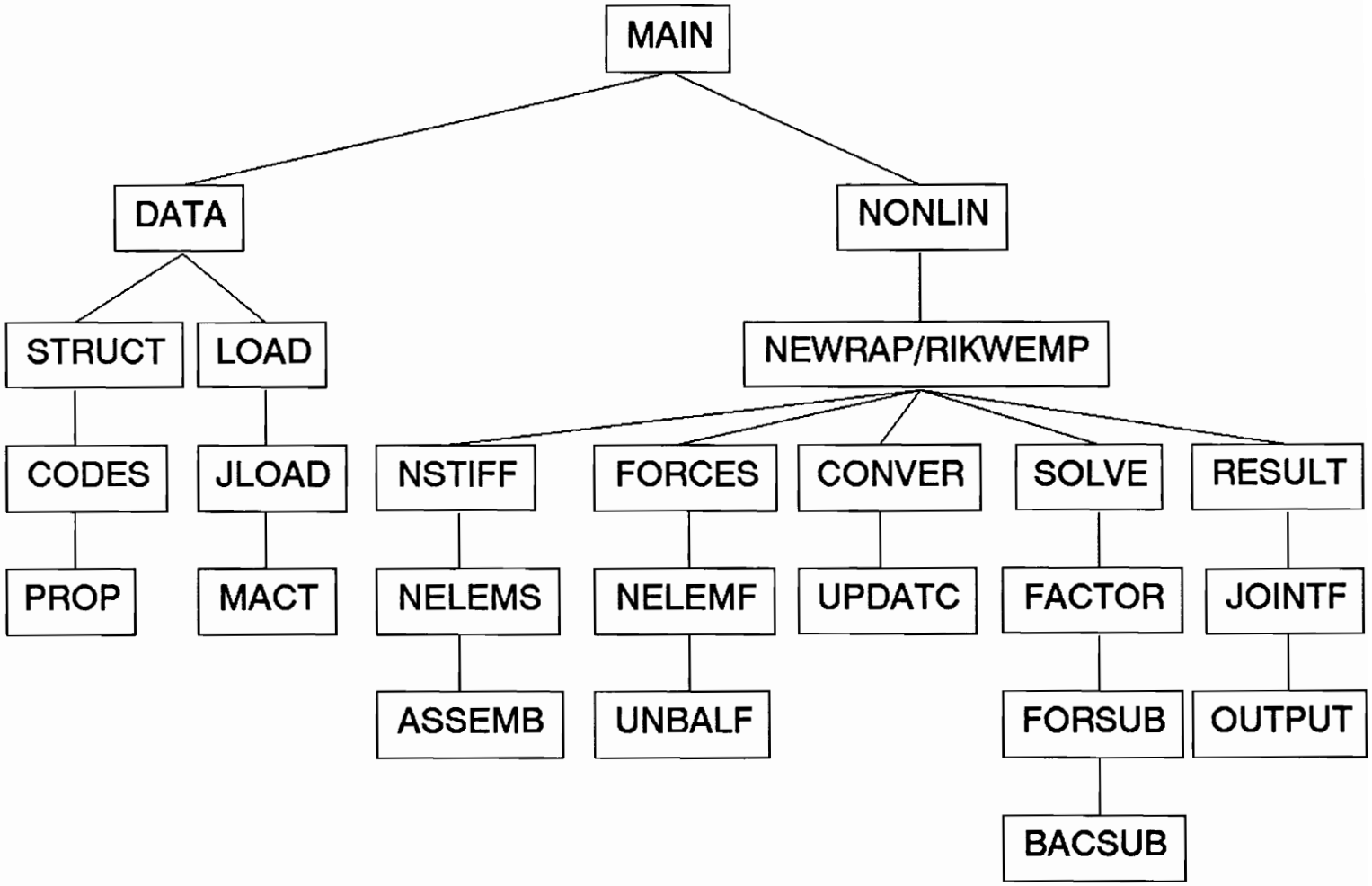


Fig. 3.2 Tree Chart

following subroutines to the linear version :

1. LINEAR - Linear analysis subroutine
2. NONLIN - Nonlinear analysis subroutine
3. NEWRAP - Newton-Raphson Method subroutine
4. RIKWEM - Riks-Wempner Method subroutine
5. UNBALF - Unbalanced Forces subroutine
6. UPDATC - Updating Coordinate subroutine
7. CONVER - Convergence Checking subroutine
8. NSTIFF - Nonlinear Stiffness Matrix subroutine
9. NELEMF - Nonlinear Element Forces subroutine
10. LDISP - Local Displacement subroutine

Basically, the program is divided into two parts, linear and nonlinear analysis, by the LINEAR and NONLIN subroutines. The LINEAR subroutine is exactly same as the second part of MAIN subroutine from the original linear version. The NONLIN subroutine is analog to the LINEAR, except the linear equation solver is replaced by NEWRAP or RIKWEM subroutines.

The NEWRAP subroutine uses the Newton-Raphson algorithm. It uses incremental analysis using variable load step and updates the tangent stiffness matrix at each iteration.

The RIKWEM subroutine is based on the modified Riks-Wempner algorithm (Holzer et al., 1981). It uses incremental analysis using variable path length and updates the tangent stiffness matrix at each iteration (see Chp. 6.4).

The UNBALF subroutine contains the calculation of the unbalanced force vector. It calls the NELEMF for calculating the nonlinear element forces with respect to the local axis and the FORCE subroutine to transform the local element forces to the global joint forces before calculating the unbalanced force vector.

The UPDATC subroutine updates the coordinates of the structure for each iteration. The other parameters as length and cosines directions are calculated on an element basis when they are needed. The updated coordinates are stored in a different array because the initial coordinates are still needed for the total displacement calculation.

The NSTIFF subroutine calculates the nonlinear/tangent stiffness matrix for each element. It uses the axial forces from the previous iteration to calculate the tangent stiffness matrix for the local axis. The local stiffness matrix is then transformed to the global coordinates using the current direction cosines.

The NELEMF subroutine calculates the total nonlinear internal forces by using the secant stiffness matrix method. The total displacement vector, needed for the calculation, is taken from the LDISP subroutine.

3.11 Test problems

The verification of the truss element is done by testing it with a simple truss arch problem (Fig. 3.3). The results are compared to those obtained using ABAQUS. Then a comparison study between the Newton-Raphson and the modified

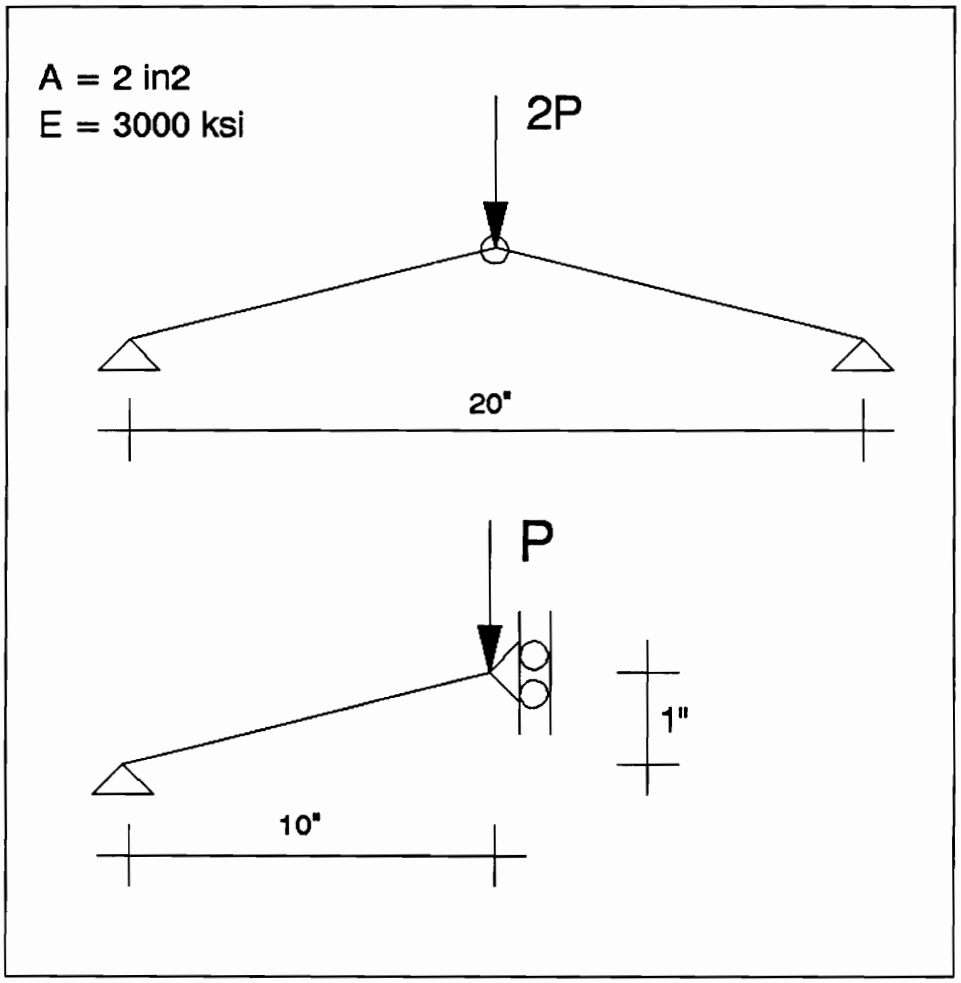


Fig. 3.3 Simple Truss Arch Model

Riks/Wempner method is made. Because of the Newton-Raphson method, the equlibrium path is only traced to 95% of the first limit load/snap-through load.

The verification problem (Fig. 3.2) is analyzed using the modified Riks-Wempner and the Newton-Raphson methods with variable step length. To be able to compare the results with those from ABAQUS, the force tolerance is set to 1.0E-4 for both programs.

Table 3.1 Number of iteration needed for the simple truss arch problem

Solution Method	$\lambda = 0.80$	$\lambda = 0.95$
Riks-Wempner	25	32
Newton-Raphson, variable step	39	44
Newton-Raphson, fixed step	116	140

note: λ = load factor, λ=1 is the first limit load/snap-through load

3.12 Discussions

From the simple truss arch problem, it is found that the nonlinear truss element using $kt = k_o + k_\sigma$, and $kt = k_o + k_\sigma + k_u$, can handle a snap-through problem. Compared to the results obtained using ABAQUS, and the analytical solution, the differences are about 0.3% (Fig. 3.4).

The comparison of the Newton-Raphson method and the Riks-Wempner method is done by running a simple truss arch problem and comparing the number of iterations needed to reach 80% of the first limit load. From the study it was found

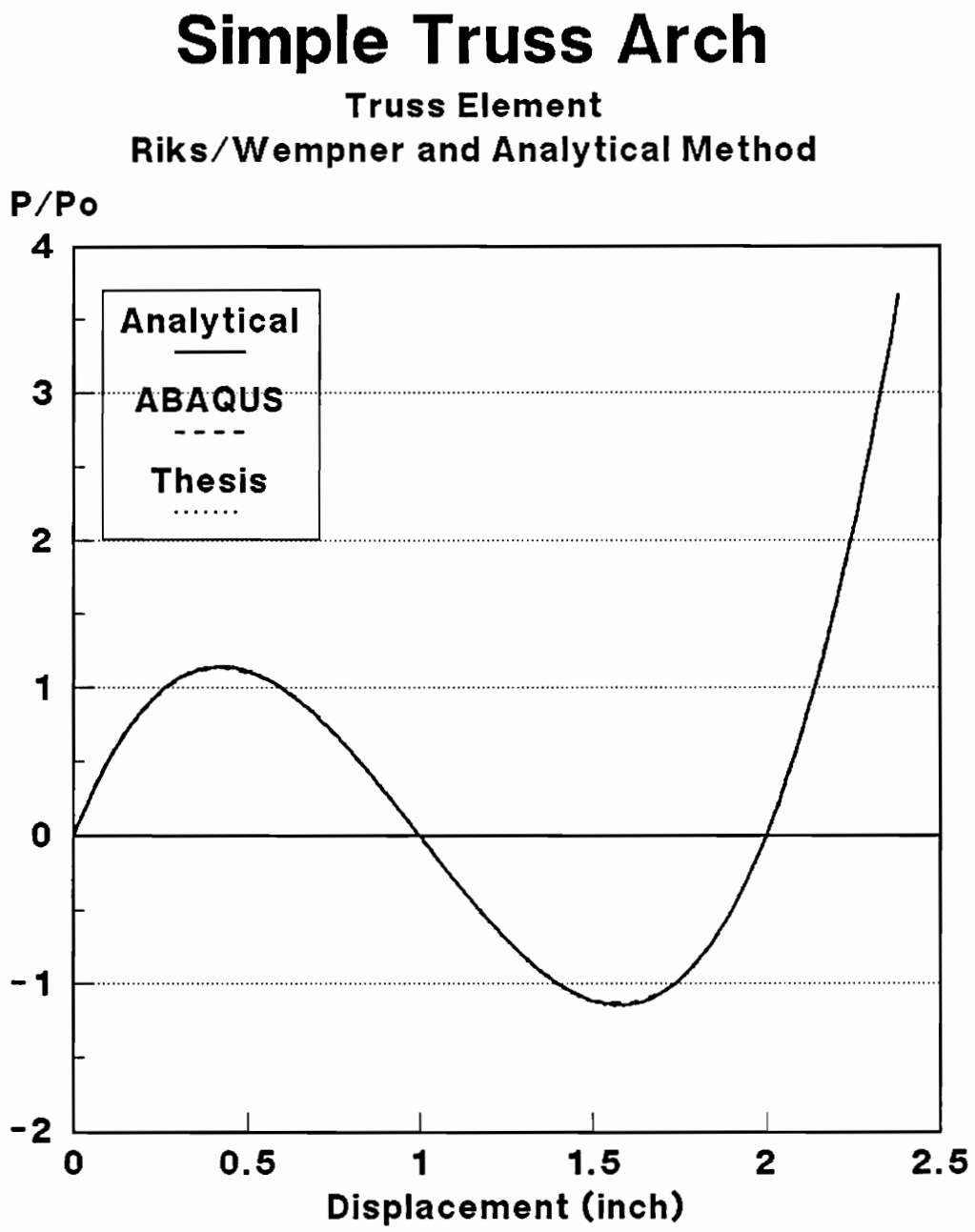


Fig. 3.4 Verification of the Truss Element

that the Riks-Wempner method needs about 25% less iterations than the Newton-Raphson method.

For comparison, the program was run using both a fixed load step and a variable load step, and the Newton-Raphson method. It is found that the number of iterations needed for the fixed load step is about three times of the variable step version.

Chapter 4
Plane Frame Element

4.1 Introduction

In this chapter, a standard plane frame element with Hermitian interpolation will be discussed. This element uses Bernoulli-Euler's slender beam theory for the formulation of its stiffness matrix. The shear strain is assumed equal to zero, and a plane section of the beam will remain plane and normal to the mid-surface during a load application.

A computer program for nonlinear analysis of this element was developed using Newton-Raphson and Riks-Wempner methods. This presentation is based on a paper by Wood et al. (1978) and Katzenberger (1983).

4.2 Strain-displacement relationship

A plane strain element has axial strain caused by axial deformation and bending deformation (Cook et al., 1989). The general strain-displacement relation for a point of a plane frame element is

$$\epsilon_x = u_{,x} + \tfrac{1}{2}\left(u_{,x}^{\,2} + v_{,x}^{\,2} \right) \tag{4-1}$$

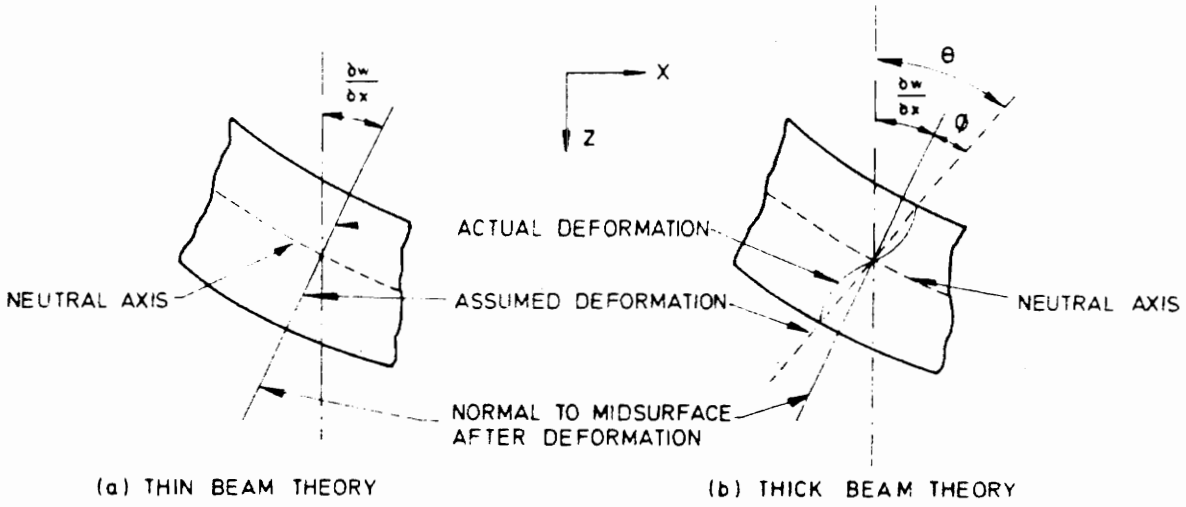


Fig.4.1 Thin Beam Theory

(Bathe, 1982)

where ϵ_x is the axial strain and u and v are the axial and transverse deflections respectively. By using u as the axial deflection at the mid-surface, and assuming that a plane section will remain plane and normal to the mid-surface (Fig. 4.1), we get

$$u(x,y) = u - y \cdot v_{,x} \tag{4-2}$$

$$\epsilon_x(x,y) = (u_{,x} - y \cdot v_{,xx}) + \tfrac{1}{2}((u_{,x} - y \cdot v_{,xx})^2 + v_{,x}^2) \tag{4-3}$$

Furthermore, using the small strain assumption, one can neglect the first quadratic term in the second bracket to get

$$\epsilon_x = u_{,x} - y \cdot v_{,xx} + \tfrac{1}{2} v_{,x}^2 \tag{4-4}$$

4.3 Strain energy

The strain energy can be expressed as

$$U = \int_{vol} \tfrac{1}{2} \cdot E \cdot \epsilon_x^2 \cdot dv = \int_{vol} \tfrac{1}{2} \cdot E \cdot (u_{,x} + \tfrac{1}{2} v_{,x}^2 - y \cdot v_{,xx})^2 \cdot dv \tag{4-5}$$

By using

$$\int_A dA = A, \quad \int_A y \cdot dA = 0, \quad \int_A y^2 \cdot dA = I, \quad \int_A E \cdot u_{,x} \cdot dA = P \tag{4-6}$$

we get

$$U = \frac{EAg}{2} \int_0^L u_{,x}^2 \cdot dx + \frac{EI}{2} \int_0^L v_{,xx}^2 \cdot dx + \frac{P}{2} \int_0^L v_{,x}^2 \cdot dx \tag{4-7}$$

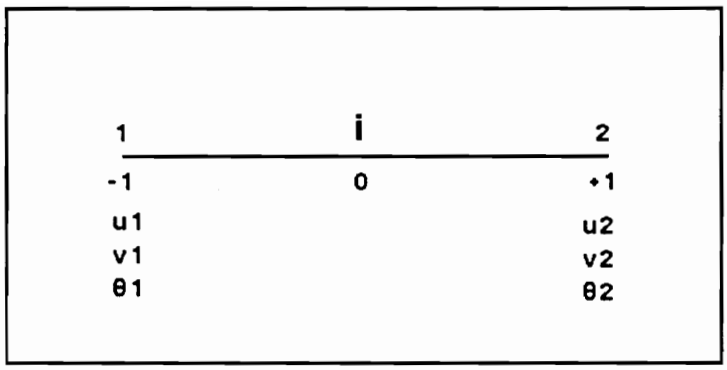


Fig 4.2 Bernoulli-Euler Frame Element

Hermitian 2-node Finite Element

4.4 Interpolation functions

The 2-node Hermitian element uses a linear Lagrangian interpolation function for the axial displacement and a cubic Hermitian interpolation function for the transverse displacement (Fig. 4.2).

The interpolation functions and their first derivatives for coordinates and axial displacements are

$$\hat{N}_1 = \tfrac{1}{2}(1-\xi) \qquad \hat{N}_2 = \tfrac{1}{2}(1+\xi) \tag{4-8a}$$

$$\hat{N}_{1,\xi} = -\tfrac{1}{2} \qquad \hat{N}_{2,\xi} = +\tfrac{1}{2} \tag{4-8b}$$

where

$$x = \sum_{i=1}^{2} \hat{N}_i \cdot x_i \tag{4-9}$$

and

$$u = \sum_{i=1}^{2} \hat{N}_i \cdot u_i \tag{4-10}$$

The Hermitian interpolation functions and their first and second derivatives with regard to ξ are defined as follows:

$$N_1 = \tfrac{1}{4}(2+\xi)(1-\xi)^2 \qquad N_{1,\xi} = \tfrac{-3}{4}(1-\xi)(1+\xi) \qquad N_{1,\xi\xi} = \tfrac{3}{2}\xi \tag{4-11a}$$

$$N_2 = \tfrac{L}{8}(1+\xi)(1-\xi)^2 \qquad N_{2,\xi} = \tfrac{-L}{8}(1-\xi)(1+3\xi) \qquad N_{2,\xi\xi} = \tfrac{L}{4}(3\xi-1) \tag{4-11b}$$

$$N_3 = \tfrac{1}{4}(2-\xi)(1+\xi)^2 \qquad N_{3,\xi} = \tfrac{3}{4}(1+\xi)(1-\xi) \qquad N_{3,\xi\xi} = -\tfrac{3}{2}\xi \tag{4-11c}$$

$$N_4 = \tfrac{L}{8}(\xi-1)(1+\xi)^2 \qquad N_{4,\xi} = \tfrac{L}{8}(1+\xi)(3\xi-1) \qquad N_{4,\xi\xi} = \tfrac{L}{4}(3\xi+1) \tag{4-11d}$$

4.5 Strain-displacement matrices

The Green-Lagrange strains can be expressed in terms of the interpolation functions defined above. The strains are separated into three parts as follows.

The extensional strain,

$$\epsilon_x = u_{,x} = B_a \cdot d \tag{4-12}$$

where

$$B_a = \begin{bmatrix} \hat{N}_{1,x} & 0 & 0 & \hat{N}_{2,x} & 0 & 0 \end{bmatrix} \tag{4-13}$$

$$d = \{ \; u_1 \quad v_1 \quad \theta_1 \quad u_2 \quad v_2 \quad \theta_2 \; \} \tag{4-14}$$

The linear bending strain,

$$\epsilon_x = - y \cdot v_{,xx} = -y \cdot B_b \cdot d \tag{4-15}$$

where

$$B_b = \begin{bmatrix} 0 & N_{1,xx} & N_{2,xx} & 0 & N_{3,xx} & N_{4,xx} \end{bmatrix} \tag{4-16}$$

For nonlinear bending strain,

$$\epsilon_x = \tfrac{1}{2} v^2_{,x} = \tfrac{1}{2} (B_v \cdot d)^T (B_v \cdot d) \tag{4-17}$$

where

$$B_v = \begin{bmatrix} 0 & N_{1,x} & N_{2,x} & 0 & N_{3,x} & N_{4,x} \end{bmatrix} \tag{4-18}$$

The displacement gradient matrix, G, defined in Wood et al. (1978), is identical to B_V, the strain-displacement operator that consists of the first derivative of the interpolation functions.

$$G = B_v = \begin{bmatrix} 0 & N_{1,x} & N_{2,x} & 0 & N_{3,x} & N_{4,x} \end{bmatrix} \tag{4-19}$$

4.6 Tangent stiffness matrix

The tangent stiffness matrix derived here is a simplified version of the tangent stiffness matrix defined by Wood et al. (1978). Only the linear strain-displacement matrix B_0 and the strain-displacement gradient matrix G are used. The nonlinear strain-displacement matrix B_1 is neglected. By substituting Eqs. (4-12) - (4-18) into Eq. (4-7) and changing the integration variable x to ξ, we get

$$U = \tfrac{1}{2} d^T \cdot k_a \cdot d + \tfrac{1}{2} d^T \cdot k_b \cdot d + \tfrac{1}{2} d^T \cdot k_\sigma \cdot d + \tfrac{1}{2} d^T \cdot k_s \cdot d \tag{4-20}$$

where,

$$k_a = EAg \int_0^L B_a{}^T \cdot B_a \cdot dx = \frac{EAg\,L}{2} \int_{-1}^{1} B_a{}^T \cdot B_a \cdot d\xi \tag{4-21}$$

$$k_b = EI \int_0^L B_b{}^T \cdot B_b \cdot dx = \frac{EIL}{2} \int_{-1}^{1} B_b{}^T \cdot B_b \cdot d\xi \tag{4-22}$$

$$k_\sigma = P \int_0^L G^T \cdot G \cdot dx = \frac{PL}{2} \int_{-1}^{1} G^T \cdot G \cdot d\xi \tag{4-23}$$

4.6.1 Local tangent stiffness matrix

By expanding Eqs. (4-22) - (4-25) into matrix forms and performing numerical integrations to find the entries of the matrices, we get explicit expressions for all terms of the tangent stiffness matrix with respect to the local axis as follows:

$$k_a = \gamma \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \qquad \gamma = \frac{EAg}{L} \tag{4-24}$$

$$k_b = \alpha \begin{bmatrix} 12 & 6L & -12 & 6L \\ 6L & 4L^2 & -6L & 2L^2 \\ -12 & -6L & 12 & -6L \\ 6L & 2L^2 & -6L & 4L^2 \end{bmatrix} \qquad \alpha = \frac{EI}{L^3} \tag{4-25}$$

$$k_\sigma = \frac{P}{30L} \begin{bmatrix} 36 & 3 & -36 & 3 \\ 3 & 4 & -3 & -1 \\ -36 & -3 & 36 & -3 \\ 3 & -1 & -3 & 4 \end{bmatrix} \qquad \text{P = positive in tension} \tag{4-26}$$

The three local stiffness matrices can be expanded to 6x6 matrices and combined together to form a 6x6 element stiffness matrix in local coordinates.

4.6.2 Global tangent stiffness matrix

The global tangent stiffness matrix is calculated by transforming the local stiffness matrix from the local axis to the global axis. The stiffness matrix assembly process uses the index matrix method (Holzer, 1985). To be able to use the same index matrix for linear and nonlinear stiffness matrices, the global initial stress matrix terms are arranged according to that of the linear stiffness matrix.

The global stiffness matrix is calculated by

$$K_t = \Lambda^T k_t \Lambda \tag{4-27}$$

where Λ is the 6x6 transformation matrix. By splitting the global stiffness matrix into four submatrices, we get

$$K_t = \begin{bmatrix} R^T & 0 \\ 0 & R^T \end{bmatrix} \begin{bmatrix} kaa & kab \\ kba & kbb \end{bmatrix} \begin{bmatrix} R & 0 \\ 0 & R \end{bmatrix} = \begin{bmatrix} R^T \cdot kaa \cdot R & R^T \cdot kab \cdot R \\ R^T \cdot kba \cdot R & R^T \cdot kbb \cdot R \end{bmatrix} \quad (4\text{-}28)$$

where R is the 3x3 direction cosine matrix. The entries of both linear and nonlinear global stiffness matrices are arranged as follows.

$$K_t = \begin{bmatrix} G1 & G2 & G4 & -G1 & -G2 & G4 \\ & G3 & G5 & -G2 & -G3 & G5 \\ & & G6 & -G4 & -G5 & G7 \\ & & & G1 & G2 & -G4 \\ & symm. & & & G3 & -G5 \\ & & & & & G6 \end{bmatrix} \quad (4\text{-}29)$$

The contribution from the axial and bending stiffness matrix are

$$\alpha = \frac{EI}{L^3} \qquad \beta = \frac{Ag\,L^2}{I} \qquad \theta = \frac{P}{30L} \quad (4\text{-}30)$$

$$G1 = \alpha \cdot (\beta \cdot cx^2 + 12\, cy^2) \quad (4\text{-}31a)$$

$$G2 = \alpha \cdot cx \cdot cy\, (\beta - 12) \quad (4\text{-}31b)$$

$$G3 = \alpha \cdot (\beta \cdot cy^2 + 12\, cx^2) \quad (4\text{-}31c)$$

$$G4 = \alpha \cdot (-6\, L\, cy) \quad (4\text{-}31d)$$

$$G5 = \alpha \cdot (6\, L\, cx) \quad (4\text{-}31e)$$

$$G6 = \alpha \cdot (4\, L^2) \quad (4\text{-}31f)$$

$$G7 = \alpha \cdot (2\, L^2) \quad (4\text{-}31g)$$

The contribution from the initial stress matrix are

$$G1 = \theta \cdot (36\, cy^2) \quad (4\text{-}32a)$$

$$G2 = \theta \cdot (-36\, cx\, cy) \quad (4\text{-}32b)$$

$$G3 = \theta \cdot (36\, cx^2) \quad (4\text{-}32c)$$

$$G4 = \theta \cdot (-3\, cy) \quad (4\text{-}32d)$$

$$G5 = \theta \cdot (3\ cx) \tag{4-32e}$$

$$G6 = \theta \cdot (4) \tag{4-32f}$$

$$G7 = \theta \cdot (-1) \tag{4-32g}$$

The index matrix for both stiffness matrices is

$$INDEX = \begin{bmatrix} 1 & 2 & 4 & -1 & -2 & 4 \\ & 3 & 5 & -2 & -3 & 5 \\ & & 6 & -4 & -5 & 7 \\ & & & 1 & 2 & -4 \\ & \text{symm.} & & & 3 & -5 \\ & & & & & 6 \end{bmatrix} \tag{4-33}$$

4.7 Internal forces

Not as in the tangent stiffness matrix formulation, the element forces are calculated using the full nonlinear formulation. Because the shear deformations are not included in the formulation, the shear forces can only be calculated from moment equilibrium of all forces in the elements (Katzenberger, 1983).

The calculation is based on the secant stiffness matrix from Katzenberger (1983), where the total displacements are transformed into the rigid body motion of the element and the local displacements of the element measured from the current configuration.

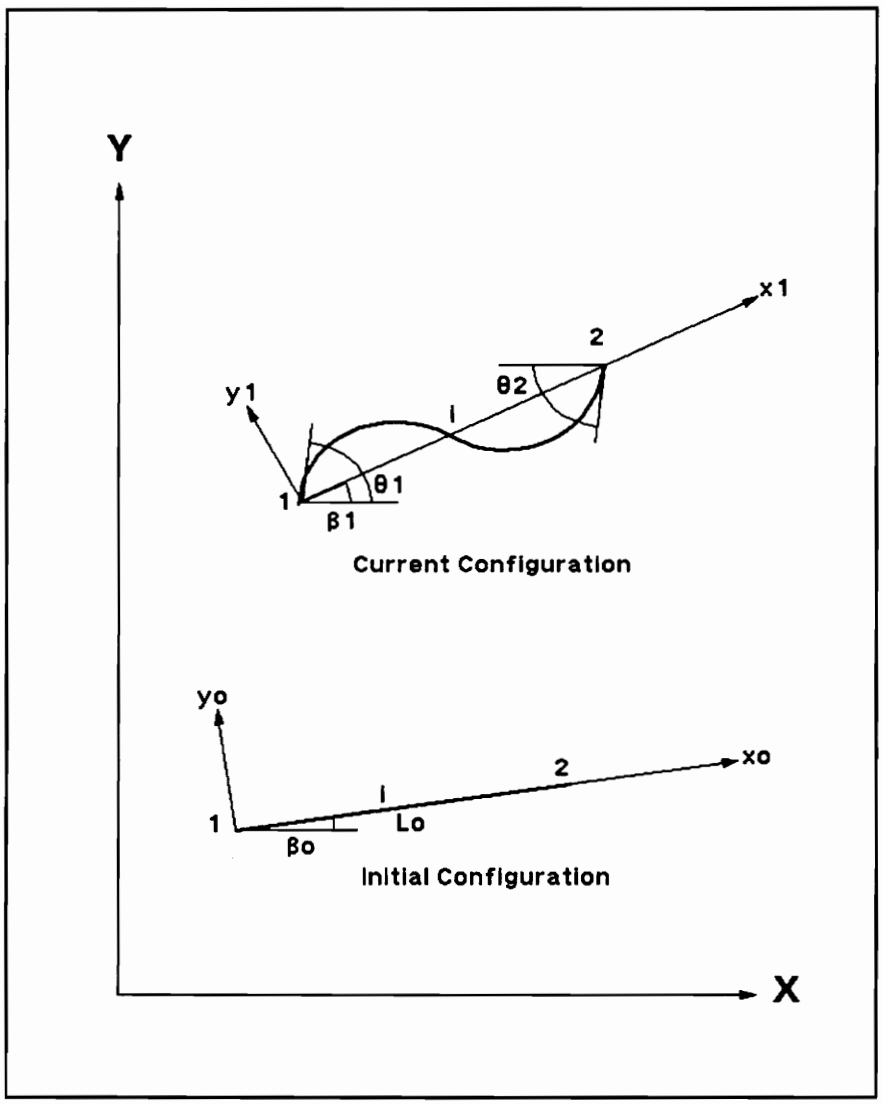


Fig. 4.3 Bernoulli-Euler Frame Element in Global Coordinates

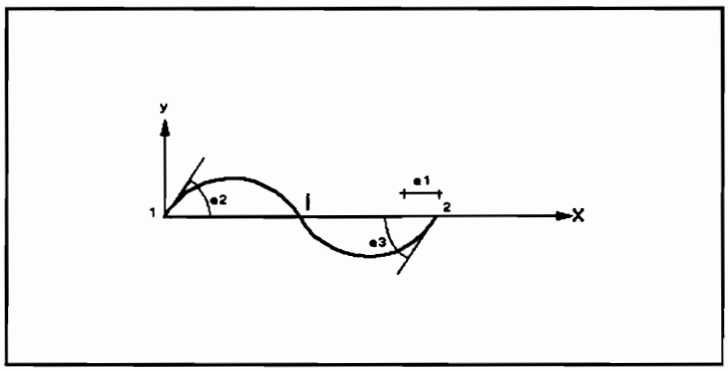


Fig 4.4 Bernoulli-Euler Frame Element in Local Coordinates

The reference frame is changing for every iteration, so the updated Lagrangian approach is used here. Although one can use the total Lagrangian approach to calculate the secant stiffness matrix, this approach uses smaller number of parameters and only 3x3 matrices compared to 6x6 matrices from total Lagrangian approach (Fig. 4.3).

First, the element is moved using a rigid body motion so that the left node is equal to the updated left node position. Then, the element is rotated by $\Delta\theta$ so that the other node will be at its updated location. The total rotations of each node that come from the previous step must be corrected by subtracting from them $\Delta\theta$, while the local transverse deflections are equal to zero. The axial deflection is equal to zero at left node, and equal to the total element elongation at the right node. The distribution of the axial deflection is assumed to be linear.

Using this approach, one will get three independent displacements or natural displacements ΔL, θ'_1, and θ'_2 (Fig. 4.4). The interpolation functions for displacements in x and y directions are,

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} N_1 & 0 & 0 \\ 0 & N_2 & N_3 \end{bmatrix} \cdot \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} \tag{4-34}$$

or

$$u = N \cdot d \tag{4-35}$$

where

$$N_1 = \xi \tag{4-36a}$$

$$N_2 = L\xi\,(1 - 2\xi + \xi^2) \tag{4-36b}$$

$$N_3 = L\xi^2\,(\xi - 1) \tag{4-36c}$$

$$e_1 = \Delta L = L_1 - L_o \tag{4-36a}$$

$$e_2 = \theta'_1 = \theta_1 - \Delta\theta \tag{4-36b}$$

$$e_3 = \theta'_2 = \theta_2 - \Delta\theta \tag{4-36c}$$

The parameter $\Delta\theta$ can be calculated easily using a vector operation involving the initial configuration and the current configuration of the element. The sign of $\Delta\theta$ will be positive if the cross product of the initial local axis vector and the updated local axis vector gives the outward vector according to the right hand rule.

Using the same approach as Wood et al. (1978), but with above interpolation functions, we will get the secant stiffness matrices of the element. The secant stiffness matrices can be symmetric or unsymmetric, but both matrices are equivalent operators.

The strain-displacement relations are

$$\epsilon_x = \epsilon_o + \epsilon_1 = u_{,x} - y \cdot v_{,xx} + \tfrac{1}{2} v_{,x}^2 \tag{4-37}$$

$$\epsilon_x = B_o \cdot d + \tfrac{1}{2} B_1 \cdot d = (B_o + \tfrac{1}{2} B_1)\, d \tag{4-38}$$

$$\delta\epsilon_x = \delta\epsilon_o + \delta\epsilon_1 = B_o \cdot \delta d + B_1 \cdot \delta d = (B_o + B_1)\, \delta d = \bar{B} \cdot \delta d \tag{4-39}$$

The unsymmetric secant stiffness matrix is (Wood et al., 1978)

$$k_s = \int_v (B_o + B_1)^t\, D\, (B_o + \tfrac{1}{2} B_1)\, dv \tag{4-40}$$

And the symmetric form of secant stiffness matrix is (Wood et al., 1978)

$$k_s = \int_v B_o^t D B_o \, dv + \frac{1}{2} \int_v (B_o^t D B_1 + B_1^t D B_o + G^t S_o G) \, dv +$$

$$\frac{1}{3} \int_v (B_1^t D B_1 + G^t S_1 G) \, dv \tag{4-41}$$

The local element forces at the current configuration can be calculated by

$$f = k_s \cdot d \tag{4-42}$$

Where {f} is a 3x1 matrix, {d} is a 3x1 matrix, and $[k_S]$ is a 3x3 matrix. Katzenberger (1983) gives the explicit expressions for calculating element forces using the symmetric form of the secant stiffness matrix as follows:

$$P_1 = \frac{EAg}{L} e_1 + \frac{EAg}{60} (4 e_2 - e_3) e_2 + \frac{EAg}{60} (4 e_3 - e_2) e_3 \tag{4-43}$$

$$P_2 = \frac{EAg}{30} (4 e_2 - e_3) e1 + \frac{4EI}{L} e_2 + \frac{2EI}{L} e_3 + \frac{EAg\,L}{420} (12 e_2^2 - 3 e_2 e_3 + e_3^2) e_2 +$$
$$\frac{EAg\,L}{840} (-3 e_2^2 + 4 e_2 e_3 - 3 e_3^2) e_3 \tag{4-44}$$

$$P_3 = \frac{EAg}{30} (4 e_3 - e_2) e1 + \frac{2EI}{L} e_2 + \frac{4EI}{L} e_3 + \frac{EAg\,L}{840} (-3 e_2^2 + 4 e_2 e_3 - 3 e_3^2) e_2 +$$
$$\frac{EAg\,L}{420} (e_2^2 - 3 e_2 e_3 + 12 e_3^2) e_3 \tag{4-45}$$

The local element forces P_1, P_2, and P_3 correspond to the displacements e_1, e_2, e_3 respectively. The other forces are calculated using the following equations:

$$N_1 = -P_1, \; N_2 = +P_1 \tag{4-46}$$

$$M_1 = +P_2 \, , M_2 = +P_3 \tag{4-47}$$

$$V_1 = (M_1 + M_2)/L, \; V_2 = -V_1 \tag{4-48}$$

4.8 Program development

Based on the above derivations, a nonlinear frame analysis program using the Bernoulli-Euler elements has been developed. The tree chart can be seen in Fig. 3.2.

The program developed here is based on the linear frame analysis program by Holzer (1985). The nonlinear version of the program is created by adding the following subroutines to the linear version :

1. LINEAR - Linear analysis subroutine
2. NONLIN - Nonlinear analysis subroutine
3. NEWRAP - Newton-Raphson Method subroutine
4. RIKWEM - Riks-Wempner Method subroutine
5. UNBALF - Unbalanced Forces subroutine
6. UPDATC - Updating Coordinate subroutine
7. CONVER - Convergence Checking subroutine
8. NSTIFF - Nonlinear Stiffness Matrix subroutine
9. NELEMF - Nonlinear Element Forces subroutine
10. LDISP - Local Displacement subroutine

Basically, the program is divided into two parts, linear and nonlinear analysis, by the LINEAR and NONLIN subroutines. The LINEAR subroutine is exactly same as the second part of MAIN subroutine from the original linear version. The NONLIN subroutine is analog to the LINEAR, except the linear equation solver is replaced by NEWRAP or RIKWEM subroutines.

The NEWRAP subroutine uses the Newton-Raphson algorithm. It uses incremental analysis using variable load step and updates the tangent stiffness matrix at each iteration.

The RIKWEM subroutine is based on the modified Riks-Wempner algorithm by Holzer et al. (1981). It uses incremental analysis using variable path length and updates the tangent stiffness matrix at each iteration.

The UNBALF subroutine contains the calculation of the unbalanced force vector. It calls the NELEMF for calculating the nonlinear element forces with respect to the local axis and the FORCE subroutine to transform the local element forces to the global joint forces before calculating the unbalanced force vector.

The UPDATC subroutine updates the coordinates of the structure for each iteration. The other parameters as length and cosines directions are calculated at element basis when they are needed. The updated coordinates are stored at different arrays because the initial coordinates are still needed for the total displacement calculation.

The NSTIFF subroutine calculates the nonlinear/tangent stiffness matrix for each element. It uses the axial forces from the previous iteration to calculate the tangent stiffness matrix with respect to the local axis. Then the local stiffness matrix is transformed to the global coordinates using the current direction cosines.

The NELEMF subroutine calculates the total nonlinear internal forces by secant stiffness matrix method. The total displacement vector needed for the calculations

is taken from the LDISP subroutine.

4.9 Test problems

The nonlinear Bernoulli-Euler frame element is verified by testing it with some problems that have analytical solutions, and by comparing it with a commercial program for large nonlinear analysis, ABAQUS. The verification of the nonlinear Bernoulli-Euler frame element is done as follows.

First, the element is tested with a simple truss arch problem to find out the capability of the element to handle a snap-through. Only the nonlinear truss modelling capability is tested here (Fig. 3.3).

Second, the element is tested for a simple rigid arch model. The results are compared with that of ABAQUS (Fig. 4.5).

All test problems are analyzed using the modified Riks-Wempner methods with variable step length. To be able to compare the results with that from ABAQUS, the force and moment tolerances are set to 1.0E-4 for both programs.

4.10 Discussions

From the simple truss arch problem, it is found that the Bernoulli-Euler frame element can also handle a snap-through problem. The result is very close to that

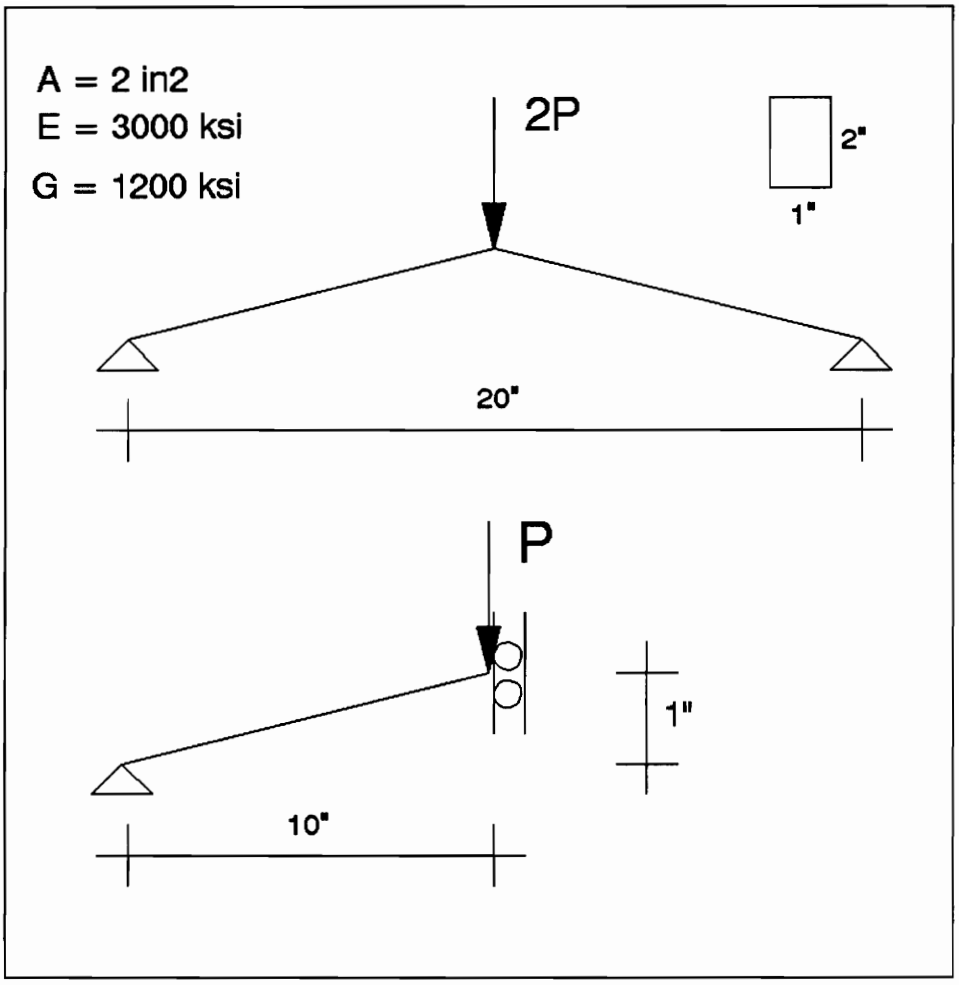


Fig. 4.5 Simple Rigid Arch Model

of the truss element discussed in Chapter 3, the analytical solution, and those obtained using ABAQUS (Fig. 4.6).

For the rigid jointed arch problem, Fig. 4.7 shows that the output from ABAQUS and from the plane frame program are very close, especially for two element mesh (Fig. 4.8).

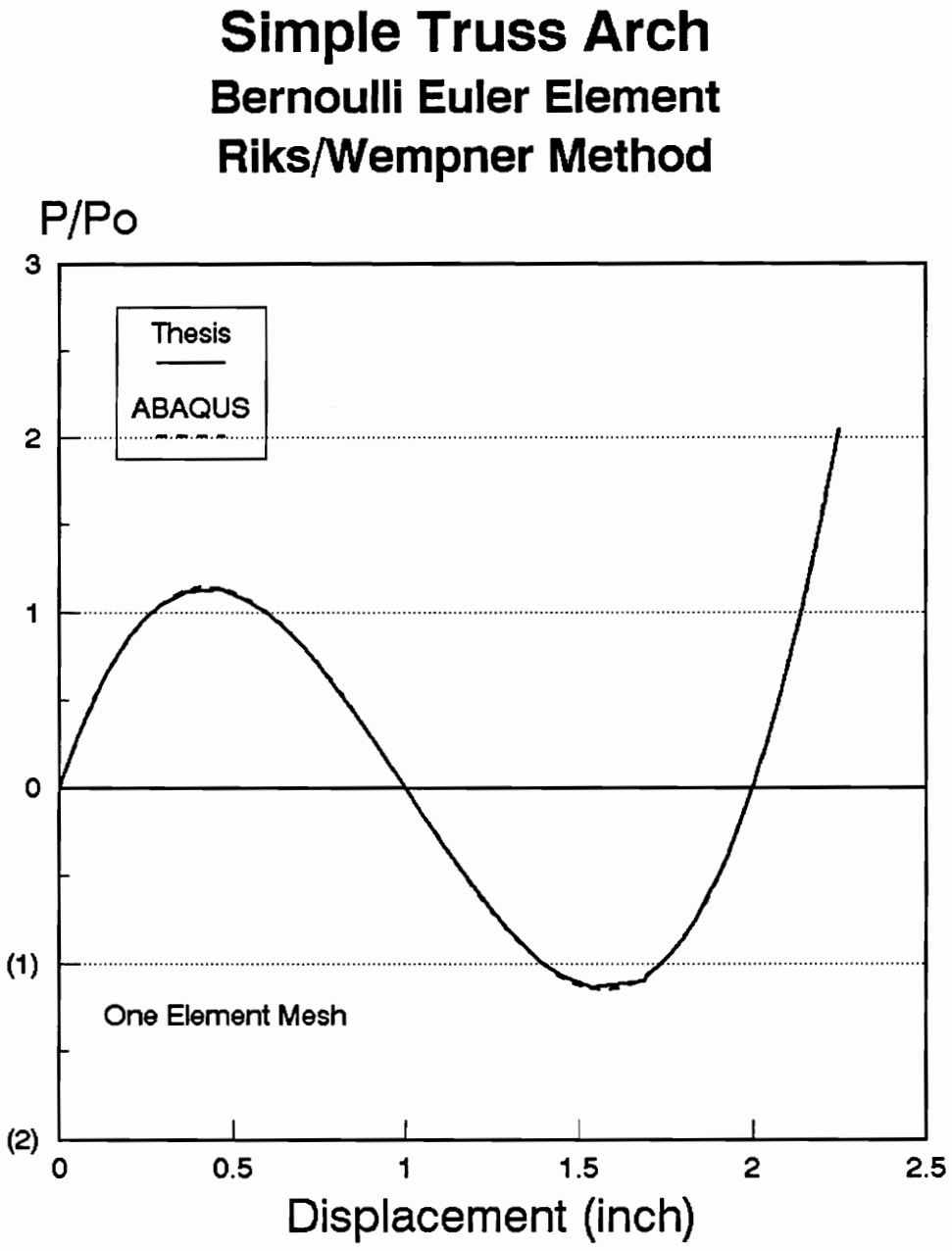


Fig. 4.6 Simple Truss Arch Problem

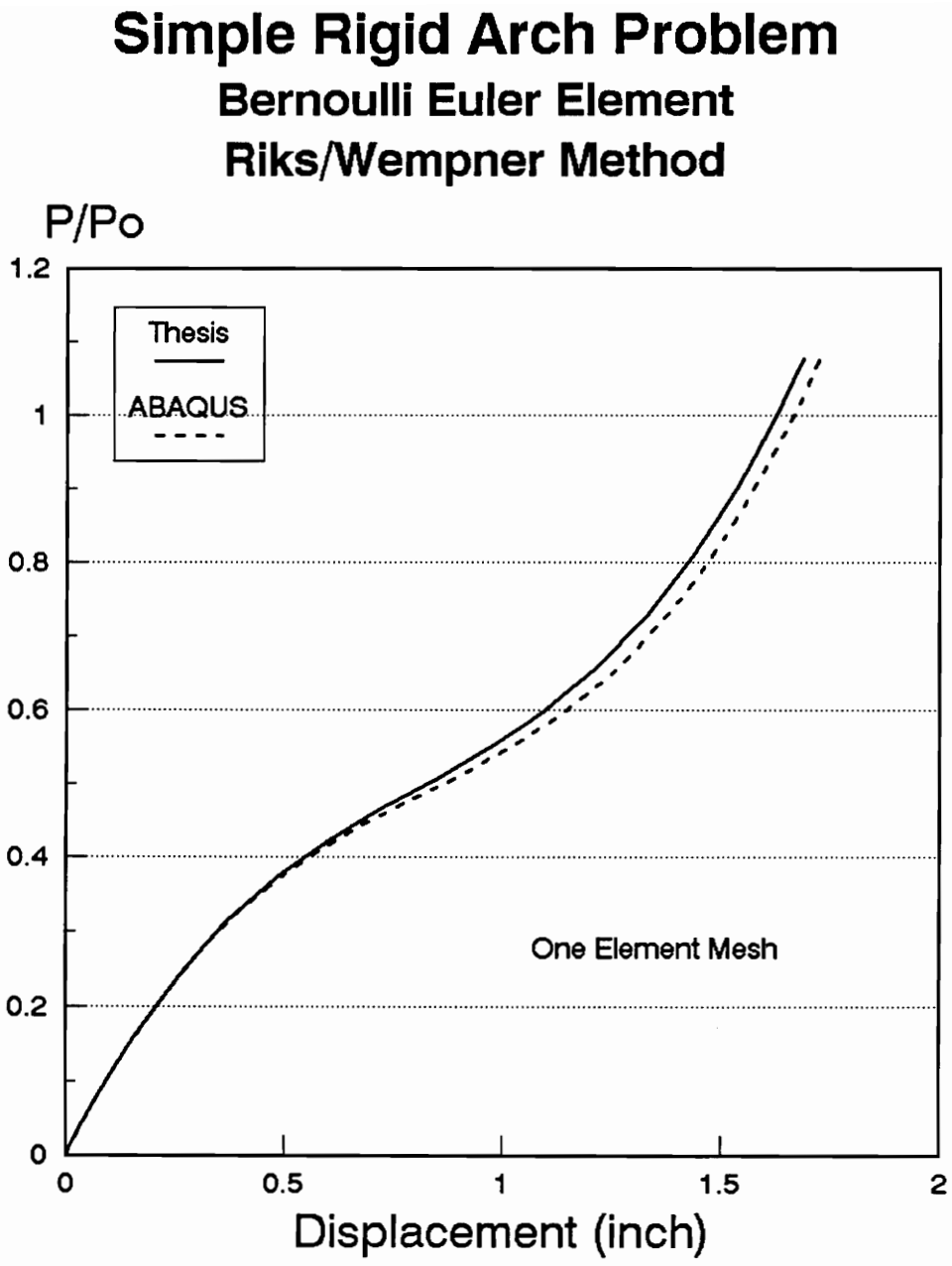


Fig. 4.7 Simple Rigid Arch Problem

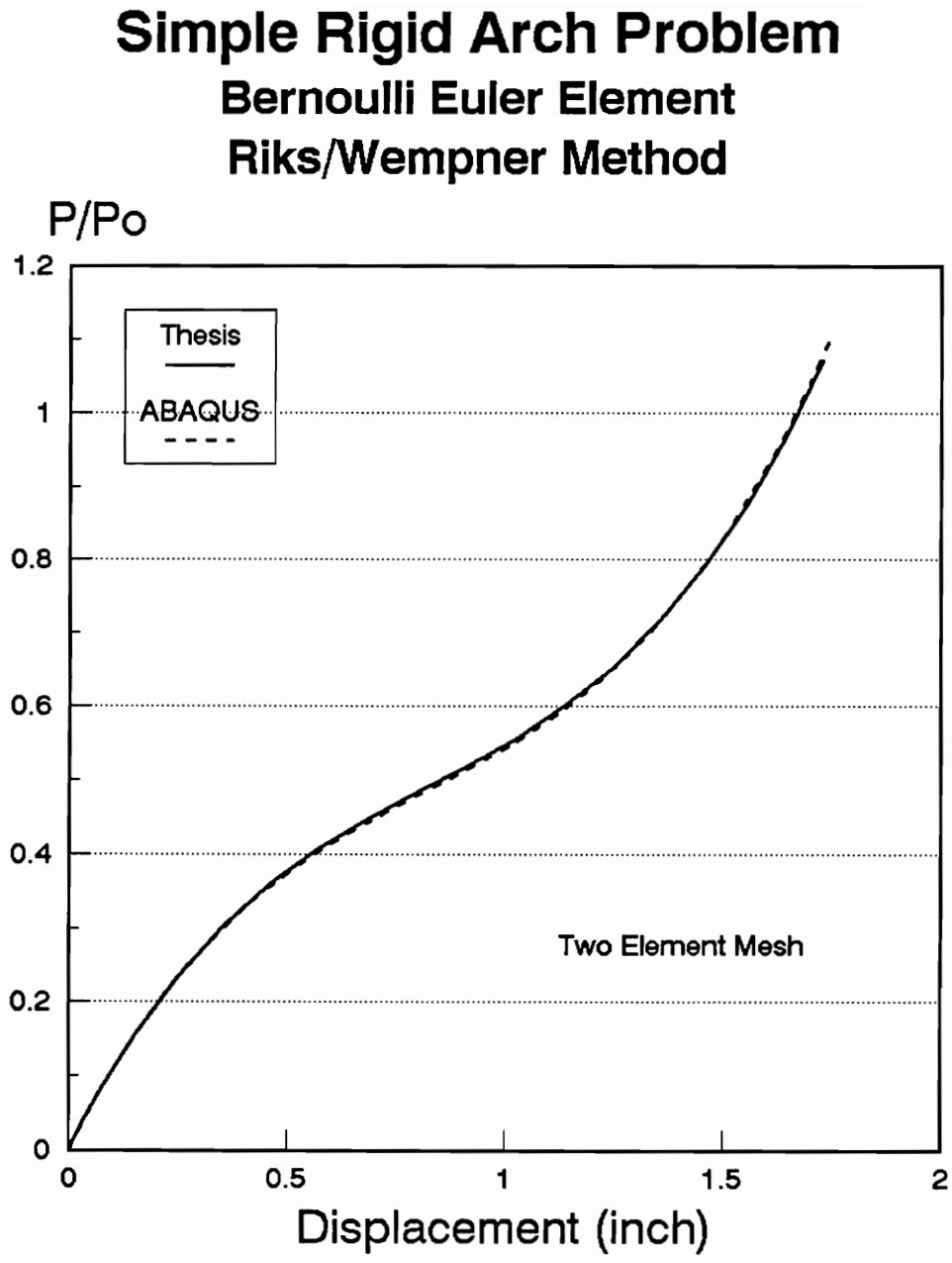


Fig. 4.8 Simple Rigid Arch Problem

Chapter 5
Mindlin Frame Element

5.1 Introduction

In the previous chapter, the Bernoulli-Euler beam element is used for modelling a plane frame structure, and does not include shear deformations in the formulation. In this chapter a 3-node Mindlin beam element will be used for modelling a plane frame structure. It has quadratic interpolation functions for axial and transverse deflections, and rotations. The shear deformations are constant across a section. This presentation is based on a paper by Wood et al. (1978) and a book by Hinton et al. (1979).

5.2 Strain-displacement relationship

Using the nonlinear relation for axial strains and the linear relation for shear strains, the nonlinear strain-displacement relations can be expressed as

$$\epsilon_x = u_{,x} + \tfrac{1}{2}(u_{,x}^2 + v_{,x}^2) \tag{5.1}$$

$$\gamma_{xy} = u_{,y} + v_{,x} \tag{5.2}$$

Where ϵ_x is the axial strain, γ_{xy} is the transverse shear strain, u is the axial deflection and v is the transverse deflection.

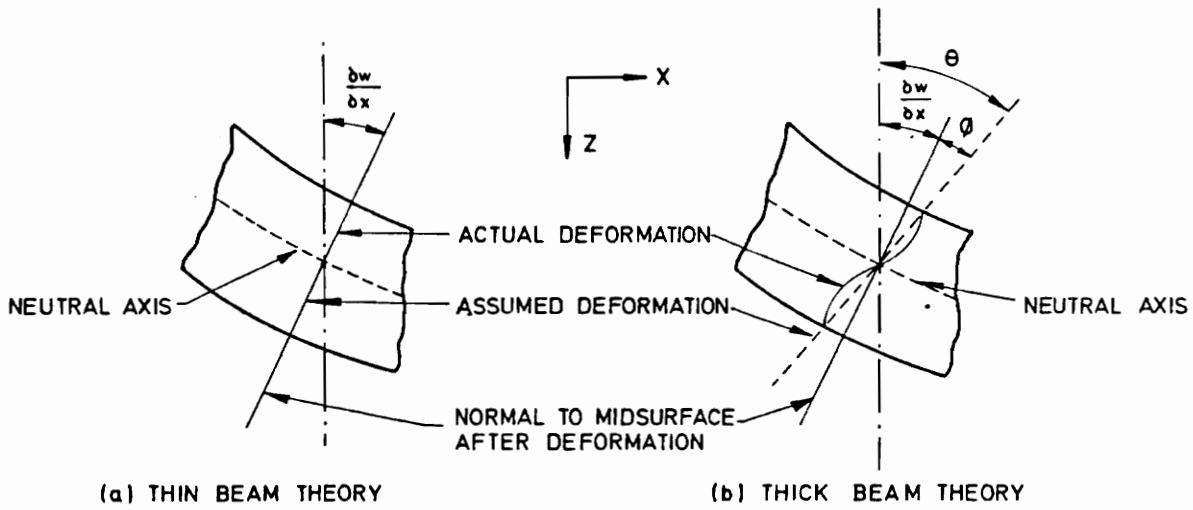


Fig.5.1 Thick Beam Theory
(Bathe, 1982)

By using u as the axial deflection at the mid-surface and assuming that a plane section will remain plane, but not necessarily normal to the mid-surface (Fig. 5.1), we get

$$u(x,y) = u - y \cdot \theta \tag{5.3}$$

$$\epsilon_x(x,y) = (u - y \cdot \theta_{,x}) + \tfrac{1}{2}((u - y \cdot \theta_{,x})^2 + v_{,x}^2) \tag{5.4}$$

$$\gamma_{xy}(x,y) = (-\theta) + v_{,x} = v_{,x} - \theta \tag{5.5}$$

Where θ is the rotation of a section of the element (Fig. 5.1). The quadratic term of u in Eq. (5.4) can be neglected for a small strain problem, so

$$\epsilon_x = u - y \cdot \theta_{,x} + \tfrac{1}{2} v_{,x}^2 \tag{5.6}$$

$$\gamma_{xy} = v_{,x} - \theta \tag{5.7}$$

5.3 Strain energy

To simplify the derivation, the total strain energy U is separated into strain energy done by axial strains and shear strains as follows:

$$U = \int_{vol} \tfrac{1}{2} \cdot E \cdot \epsilon_x^2 \cdot dv + \int_{vol} \tfrac{1}{2} \cdot Gs \cdot \gamma^2_{xy} \cdot dv \tag{5.8}$$

The shear deformations usually vary across a section. Because we assume a constant shear deformation across a section, we need to correct the strain energy done by the shear deformation (Bathe, 1980). By introducing k_f as a correction factor, A as the section area, I as the moment of inertia of the section, E as the

modulus of elasticity, Gs as the shear modulus, and by using Eqs. (5.7)-(5.8) and

$$\int_A dA = Ag, \quad \int_A y \cdot dA = 0, \quad \int_A y^2 \cdot dA = I, \quad \int_A E \cdot u_{,x} \cdot dA = P \tag{5.9}$$

we get

$$U = \frac{EAg}{2} \int_0^L u_{,x}{}^2 \cdot dx + \frac{EI}{2} \int_0^L \theta_{,x}{}^2 \cdot dx + \frac{P}{2} \int_0^L v_{,x}{}^2 \cdot dx + \frac{GsAg}{2k_f} \int_0^L \gamma_{xy}{}^2 \cdot dx \tag{5.10}$$

5.4 Interpolation functions

The 3-node isoparametric Mindlin element has quadratic interpolation functions for coordinates and deflections (Fig. 5.2). The interpolation functions for coordinates and rotations are

$$x = \sum_{i=1}^{3} N_i \cdot x_i \qquad y = \sum_{i=1}^{3} N_i \cdot y_i \qquad \theta = \sum_{i=1}^{3} N_i \cdot \theta_i \tag{5.11}$$

where

$$N_1 = \tfrac{1}{2}\xi(\xi-1) \qquad N_2 = \tfrac{1}{2}\xi(\xi+1) \qquad N_3 = \tfrac{1}{2}(1-\xi^2) \tag{5.12}$$

For an isoparametric element, the interpolation functions for coordinates and deflections are identical, so

$$u = \sum_{i=1}^{3} N_i \cdot u_i \qquad v = \sum_{i=1}^{3} N_i \cdot v_i \qquad \theta = \sum_{i=1}^{3} N_i \cdot \theta_i \tag{5.13}$$

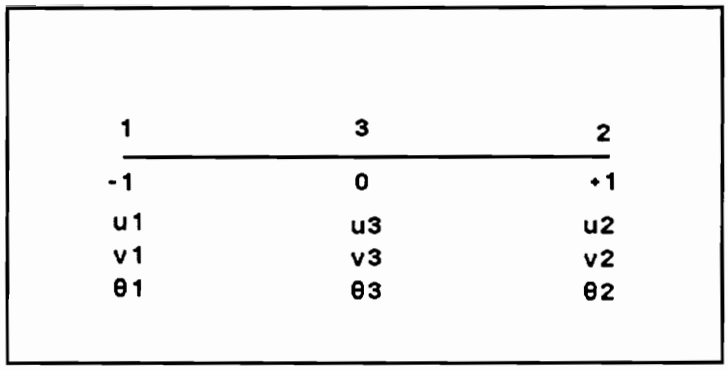


Fig 5.2 Mindlin Frame Element

Lagrangian 3-node Finite Element

5.5 Strain-displacement matrices

The Green-Lagrange strains can be expressed in terms of the interpolation functions defined above. The strains are separated into four parts as follows.

For linear axial strain,

$$\epsilon_x = u_{,x} = B_a \cdot d \tag{5.14}$$

where

$$B_a = \begin{bmatrix} N_{1,x} & 0 & 0 & N_{2,x} & 0 & 0 & N_{3,x} & 0 & 0 \end{bmatrix} \tag{5.15}$$

$$d = \{u_1 \;\; v_1 \;\; \theta_1 \;\; u_2 \;\; v_2 \;\; \theta_2 \;\; u_3 \;\; v_3 \;\; \theta_3\} \tag{5.16}$$

For linear axial strain caused by bending,

$$\epsilon_x = -y \cdot \theta_{,x} = -y \cdot B_\theta \cdot d \tag{5.17}$$

where

$$B_\theta = \begin{bmatrix} 0 & 0 & N_{1,x} & 0 & 0 & N_{2,x} & 0 & 0 & N_{3,x} \end{bmatrix} \tag{5.18}$$

For nonlinear axial strain,

$$\epsilon_x = \tfrac{1}{2} v^2_{,x} = \tfrac{1}{2} (B_v \cdot d)^T (B_v \cdot d) \tag{5.19}$$

where

$$B_v = \begin{bmatrix} 0 & N_{1,x} & 0 & 0 & N_{2,x} & 0 & 0 & N_{3,x} & 0 \end{bmatrix} \tag{5.20}$$

And for shear strain,

$$\gamma_{xy} = v_{,x} - \theta = B_v \cdot d - N_\theta \cdot d = (B_v - N_\theta)\, d \tag{5.21}$$

where

$$N_\theta = \begin{bmatrix} 0 & 0 & N_1 & 0 & 0 & N_2 & 0 & 0 & N_3 \end{bmatrix} \tag{5.22}$$

5.6 Tangent Stiffness Matrix

The total strain energy of the element can be expressed as

$$U = \frac{1}{2} d^T \cdot k_a \cdot d + \frac{1}{2} d^T \cdot k_b \cdot d + \frac{1}{2} d^T \cdot k_\sigma \cdot d + \frac{1}{2} d^T \cdot k_v \cdot d \tag{5.23}$$

where,

$$k_a = EAg \int_0^L B_a{}^T \cdot B_a \cdot dx = \frac{EAg\, L}{2} \int_{-1}^{1} B_a{}^T \cdot B_a \cdot d\xi \tag{5.24}$$

$$k_b = EI \int_0^L B_\theta{}^T \cdot B_\theta \cdot dx = \frac{EIL}{2} \int_{-1}^{1} B_\theta{}^T \cdot B_\theta \cdot d\xi \tag{5.25}$$

$$k_\sigma = P \int_0^L B_v{}^T \cdot B_v \cdot dx = \frac{PL}{2} \int_{-1}^{1} B_v{}^T \cdot B_v \cdot d\xi \tag{5.26}$$

$$k_v = \frac{Gs\, Ag}{k_f} \int_0^L (B_v - N_\theta)^T \cdot (B_v - N_\theta) \cdot dx = \frac{Gs\, Ag\, L}{2k_f} \int_{-1}^{1} (B_v - N_\theta)^T \cdot (B_v - N_\theta) \cdot d\xi \tag{5.27}$$

Thus,

$$U = \frac{1}{2} d^t\, k_t\, d \tag{5.28}$$

where the element tangent stiffness matrix k_t is

$$k_t = k_a + k_b + k_\sigma + k_v \tag{5.29}$$

This k_t is only an approximation of the real tangent stiffness matrix because several nonlinear terms have been neglected in the nonlinear strain-displacement relations. The simplification can be justified because k_t is only used for predicting the solution, which will be corrected by checking the unbalanced load vector.

5.6.1 Local tangent stiffness matrix

By expanding Eqs. (5.24)-(5.27) into matrix forms and performing numerical integrations to find the entries of the matrices, we get explicit expressions for all terms of the local tangent stiffness matrix:

$$\mathbf{k_a} = \frac{EAg}{3L}\begin{bmatrix} 7 & 0 & 0 & 1 & 0 & 0 & -8 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 7 & 0 & 0 & -8 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -8 & 0 & 0 & -8 & 0 & 0 & 16 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.30}$$

$$\mathbf{k_b} = \frac{EI}{3L}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 7 & 0 & 0 & 1 & 0 & 0 & -8 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 7 & 0 & 0 & -8 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -8 & 0 & 0 & -8 & 0 & 0 & 16 \end{bmatrix} \tag{5.31}$$

$$k_\sigma = \frac{P}{6L}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 7 & 0 & 0 & 1 & 0 & 0 & -8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 7 & 0 & 0 & -8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -8 & 0 & 0 & -8 & 0 & 0 & 16 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.32}$$

$$k_v = \frac{Gs\ Ag}{30k_fL}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 70 & 15L & 0 & 10 & -5L & 0 & -80 & 20L \\ 0 & 15L & 4L^2 & 0 & 5L & -L^2 & 0 & -20L & 2L^2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 5L & 0 & 70 & -15L & 0 & -80 & -20L \\ 0 & -5L & -L^2 & 0 & -15L & 4L^2 & 0 & 20L & 2L^2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -80 & -20L & 0 & -80 & 20L & 0 & 160 & 0 \\ 0 & 20L & 2L^2 & 0 & -20L & 2L^2 & 0 & 0 & 16L^2 \end{bmatrix} \tag{5.33}$$

5.6.2 Global tangent stiffness matrix

The global tangent stiffness matrix is calculated by transforming the local tangent stiffness matrix from the local axis to the global axis as follows:

$$K_t = \Lambda^T k_t \Lambda \tag{5.34}$$

where K_t is the 9x9 global tangent stiffness matrix, and Λ is the 9x9 transformation matrix consists of three 3x3 submatrices for each node as follows:

$$\Lambda = \begin{bmatrix} \mathbf{R} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{R} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R} \end{bmatrix} \tag{5.35}$$

where

$$\mathbf{R} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.36}$$

5.7 Internal forces

The element forces calculation is based on the method used by Katzenberger (1983), where the total deflections are separated into the rigid body displacement of the element and the local deflections measured from the current configuration of the element (Fig. 5.3).

First, the element is moved by a rigid body motion so that the left node is equal to the updated left node position. Then, the element is rotated by an angle $\Delta\theta$ so that the other end node will be at its current location (Fig. 5.3).

After that, the current total deflections must be referred to the new local axes. The total rotation of each node must be corrected by subtracting from them $\Delta\theta$. The axial deflection is equal to zero at left node, and equal to the total element elongation at the right node (Fig. 5.4).

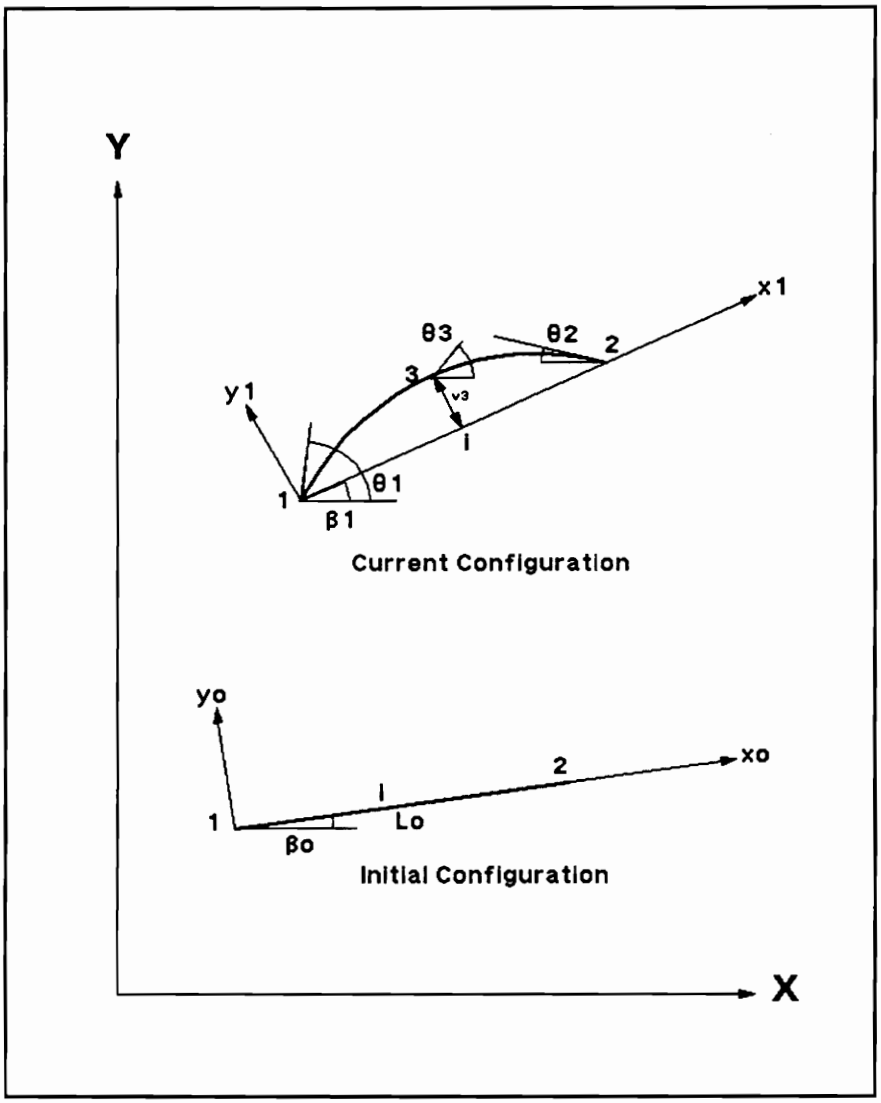


Fig. 5.3 Mindlin Frame Element in Global Coordinates

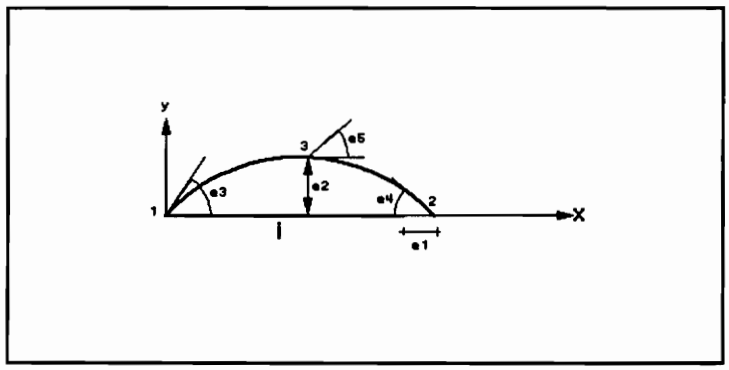


Fig 5.4 Mindlin Frame Element in Local Coordinates

Using this approach, we will get five independent deflections or natural deflections e_1, e_2, e_3, e_4, and e_5. The interpolation functions for the deflections in the x and y directions and the rotation are

$$\begin{bmatrix} u \\ v \\ \theta \end{bmatrix} = \begin{bmatrix} N_1 & 0 & 0 & 0 & 0 \\ 0 & N_2 & 0 & 0 & 0 \\ 0 & 0 & N_3 & N_4 & N_5 \end{bmatrix} \cdot \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \end{bmatrix} \tag{5.37a}$$

or

$$u = N \cdot d \tag{5.37b}$$

where

$$N_1 = \tfrac{1}{2}(1+\xi) \tag{5.38a}$$

$$N_2 = (1-\xi^2) \tag{5.38b}$$

$$N_3 = \tfrac{1}{2}\xi(\xi-1) \tag{5.38c}$$

$$N_4 = \tfrac{1}{2}\xi(\xi+1) \tag{5.38d}$$

$$N_5 = (1-\xi^2) \tag{5.38e}$$

and

$$e_1 = \Delta L = L - L_o \tag{5.39a}$$

$$e_2 = v_3 \tag{5.39b}$$

$$e_3 = \theta_1 - \Delta\theta \tag{5.39c}$$

$$e_4 = \theta_2 - \Delta\theta \tag{5.39d}$$

$$e_5 = \theta_3 - \Delta\theta \tag{5.39e}$$

The parameter $\Delta\theta$ and ΔL can be calculated easily using vector operations involving the initial and current configurations of the element. The sign of $\Delta\theta$ is ositive if the cross product of the initial local axis vector and the updated local axis vector gives the outward vector according to the right hand rule. The local transverse deflection v_3 can also be calculated by vector operations.

The secant stiffness matrix of the element in local coordinates can be found by using the and Schrefler approach (1978). The axial strain is separated into the linear axial strain ϵ_o and the nonlinear axial strain ϵ_l. Only the linear part of the shear strain is used in the formulation. The strain-displacement relations of the element are

$$\epsilon_x = \epsilon_o + \epsilon_l = u_{,x} - y \cdot \theta_{,x} + \tfrac{1}{2} v_{,x}^2 \tag{5.40}$$

$$\gamma_{xy} = \gamma_o = v_{,x} - \theta \tag{5.41}$$

Using B_{o1} and B_l as the strain-displacement operators for linear and nonlinear axial strains respectively, B_{o2} as the linear shear strain-displacement operator, and d as the deflection vector, we get

$$\epsilon_x = B_{o1} \cdot d + \tfrac{1}{2} B_l \cdot d = (B_{o1} + \tfrac{1}{2} B_l)\, d \tag{5.42}$$

$$\gamma_{xy} = B_{o2} \cdot d \tag{5.43}$$

Where

$$B_{o1} = \frac{1}{L} \begin{bmatrix} 1 & 0 & -y(2\xi-1) & -y(2\xi+1) & 4y\xi \end{bmatrix} \tag{5.44}$$

$$B_{o2} = \frac{1}{L} \begin{bmatrix} 0 & -4\xi & \frac{-L}{2}\xi(\xi-1) & \frac{-L}{2}\xi(\xi+1) & -L(1-\xi^2) \end{bmatrix} \tag{5.45}$$

$$B_l = \frac{1}{L} \begin{bmatrix} 0 & \frac{16\xi^2 e2}{L} & 0 & 0 & 0 \end{bmatrix} \tag{5.46}$$

The first variant of the strains are

$$\delta\epsilon_x = \delta\epsilon_o + \delta\epsilon_l = B_{o1} \cdot \delta d + B_l \cdot \delta d = (B_{o1} + B_l)\, \delta d \tag{5.47}$$

$$\delta\gamma_{xy} = \delta\gamma_o = B_{o2} \cdot \delta d \tag{5.48}$$

The unsymmetric form of the secant stiffness matrix is (Wood et al., 1978),

$$k_s = \int_v (B_o + B_l)^t\, D\, (B_o + \tfrac{1}{2} B_l)\, dv \tag{5.49}$$

By separating the axial strain and the shear strain we get

$$k_s = \int_v (B_{o1} + B_l)^t \, E \, (B_{o1} + \tfrac{1}{2} B_l) \, dv + \int_v B^t_{o2} \cdot \frac{G}{k} \cdot B_{o2} \, dv = k_{sb} + k_{sv} \qquad (5.50)$$

By substituting Eqs. (5-45) - (5-47) into Eq. (5-51) and integrating numerically Eq. (5-51), we get all entries of the 5x5 unsymmetric secant stiffness matrix. This secant stiffness matrix can then be used to calculate the element forces with respect to the current configuration. Although only five forces can be calculated directly, the others can be easily found by equilibrium. Thus

$$k_{sb} = \frac{E}{L^2} \begin{bmatrix} Ag\,L & \frac{8Ag\,e2}{3} & 0 & 0 & 0 \\ \frac{16Ag\,e2}{3} & \frac{128Ag\,e2^2}{5L} & 0 & 0 & 0 \\ 0 & 0 & \frac{7IL}{3} & \frac{IL}{3} & \frac{-8IL}{3} \\ 0 & 0 & \frac{IL}{3} & \frac{7IL}{3} & \frac{-8IL}{3} \\ 0 & 0 & \frac{-8IL}{3} & \frac{-8IL}{3} & \frac{16IL}{3} \end{bmatrix} \qquad (5.51)$$

$$k_{sv} = \frac{Gs\,Ag}{2k_f L} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{32}{3} & \frac{-4L}{3} & \frac{4L}{3} & 0 \\ 0 & \frac{-4L}{3} & \frac{8L^2}{30} & \frac{-2L^2}{30} & \frac{4L^2}{30} \\ 0 & \frac{4L}{3} & \frac{-2L^2}{30} & \frac{8L^2}{30} & \frac{4L^2}{30} \\ 0 & 0 & \frac{4L^2}{30} & \frac{4L^2}{30} & \frac{32L^2}{30} \end{bmatrix} \qquad (5.52)$$

And the local element forces w.r.t. the current configuration can be calculated by

$$f = k_s \cdot d \qquad (5.53)$$

5.8 Program development

Based on above derivations, a nonlinear frame analysis program using Mindlin elements has been developed. Except the difference in node numbers, the program looks very close to the program based on Bernoulli-Euler element presented in chapter 4.

The program developed here is based on the linear frame analysis program by Holzer (1985). The nonlinear version of the program is created by adding the following subroutines to the linear version :

1. LINEAR - Linear analysis subroutine
2. NONLIN - Nonlinear analysis subroutine
3. NEWRAP - Newton-Raphson Method subroutine
4. RIKWEM - Riks-Wempner Method subroutine
5. UNBALF - Unbalanced Forces subroutine
6. UPDATC - Updating Coordinate subroutine
7. CONVER - Convergence Checking subroutine
8. NSTIFF - Nonlinear Stiffness Matrix subroutine
9. NELEMF - Nonlinear Element Forces subroutine
10. LDISP - Local deflection subroutine

Basically, the program is divided into two parts, linear and nonlinear analysis, by the LINEAR and NONLIN subroutines. The LINEAR subroutine is exactly same as the second part of MAIN subroutine from the original linear version. The NONLIN subroutine is analog to the LINEAR, except the linear equation solver is replaced by REWRAP or RIKWEM subroutines.

The NEWRAP subroutine uses the Newton-Raphson algorithm. It uses incremental analysis using variable load step and updates the tangent stiffness matrix at each iteration.

The RIKWEM subroutine is based on the modified Riks-Wempner algorithm by Holzer et al. (1981). It uses incremental analysis using variable path length.

The UNBALF subroutine contains the calculation of the unbalanced force vector. It calls the NELEMF for calculating the nonlinear element forces w.r.t. the local axis and the FORCE subroutine to transform the local element forces to the global joint forces before calculating the unbalanced force vector.

The UPDATC subroutine updates the coordinates of the structure for each iteration. The other parameters such as length and cosines directions are calculated on an element basis when they are needed. The updated coordinates are stored at different arrays because the initial coordinates are needed for the total deflection calculation.

The NSTIFF subroutine calculates the nonlinear/tangent stiffness matrix for each element. It uses the axial forces from the previous iteration to calculate the tangent stiffness matrix w.r.t. local axis. The local stiffness matrix is then transformed to the global coordinates using the current direction cosines.

The NELEMF subroutine calculates the total nonlinear internal forces using a secant stiffness matrix method. The total deflection vector, needed for the calculations, is taken from the LDISP subroutine.

5.9 Test problems

The nonlinear Mindlin frame element is verified by testing it with some test problems, and by comparing it with a commercial program for large nonlinear analysis, ABAQUS. The verification of the nonlinear Mindlin frame element is done as follows.

First, the element is tested for its capability to model shear deformations. A simple cantilever problem is analyzed linearly with different number of element to find the accuracy of the Mindlin element (Fig. 5.5).

Second, the element is tested for its capability to model a simple truss arch problem to find out the capability of the element to handle a snap-through. So, only the nonlinear truss modelling capability is tested here (Fig. 3.4).

Third, the element is tested for a simple rigid arch model. To avoid the shear locking effects (Bathe, 1981), the cross section height to width ratio of elements used for this test problem is limited to 2 (Fig. 5.6).

All test problems were analyzed using the modified Riks-Wempner methods with variable step length. To be able to compare the results with those from ABAQUS, the force and moment tolerances are set to 1.0E-4 for both programs.

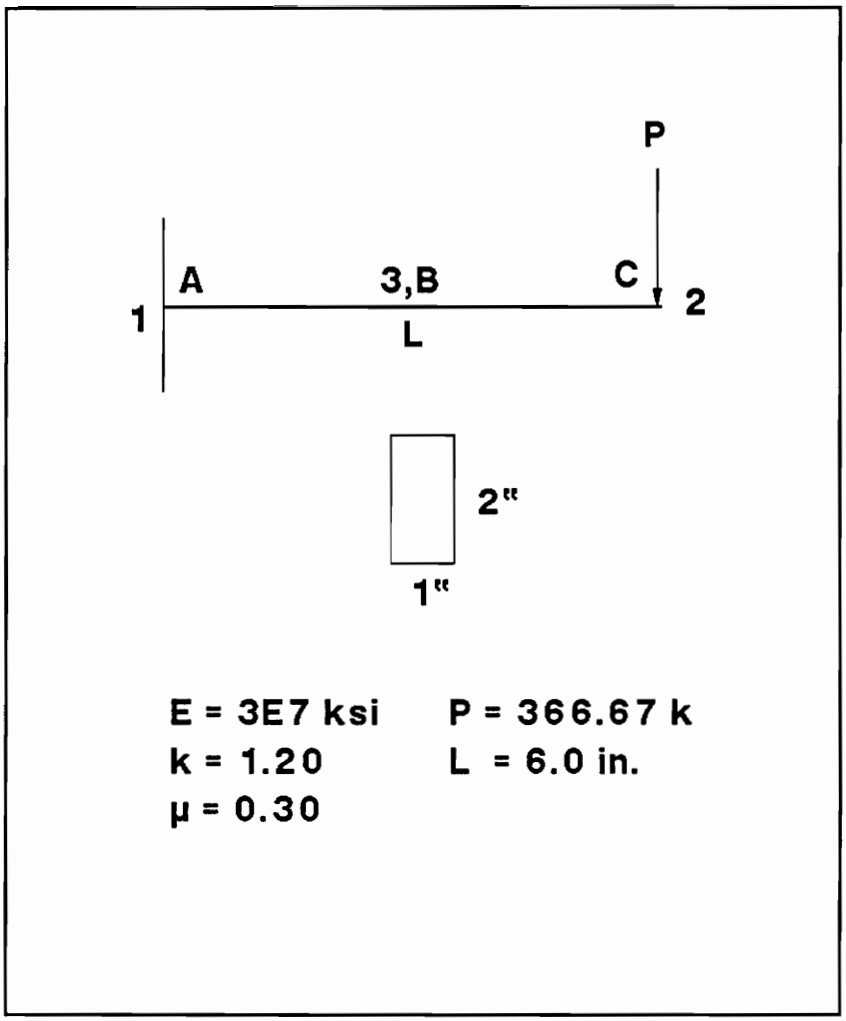


Fig. 5.5 Cantilever Problem

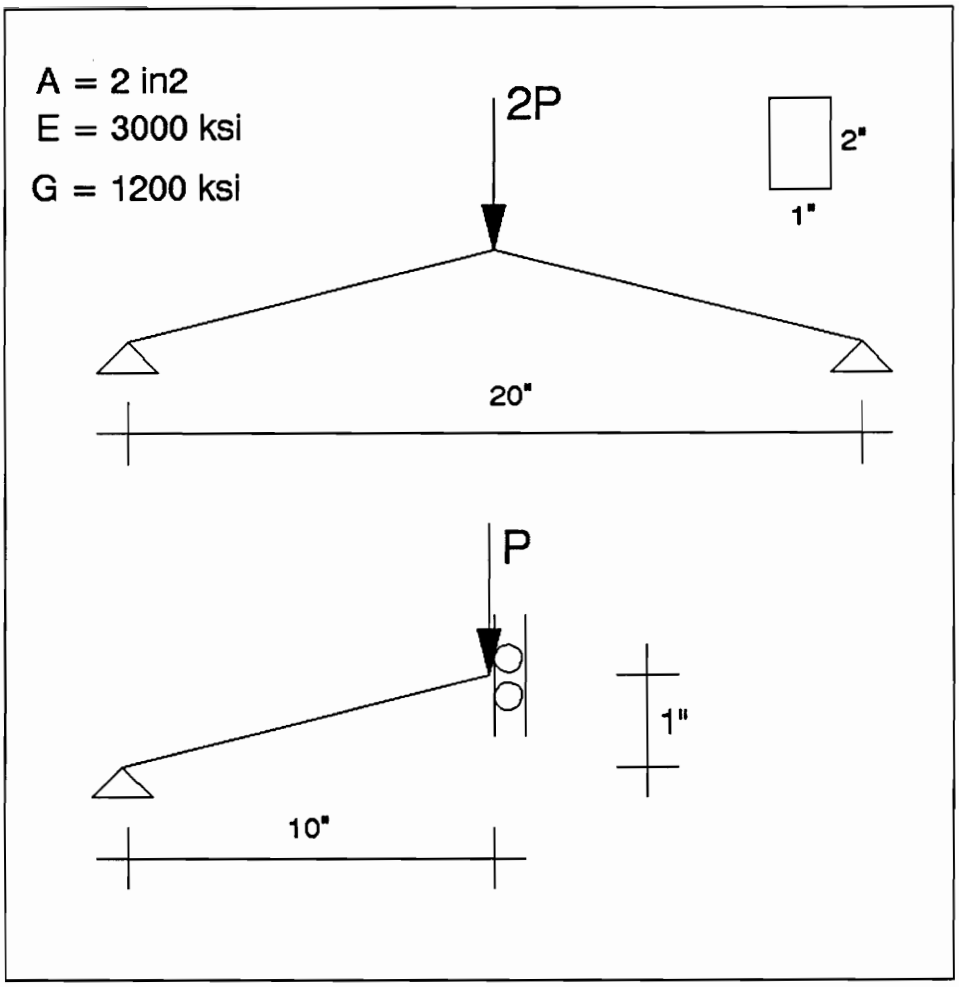


Fig. 5.6 Simple Rigid Arch Model

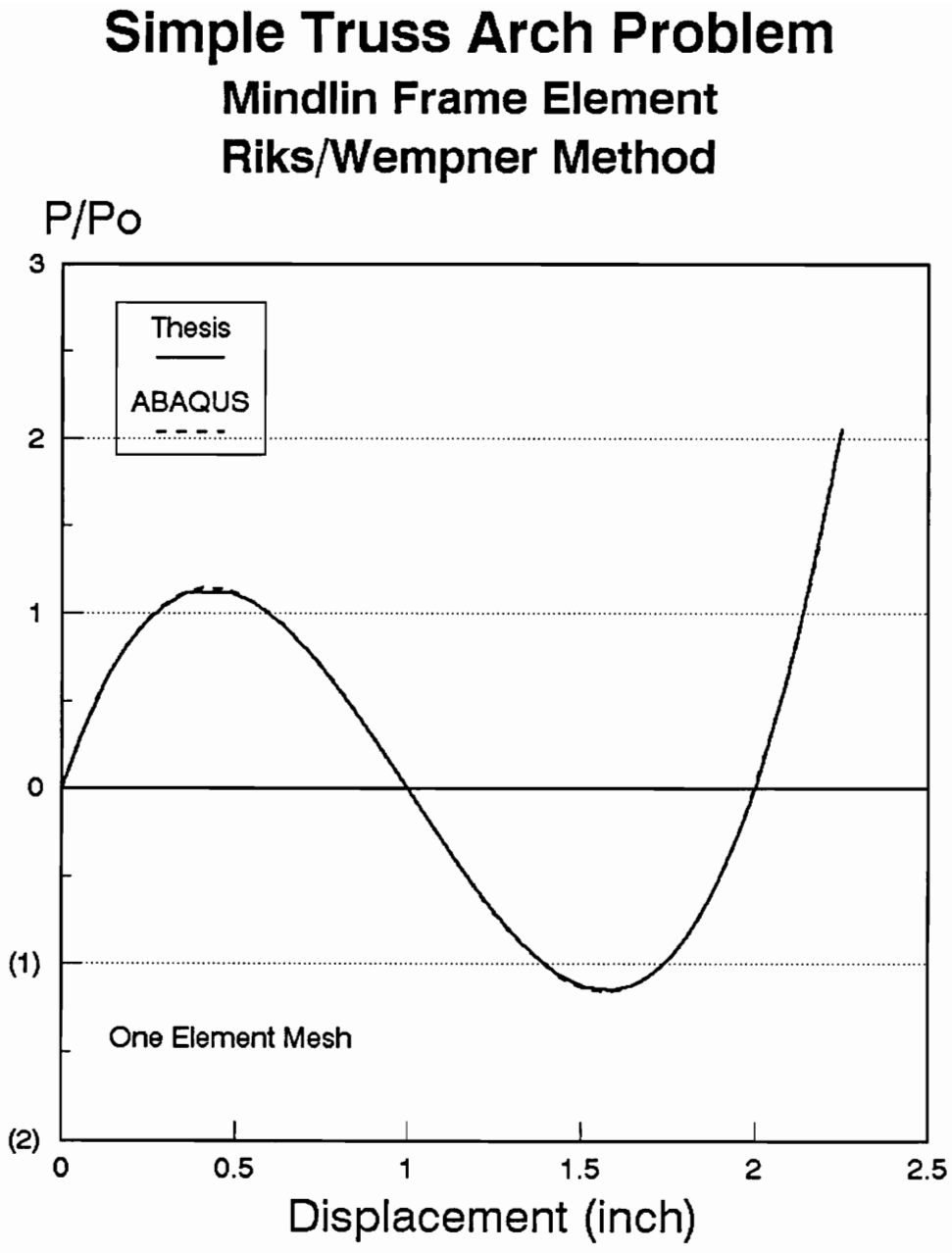


Fig. 5.7 Simple Truss Arch Problem

5.10 Discussions

The cantilever problem is analyzed using different number of elements. Comparisons are made using results from the Bernoulli-Euler frame element and from ABAQUS. Three elements from ABAQUS are used for this purpose: the 3-node Mindlin frame element (B22), the 8-node constant plane stress element (CPS8), and the 2-node Bernoulli-Euler frame element (B23). The results are presented in the following tables.

For the cantilever problem, A is the fixed end, B is the middle point, and C is the free end. R is the vertical reactions, and M is the Moment reactions. V is the transverse deflection, and θ is the rotation.

Table 5.1 Cantilever Problem, Mindlin 3-Node Element

No. of element	v_b	Θ_b	v_c	Θ_c
1	-0.00040933	-0.00021732	-0.00131367	-0.0003300
2	-0.00046450	-0.00024750	-0.00142400	-0.0003300
4	-0.00046934	-0.00024750	-0.00143368	-0.0003300
8	-0.00046968	-0.00024750	-0.00143435	-0.0003300

Note: All reactions $R_a = 366.6667$ and $M_a = 2200.00$

Table 5.2 Cantilever Problem, Bernoulli Euler 2-Node Element

No. of element	v_b	Θ_b	v_c	Θ_c
1	N/A	N/A	-0.00132000	-0.0003300
2	-0.00041250	-0.00024750	-0.00132000	-0.0003300
4	-0.00041250	-0.00024750	-0.00132000	-0.0003300

Note: All reactions $R_a = 366.6667$ and $M_a = 2200.00$

Table 5.3 Cantilever Problem, ABAQUS Ver. 4.8 and Analytical

Type of element	:	1-CPS8	2-B22	1-B23	Analytical
Tip deflection, v_c	:	-1.434E-3	-1.432E-3	-1.32E-3	-1.399445

Note: All reactions $R_a = 366.6667$ and $M_a = 2200.00$

From the cantilever problem it is found that the Mindlin element derived here is capable to model shear deformations. Because the interpolation functions for the element are quadratic functions, at least two elements are needed to get a good result, as can be seen from table 5.1.

From the simple truss arch problem, it is found that the Mindlin element can also handle a snap-through problem. The result is very close to those of the truss element, the frame element, and those of ABAQUS (Fig. 5.7).

The third test problem, a simple rigid jointed arch, is analyzed using various number of elements, and the results are compared to those of the frame element to find out the effect of shear deformations on the solution.

Fig. 5.8 shows that using only one Mindlin element is not enough. By using two elements one can get a good result (Fig. 5.9). The Mindlin element from this study can be further improved, for example, by using reduced integration and discrete Kirchoff formulation.

By comparing results from Mindlin and Bernoulli-Euler frame elements, it is found that Mindlin elements will give a more flexible structure (Fig. 5.10), although, for the test problems, the effects are not significant.

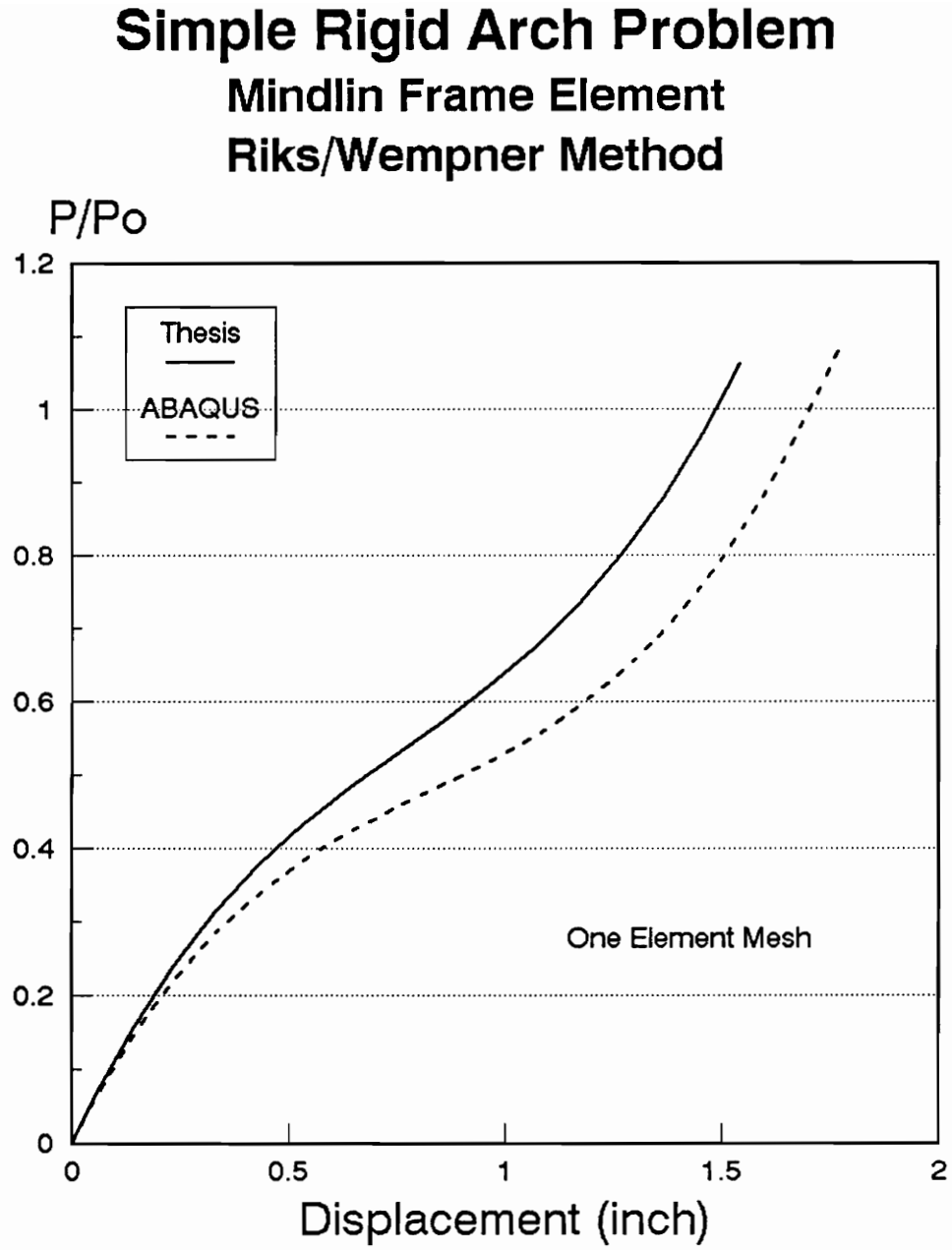


Fig. 5.8 Simple Rigid Arch Problem

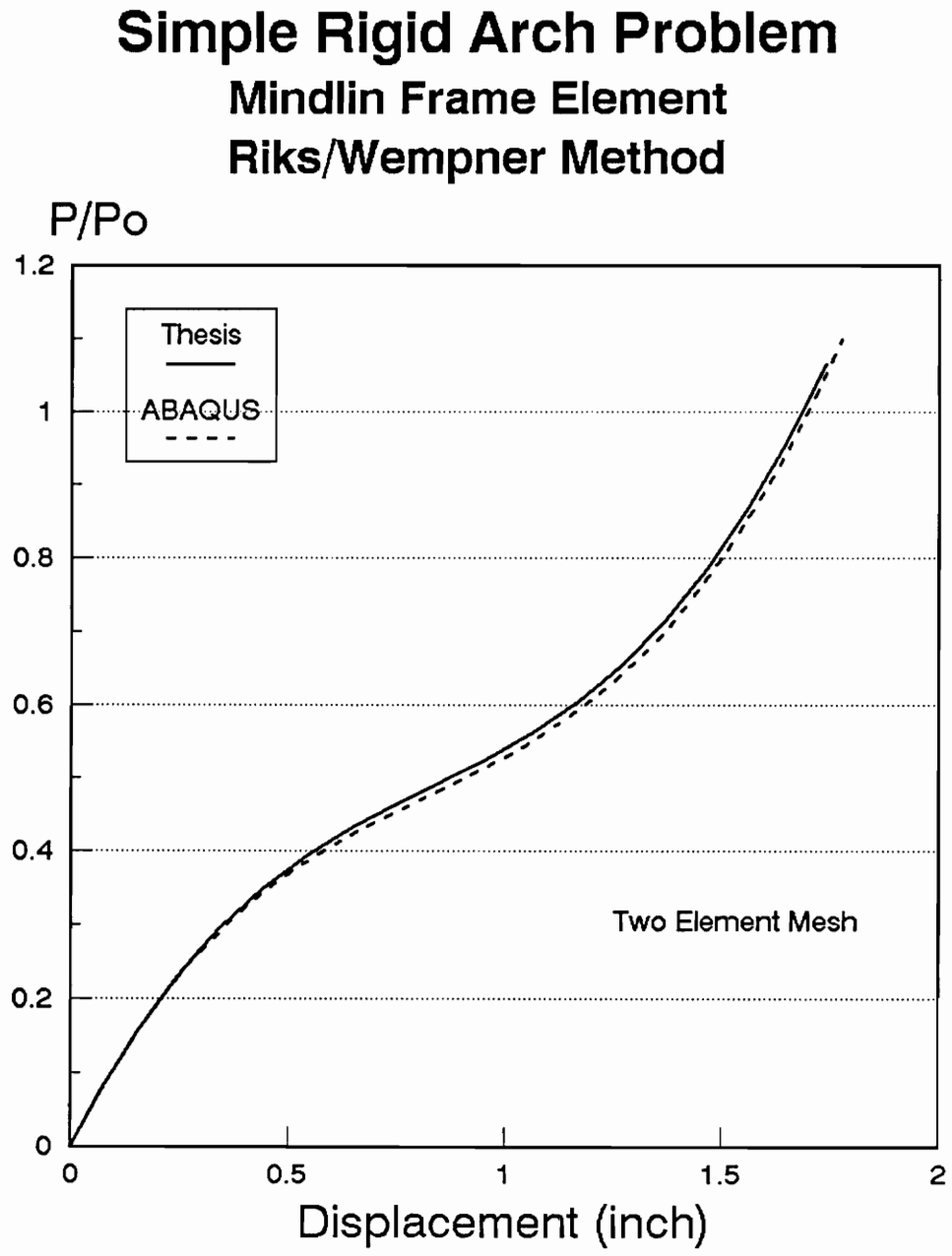


Fig. 5.9 Simple Rigid Arch Problem

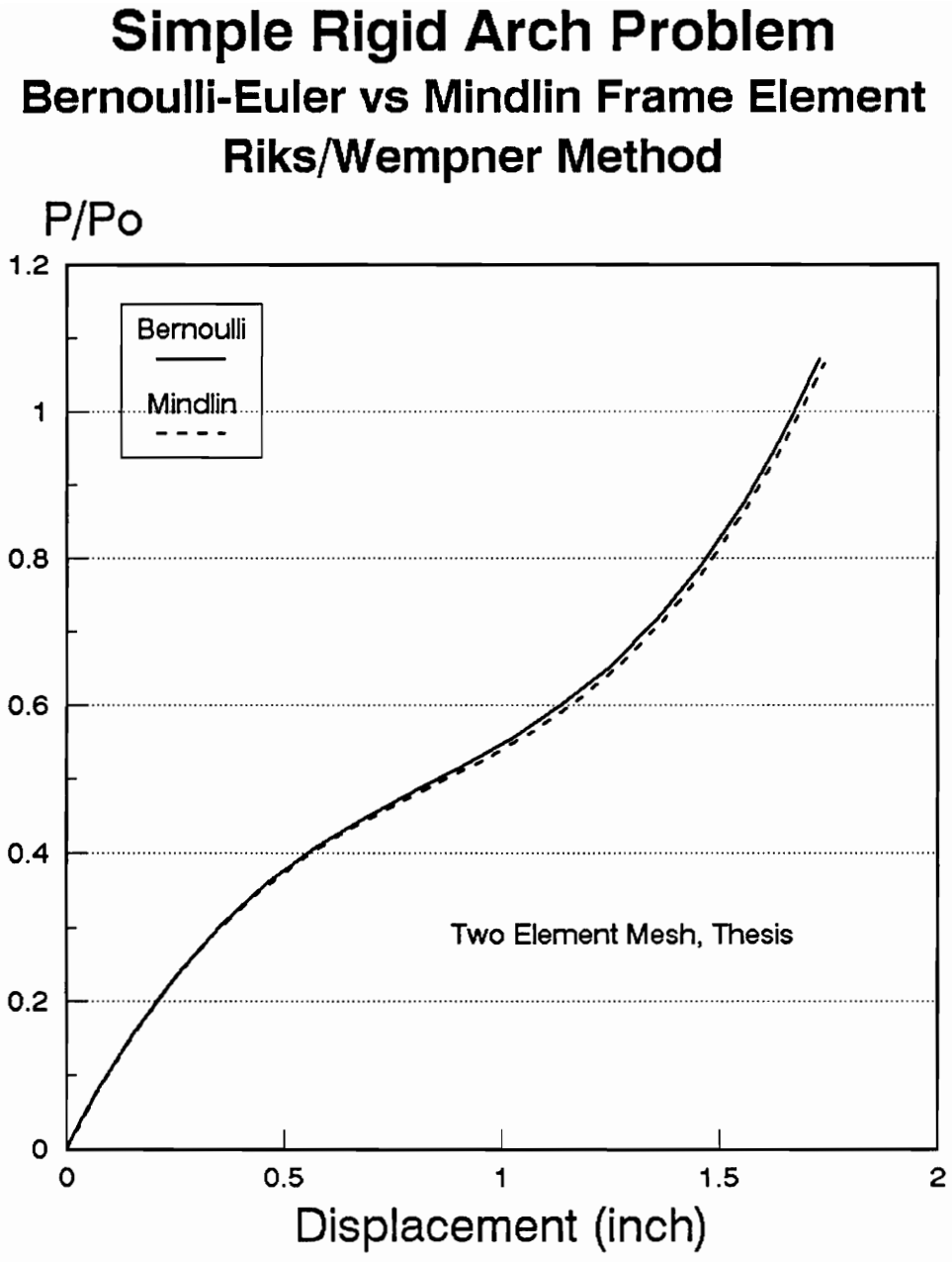


Fig. 5.10 Bernoulli-Euler vs Mindlin Element, Thesis

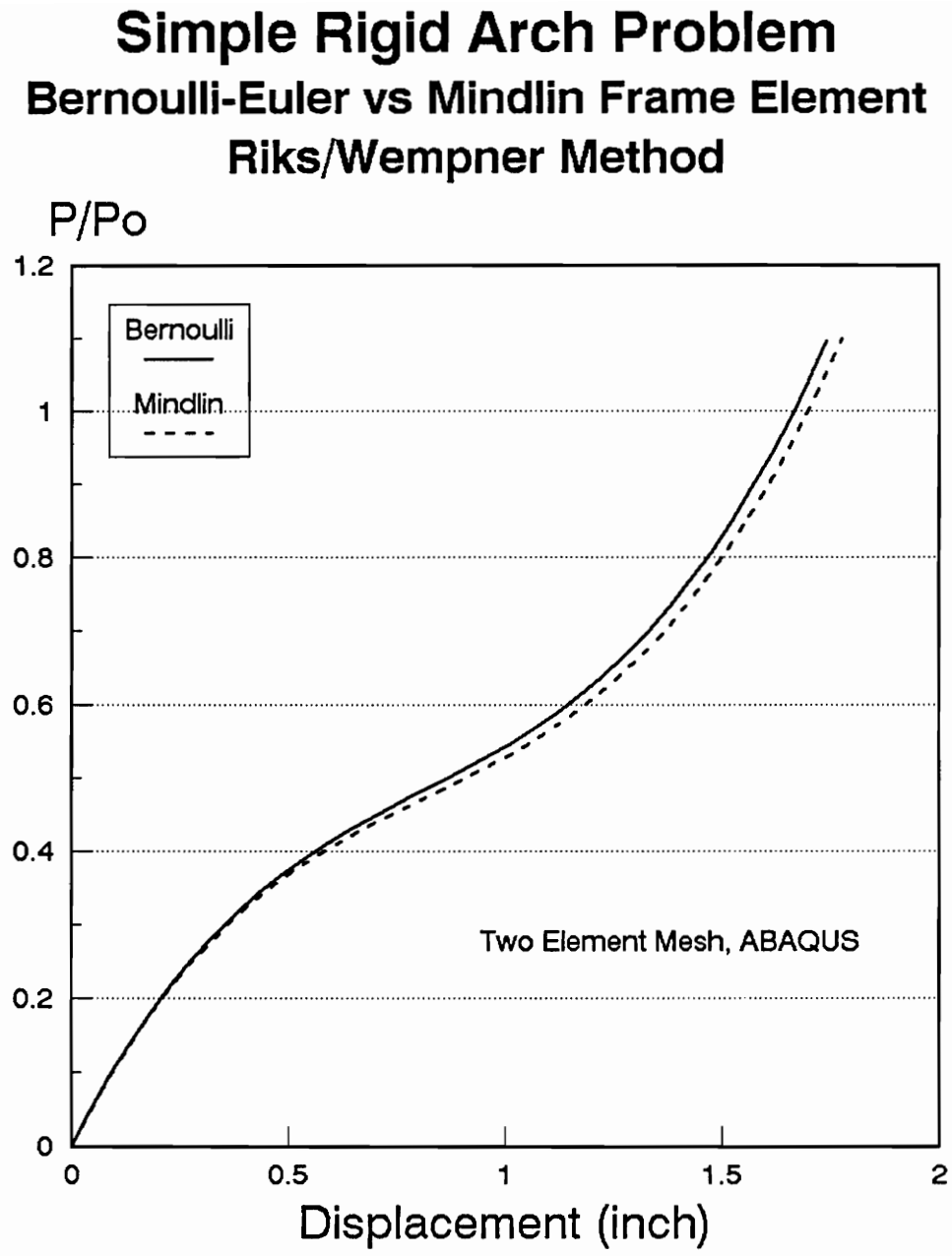


Fig. 5.11 Bernoulli-Euler vs Mindlin Frame Element, ABAQUS

Chapter 6
Solution Methods

6.1 Introduction

There are several methods available for nonlinear analysis. Some methods can only trace the equilibrium path until the first limit point is reached. Others can trace the primary equilibrium path through limit points.

In this chapter two methods of analysis, the Newton-Raphson method and the Riks-Wempner method, will be discussed. This presentation is based on a paper by Holzer et al. (1981). The NS-diagrams for both methods are given at appendix A, while the FORTRAN codes are given at appendix B.

6.2 Newton-Raphson Method

The Newton-Raphson method is the simplest method available for tracing the equilibrium path up to the first limit point. The tangent stiffness matrix is recalculated after each iteration (Fig. 6.1). The basic equations for the Newton-Raphson method are

$$K^k \cdot \Delta q^k = R^k \tag{6.1}$$

$$q^{k+1} = q^k + \Delta q^k \tag{6.2}$$

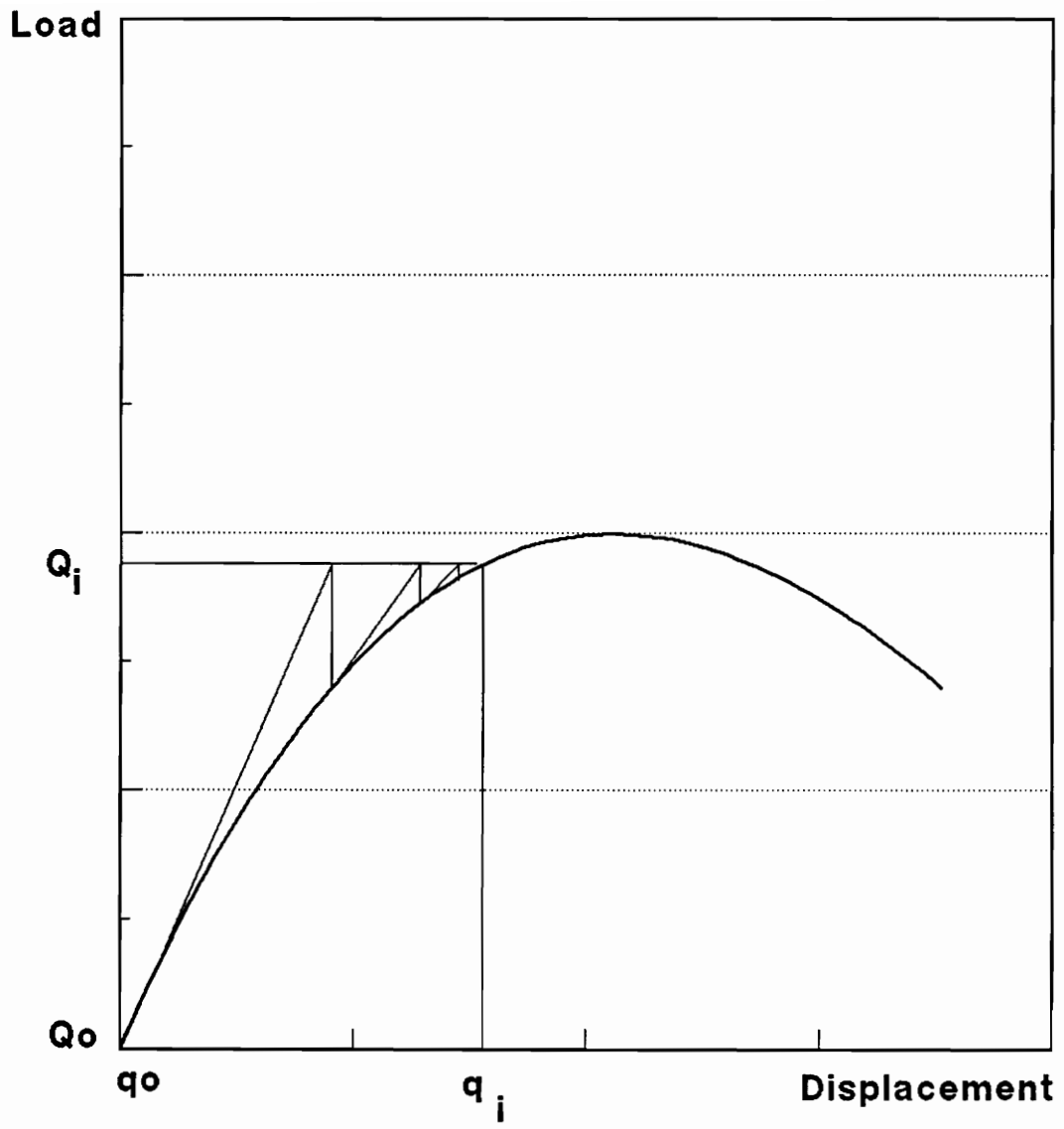


Fig. 6.1 Newton-Raphson Method (Cook, 1989)

where

K^k - tangent stiffness matrix at iteration k

Δq^k - incremental displacement vector at iteration k

R^k - residual force vector at iteration k

q^{k+1} - total displacements at iteration k+1

q^k - total displacement at iteration k

The load is held constant during a given load increment. The method can not handle any load decrement; for this reason, the Newton-Raphson method can not go beyond the first limit point. Furthermore, the method needs more iterations to converge near the limit point.

6.3 Modified Newton-Raphson

The Modified Newton-Raphson is a variant of the standard Newton-Raphson method (Fig. 6.2). The only difference is that the Modified Newton-Raphson will not recalculate the tangent stiffness matrix at every iteration, but only at the beginning of a load increment or wherever the previous tangent stiffness matrix fails to converge (Cook, 1989). The basic equations for the Modified Newton-Raphson method are

$$K^i \cdot \Delta q^k = R^k \tag{6.4}$$

$$q^{k+1} = q^k + \Delta q^k \tag{6.5}$$

$$\lambda^{k+1} = \lambda^k \tag{6.6}$$

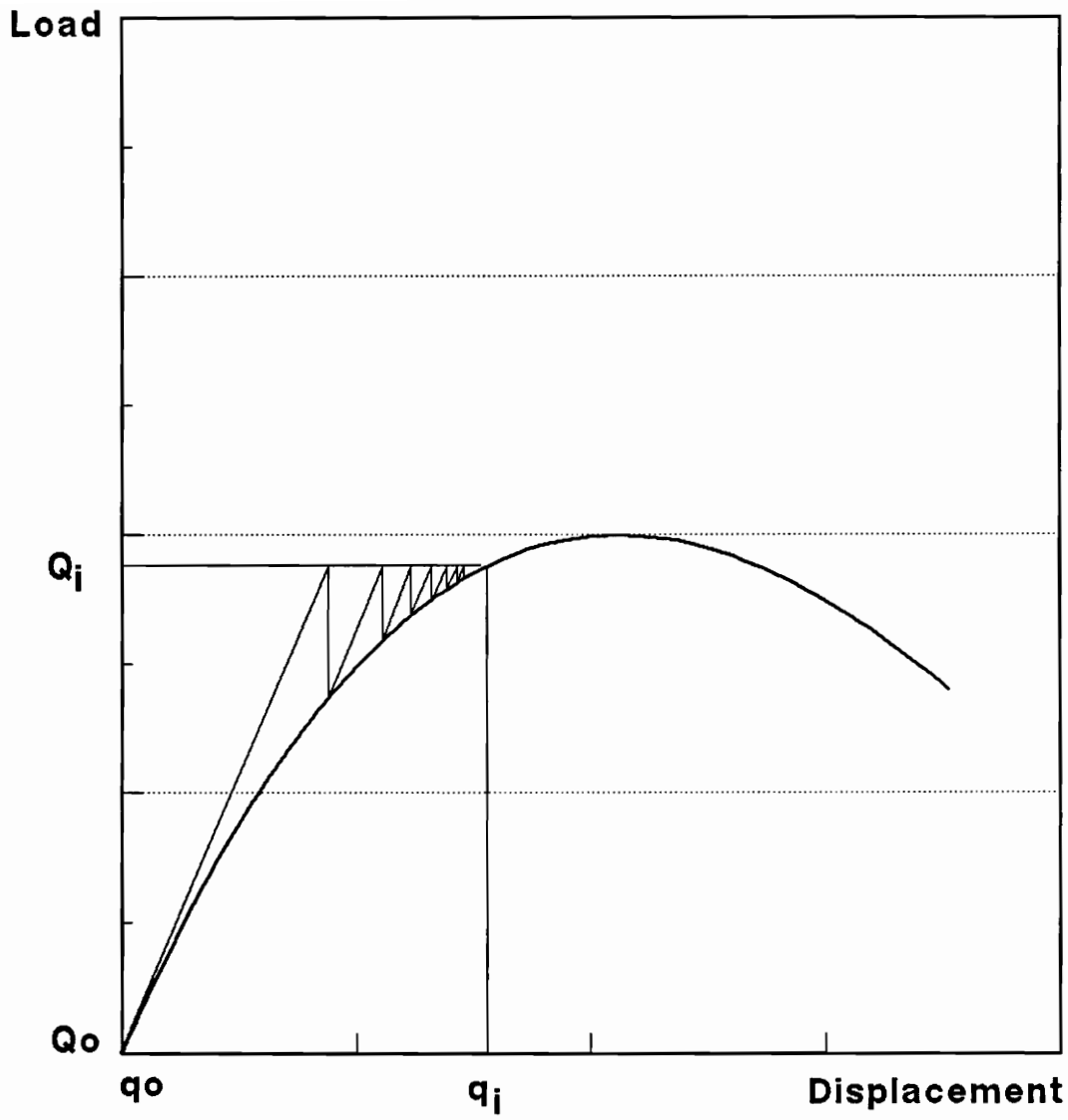


Fig. 6.2 Modified Newton-Raphson Method (Cook, 1989)

where K^i is the tangent stiffness matrix at the beginning of load step i. This scheme will make the method more efficient than the standard method, especially if one need responses from the points significantly lower than the first limit point.

6.4 Modified Riks/Wempner

The modified Riks/Wempner method can be used to trace along the equilibrium path beyond limit points, even for the most complex equilibrium path. The path is incremented for each step and the load vector is changed during a step.

The original Riks-Wempner formulation requires operations that will destroy the bandedness and the symmetric properties of the stiffness matrix. Using the modified Riks-Wempner, one can preserve the size of the K matrix by paying the price of solving two linear equations instead of one (Holzer et al., 1981).

The next equilibrium point is search along a normal plane perpendicular to the searching path/a tangent line from the previous equilibrium point. With the variable path scheme, the length of the searching path varies during the iteration process, so that the number of iterations needed for all load increments are close to the desired number of iterations (Fig. 6.3)

Modified Riks-Wempner using sphere without updating is the most efficient method available to trace an equilibrium path. The basic procedure is same as the iteration using normal plane, but all the iteration points now lie on a sphere with the current equilibrium point O as the center and with the radius equal to ρ (Fig.

6.4).

Because the searching path and the sphere can intersect at two points, to find the intersection one must solve a quadratic equation. This will give another problem, because the discriminant of the quadratic equation can be positive, and give two roots, or negative, and give no real roots at all. Criesfield (1981, 1983) gave some guidances for selecting the right root of the equation.

In this study, the modified Riks-Wempner with normal plane is used. The Riks-Wempner with sphere was also tried, but it is found that it only gives little improvement, and needs more complicated algorithm to handle the root selection and the negative discriminant problems.

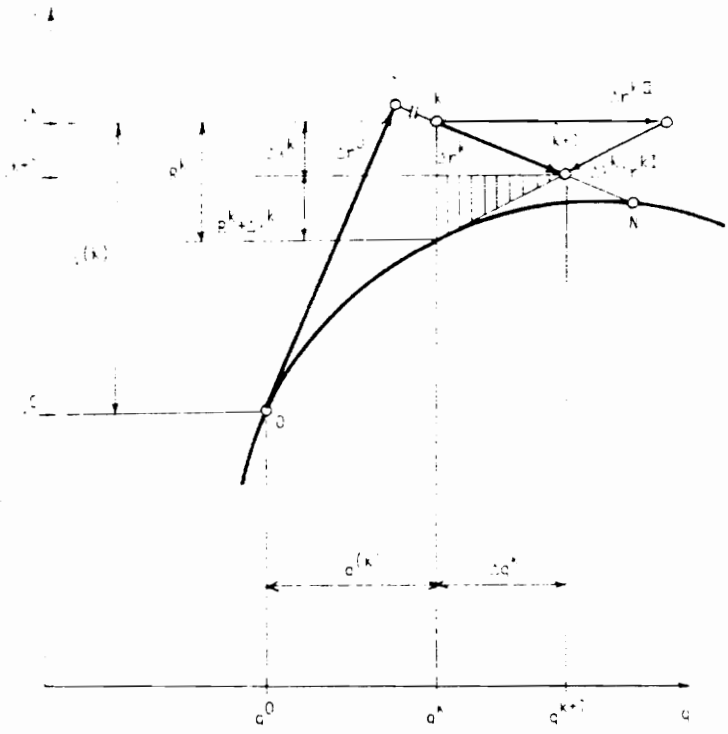


Fig.6.3 Modified Riks/Wempner Method with Normal Plane
(Holzer et al., 1981)

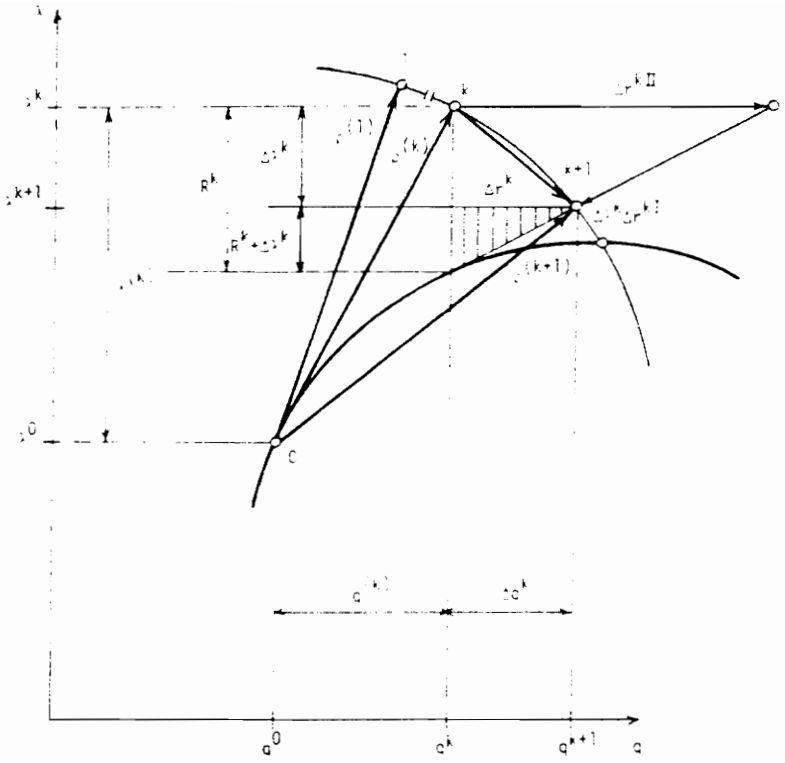


Fig.6.4 Modified Riks/Wempner Method with Sphere
(Holzer et al., 1981)

Chapter 7
Conclusions and Recommendations

7.1 Conclusions

The results of this work are the closed-form expressions for the tangent and secant stiffness matrices of the 3-node Mindlin frame element. From the study of two solution methods and the behavior of the three elements, several conclusions have been made as follows:

1. Because the Mindlin frame element uses quadratic polynoms for the interpolation functions, at least two elements are needed for modelling a member of a plane frame structure.

2. The Mindlin frame element will give a more flexible structure due to the shear deformation effect, but the effect is not always significant.

3. The Mindlin frame element requires about twice degree-of-freedoms than the Bernoulli-Euler frame element requirement.

4. The formulation of tangent stiffness matrix can be simplified, in condition that the internal forces calculations are done using the appropriate nonlinear formulation. The calculation of the tangent stiffness matrix and the internal element forces can be done using different formulations.

7.2 Recommendations

The following recommendations are made to improve and extend this present work:

1. Improve the 3-node Mindlin frame element by reduced integration, discrete Kirchoff constraint, or by using a higher order element, to avoid shear locking effect for slender elements.

2. Use the Bernoulli-Euler element for the first nonlinear analysis of a plane frame problem because it requires less degree-of-freedom than the 3-node Mindlin element while giving a good approximations.

3. Use the modified Riks/Wempner method for nonlinear analysis of a structure with unknown equilibrium path characteristic. If the response is below the limit points, the variable step Newton-Raphson can be used effectively.

4. Add capability to handle material nonlinearity.

References

1. Argyris, J. H. et al., "Finite Element Method - The Natural Approach", Computer Methods in Applied Mechanics and Engineering, Vol. 17/18, pt. 1, 1979, pp. 1-106.

2. Bathe, K.-J., Ramm, E. and Wilson, E. L., "Finite Element Formulation for Large Deformation Dynamic Analysis", International Journal for Numerical Methods in Engineering, Vol. 9, 1975, pp. 353-386.

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

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

5. Bathe, K.-J., "Finite Element Procedures for Solids and Structures - Nonlinear Analysis", Video Course Study Guide, MIT Video Course, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1984.

6. Butler, M. J., " A Comparison of Two Models for Geometrically Nonlinear Finite Element Analysis of Plane Frame", M. S. Thesis, Virginia Polytechnic

Institute and State University, Blacksburg, Virginia, December, 1983.

7. Carey, G. F., " A Unified Approach to Three Finite Element Theories for Geometric Nonlinearity", Computer Methods in Applied Mechanics and Engineering, Vol 4, No. 1, 1974, pp. 69-79.

8. Chajes, A. and Churcill, J. E., "Nonlinear Frame Analysis by Finite Element Methods", Journal of Structural Engineering, Vol. 113, No. 6, American Society of Civil Engineers, June, 1987, pp. 1221-1235.

9. Cook, R. D. and Plesha, M., Concepts and Applications of Finite Element Analysis, Third edition, John Wiley and Sons, 1989.

10. Criesfield, M. A., "A Fast Incremental/Iterative Solution Procedure that Handles 'Snap-Through' ", Computers and Structures, Vol. 13, Pergamon Press, Ltd., 1981, pp. 55-62.

11. Criesfield, M. A., "An Arc-Length Method Including Line Searches and Accelerations", International Journal for Numerical Methods in Engineering, Vol. 19, 1983, pp. 1269-1289.

12. Gadala, M. S., Dokainish, M. A. and Oravas, G. AE., "Formulation Methods of Geometric and Material Nonlinearity Problems", International Journal for Numerical Methods in Engineering, Vol. 20, John Wiley and Sons, Ltd., 1981, pp. 887-914.

13. Hinton, E. and Owen, D. R. J., Finite Element Programming, Academic Press, Inc., London, 1977.

14. Hinton, E. and Owen, D. R. J., An Introduction to Finite Element Computations, Pineridge Press Limited, Swansea, UK, 1979.

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

16. Holzer, S. M., Matrix Structural Analysis - Structured Programming, Elsevier Science Publisher, 1986.

17. Holzer, S. M., "Lecture Notes on Dynamics Analysis of Structures", Virginia Polytechnic Institute and State University, Blacksburg, Virginia, Fall, 1990.

18. Holzer, S. M., "Lecture Notes on Finite Element Analysis of Structures", Virginia Polytechnic Institute and State University, Blacksburg, Virginia, Spring, 1991.

19. Holzer, S. M., "Lecture Notes on Computer Analysis of Structures II", Virginia Polytechnic Institute and State University, Blacksburg, Virginia, Spring, 1991.

20. Katzenberger, G. S., "Geometrically Nonlinear Analysis of Pressure-Loaded Arches, Rings, and Frames Using a Follower Force Algorithm", M. S. Thesis,

Virginia Polytechnic Institute and State University, Blacksburg, Virginia, August, 1983.

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

22. Meek, J. L. and Tan, H. S., "Geometrically Nonlinear Analysis of Space Frame by an Incremental Iterative Technique", Computer Methods in Applied Mechanics and Engineering, Vol. 47, Elsevier Science Publisher, 1984, pp. 261-282.

23. Suhirtharatnasingam, N., "Isobeam Finite Element Analysis of Curved and Tapered Beams", M. S. Thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, February, 1989.

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

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

25. Wood, R. D. and Screfler, B., "Geomterically Nonlinear Analysis - A Correlation of Finite Element Notations", International Journal for

Numerical Methods in Engineering, Vol. 12, 1978, pp. 635-642.

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

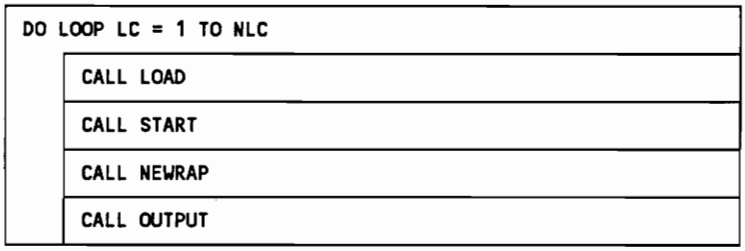
SUBROUTINE : MAIN PROGRAM
PURPOSE : OPEN FILE,ALLOCATE MEMORY

READ FILE NAME AND OPEN FILE READ CONTROL VARIABLES ALLOCATE MEMORY SPACE			
IF ENOUGH MEMORY THEN			ELSE
CALL STRUCT			
ALLOCATE MEMORY SPACE			
IF ENOUGH MEMORY THEN		ELSE	
IF LINEAR PROBLEM THEN	ELSE		
CALL LINEAR	CALL NONLINEAR	PRINT NOT ENOUGH MEMORY	

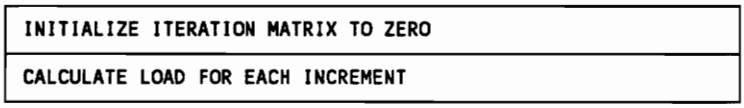
SUBROUTINE : LINEAR
PURPOSE : SOLVE LINEAR PROBLEM

CALL STIFF
CALL FACTOR
DO LOOP LC = 1 TO NLC
CALL LOAD
CALL SOLVE
CALL RESULT

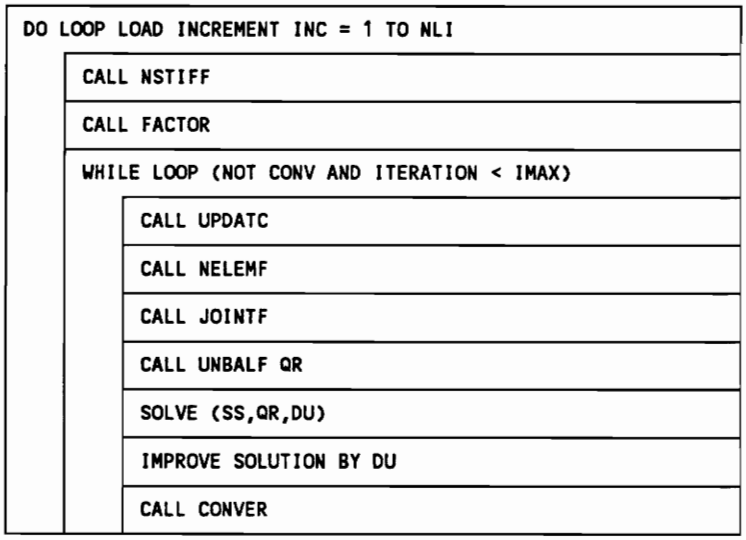
SUBROUTINE : NONLIN
PURPOSE : SOLVE NONLINEAR PROBLEM



SUBROUTINE : START
PURPOSE : INITIALIZE MATRIX FOR NONLINEAR ANALYSIS



SUBROUTINE : NEWRAP
PURPOSE : NONLINEAR ANALYSIS BY MODIFIED NEWTON-RAPHSON



```
SUBROUTINE : RIKWEM
PURPOSE    : NONLINEAR ANALYSIS BY MODIFIED RIKS-WEMPNER METHOD

INITIALIZE ITERATION:
IRCM = 5
CALL STIFF
CALL FACTOR
DLO = 1/NLI
CALL SOLVE(SS,QO,UO)
DSO = DLO * DSQRT(DOTPRD(UO,UO) + 1)
DSMIN = DSO/4
DSMAX = DSO*4

DO LOOP UNTIL LD > LDMAX
    CALL UPDATC
    CALL UNBALF
    CALL NSTIFF
    CALL FACTOR
    WHILE LOOP (NOT CONV AND ITER < IMAX)
        SOLVE FOR DUK1,DUK2:
        CALL SOLVE(SS,QO,DUK1)
        CALL SOLVE(SS,QR,DUK2)
        DUK = DLK * DUK1 + DUK2  (VECTOR OPERATION)
        IMPROVE SOLUTION BY DUK,DLK
        CALL UPDATC
        CALL UNBALF
        CALL NSTIFF
        CALL FACTOR
        CALL STURM
        CALL CONVER
    PREPARE NEXT STEP, CALCULATE DS, DLIO, DUIO:
    RATIO = DSQRT(IRCM/ITER)
    DS = DS * RATIO
    CALL SOLVE(SS,QO,UO)
    DELTA = DS/DSQRT(DOTPRD(UO,UO)+1)
    TEMP  = DOTPRD(DUII,UO)+DLII
    DLIO  = SIGN(TEMP)*DELTA
    DUIO  = DLIO*UO
```

SUBROUTINE : UPDATC
PURPOSE : UPDATE JOINT COORDINATES, LENGTH, COSINES

```
DO LOOP J = 1,NJ
DO LOOP L = 1,2
    K = JCODE(L,J)
    IF K > 0 THEN
            THEN                    |   ELSE
    X(L,J) = X(L,J) + Q(K)          |
DO LOOP I = 1,NE
    J = MINC(1,I)
    K = MINC(2,I)
    DX = X(1,K) - X(1,J)
    DY = X(2,K) - X(2,J)
    EL = DSQRT(DX*DX+DY*DY)
    ELENG(I) = EL
    C1(I) = DX/EL
    C2(I) = DY/EL
```

SUBROUTINE : CONVER
PURPOSE : TEST FOR CONVERGENCE

```
IFLAG = 1, I = 1
WHILE LOOP (I < NEQ) AND (IFLAG = 1)
    TEMP = QR(I)/QI(I)
    IF (ABS(TEMP) > RTOL)
            THEN                    |   ELSE
    IFLAG = 0                       |
    I = I + 1
```

SUBROUTINE : STRUCT
PURPOSE : READ STRUCTURAL DATA, COMPUTE STRUCTURAL PARAMETERS

```
DO LOOP I = 1,NE
    READ MINC(1,I), MINC(2,I)
DO LOOP J = 1,NJ
DO LOOP L = 1,2
    JCODE(L,J) = 1
READ JNUM,JDIR
WHILE LOOP (JNUM > 0)
    IF (JNUM > 0)
            THEN                    |   ELSE
    JCODE(JDIR,JNUM) = 0            |
    READ  (JNUM,JDIR)
CALL CODES, SKYLIN, PROP
```

SUBROUTINE : CODES
PURPOSE : GENERATE JCODE MATRIX

```
NEQ = 0
DO LOOP J = 1,NJ
DO LOOP L = 1,2
    IF (JCODE(L,J) > 0)
            THEN                    |   ELSE
    NEQ = NEQ + 1                   |
    JCODE(L,J) = NEQ                |
DO LOOP I = 1,NE
    J = MINC(1,I)
    K = MINC(2,I)
    DO LOOP L = 1,2
        MCODE(L,I)   = JCODE(L,J)
        MCODE(L+2,I) = JCODE(L,K)
```

SUBROUTINE : SKYLIN
PURPOSE : GENERATE KHT,MAXA

```
DO LOOP I = 1,NEQ
    KHT(I) = 0
DO LOOP I = 1,NE
    J = 1
    WHILE (MCODE(J,I) = 0) AND (J < 4)
        J = J + 1
    MIN = MCODE(J,I)
    J = J + 1
    DO LOOP L = J,4
        K = MCODE(L,I)
        IF (K > 0)
                THEN                        |   ELSE
        KHT(K) = MAXO(KHT(K),K-MIN))        |
MAXA(1) = 1
DO LOOP I = 1,NEQ
    MAXA(I+1) = MAXA(I) + KHT(I) + 1
LSS = MAXA(NEQ+1)-1
```

```
SUBROUTINE : PROP
PURPOSE    : READ JOINT COORDINATES, ELEMENT PROPERTIES
```

```
DO LOOP J = 1,NJ
    READ X(1,J), X(2,J)
DO LOOP I = 1,NE
    J = MINC(1,I)
    K = MINC(2,I)
    DX = X(1,K) - X(1,J)
    DY = X(2,K) - X(2,J)
    EL = DSQRT(DX*DX+DY*DY)
    ELENG(I) = EL
    C1(I) = DX/EL
    C2(I) = DY/EL
DO I = 1,NE
    READ EMOD(I),AREA(I)
```

```
SUBROUTINE : LOAD
PURPOSE    : INITIALIZE LOAD VECTOR, ELEMENT FORCES
```

```
DO LOOP K = 1,NEQ
    Q(K) = 0
DO LOOP I = 1,NE
DO LOOP L = 1,4
    F(L,I) = 0
CALL JLOAD
```

```
SUBROUTINE : JLOAD
PURPOSE    : READ JOINT LOAD
```

```
READ  JNUM,JDIR,FORCE
WHILE LOOP (JNUM > 0)
    K = JCODE(JDIR,JNUM)
    Q(K) = FORCE
    READ JNUM,JDIR,FORCE
```

SUBROUTINE : STIFF
PURPOSE : INIT, BUILD, ASSEMBLY LINEAR STIFFNESS MATRIX

```
DO LOOP I = 1,LSS
    SS(I) = 0
DO LOOP I = 1,NE
    CALL ELEMS
    CALL ASSEMS
```

SUBROUTINE : ELEMS
PURPOSE : ELEMENT GLOBAL LINEAR STIFFNESS MATRIX

```
EAL = EMOD(I) * AREA(I) /  ELENG(I)
G(1) = EAL*C1(I)**2
G(2) = EAL*C1(I)*C2(I)
G(3) = EAL*C2(I)**2
```

SUBROUTINE : NSTIFF
PURPOSE : INIT, BUILD, ASSEMBLY NONLINEAR STIFFNESS MATRIX

```
DO LOOP I = 1,LSS
    SS(I) = 0
DO LOOP I = 1,NE
    CALL NELEMS
    CALL ASSEMS
```

SUBROUTINE : ELEMS
PURPOSE : GLOBAL NONLINEAR STIFFNESS MATRIX OF TRUSS ELEMENT

```
CALL LDISP
EAL = EMOD(I) * AREA(I) / ELENG(I)
EL  = ELENG(I)
FO  = F(3,I)
A = (DL(3)-DL(1))/EL
B = (DL(4)-DL(2))/EL
g1 = EAL*(1+2*A+A**2) + FO/EL
g2 = EAL*(A*B+B)
g3 = EAL*(B**2)+FO/EL
CA = C1(I)*C1(I)
CB = C2(I)*C2(I)
CC = C1(I)*C2(I)
G(1) = g1*CA + g3*CB - 2*g2*CC
G(2) = g2*CA - g2*CB + g1*CC - g3*CC
G(3) = g3*CA + g1*CB + 2*g2*CC
```

SUBROUTINE : ASSEMS
PURPOSE : ASSEMBLY ELEMENT GLOBAL STIFFNESS MATRIX

```
DO LOOP JE = 1,4
    J = MCODE(JE,N)
    IF (J > 0)
                    THEN                          | ELSE
    DO IE = 1,JE
        I = MCODE(IE,N)
        IF (I > 0)
                THEN                              | ELSE
        K = MAXA(J) + J - I
        L = INDEX(IE,JE)
        IF (L > 0)
            THEN                  | ELSE
        SS(K) = SS(K) +           | SS(K) = SS(K) -
                G(L)              |         G(-L)
```

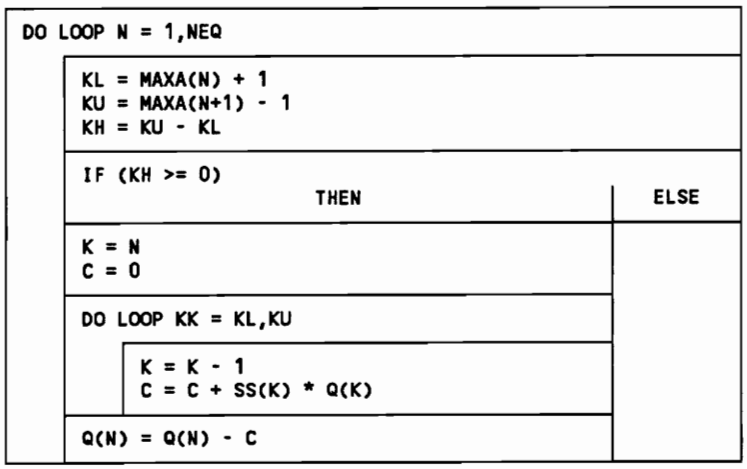
```
SUBROUTINE : SOLVE
PURPOSE    : SOLVE K x q = Q, FOR A LOAD VECTOR
```

```
CALL FORSUB
CALL BACSUB
```

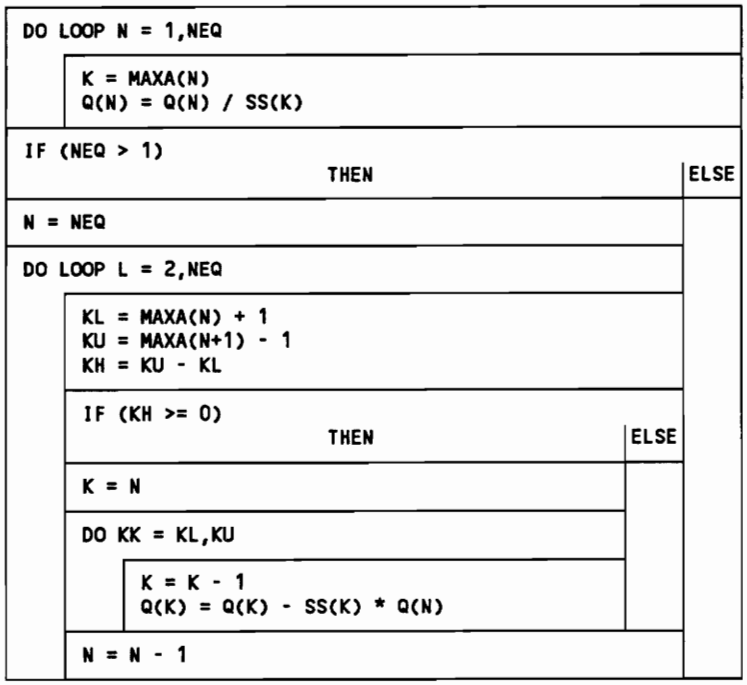
```
SUBROUTINE : FACTOR
PURPOSE    : FACTORIZE K TO LDU FORM
```

```
DO LOOP N = 1,NEQ
    KN = MAXA(N)
    KL = MAXA(N+1)-1
    KU = MAXA(N+1)-1
    KH = KU-KL
    IF (KH >= 0)
                    THEN                                    | ELSE
        IF (KH > 0)
                    THEN                                    | ELSE
            K   = N - KH
            IC  = 0
            KLT = KU
            DO J = 1,KH
                IC  = IC + 1
                KLT = KLT - 1
                KI  = MAXA(K)
                ND  = MAXA(K+1) - KI - 1
                IF (ND > 0)
                            THEN                            | ELSE
                    KK = MINO(IC,ND)
                    C  = 0.0
                    DO LOOP L = 1,KK
                        C = C + SS(KI+L) * SS(KLT+L)
                        SS(KLT) = SS(KLT) - C
                    K = K + 1
        K = N
        B = 0.0
        DO KK = KL,KU
            K  = K - 1
            KI = MAXA(K)
            C  = SS(K)/SS(KI)
            B  = B + C*SS(KK)KI - 1
            SS(K) = C
    IF (SS(KN) <= 0)
                    THEN                                    | ELSE
        PRINT NONPOSITIVE DEFINITE STIFFNESS MATRIX
        STOP
```

SUBROUTINE : FORSUB
PURPOSE : FORWARD SUBSTITUTION



SUBROUTINE : BACSUB
PURPOSE : BACK SUBSTITUTION



SUBROUTINE : RESULT
PURPOSE : CALCULATE JOINT FORCES, ELEMENT FORCES, OUTPUT

```
INITIALIZE JOINT FORCES MATRIX TO ZERO
CALL FORCES
CALL OUTPUT
```

SUBROUTINE : FORCES
PURPOSE : CALCULATE JOINT FORCES, ELEMENT FORCES

```
INITIALIZE JOINT FORCES MATRIX TO ZERO
CALL ELEMF
CALL JOINTF
```

SUBROUTINE : ELEMF
PURPOSE : CALCULATE TRUSS ELEMENT FORCES (LINEAR)

```
CALL LDISP
EAL = EMOD(I) * AREA(I) / ELENG(I)
F1  = EAL * (DL(1)-DL(3))
F3  = -F1
F(1,I) = F(1,I) + F1
F(3,I) = F(3,I) + F3
```

SUBROUTINE : NELEMF
PURPOSE : CALCULATE TRUSS ELEMENT FORCES (NONLINEAR)

```
DO LOOP I = 1 TO NE
    CALL LDISP
    EL  = ELENG(I)
    EAL = EMOD(I) * AREA(I) / ELENG(I)
    FO  = EAL * (DLI(3) - DLI(1))
    F(1,I) = -FO
    F(3,I) =  FO
    FF(1,I) = FF(1,I) - FO
    FF(3,I) = FF(3,I) + FO
```

SUBROUTINE : LDISP
PURPOSE : GET LOCAL DISPLACEMENT

```
DO LOOP L = 1,4
    K = MCODE(L,I)
    IF (K > 0)
        D(L) = Q(K)
    ELSE
        D(L) = 0.0
DL(1) =  D(1)*C1(I)+D(2)*C2(I)
DL(2) = -D(1)*C2(I)+D(2)*C1(I)
DL(3) =  D(3)*C1(I)+D(4)*C2(I)
DL(4) = -D(3)*C2(I)+D(4)*C1(I)
```

SUBROUTINE : JOINTF
PURPOSE : CALCULATE JOINT FORCES

```
DO LOOP I = 1 TO NE
    J = MINC(1,I)
    K = MINC(2,I)
    P(1,J) = P(1,J) + ( F(1,I)*C1(I) - F(2,I)*C2(I) )
    P(2,J) = P(2,J) + ( F(1,I)*C2(I) + F(2,I)*C1(I) )
    P(1,K) = P(1,K) + ( F(3,I)*C1(I) - F(4,I)*C2(I) )
    P(2,K) = P(2,K) + ( F(3,I)*C2(I) + F(4,I)*C1(I) )
```

SUBROUTINE : OUTPUT
PURPOSE : PRINT JOINT DISPLACEMENT AND ELEMENT FORCES

```
DO LOOP J= 1,NJ
    PRINT JOINT DISPLACEMENTS
DO LOOP I = 1,NE
    PRINT ELEMENT FORCES
```

```
================================================================
| FORTRAN-77 listing for Newton-Raphson and Riks/Wempner methods |
================================================================


C*********************************************************************
C*                          NONLINEAR ANALYSIS                       *
C*********************************************************************
      SUBROUTINE NONLIN(A,IA,NXO,NX,NAREA,NEMOD,NZI,NCTE,NMINC,NMCODE,
     $           NJCODE,NQ,NNA,NSS,NMAXA,NF,NP,NQO,NDQ,NQI,NQR,NQP,
     $           NQQ,NQT,NPP,NFF,NE,NJ,NDUIO,NDUII,NUO,NDUK,NQK,
     $           NDUK1,NDUK2,NLC,NEQ,LSS,METHOD,METK,METF,
     $           NLI,NIMAX,QMAX,TOLF,TOLM,TOLD,TOLR,JOINT,JDIR)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION A(*),IA(*)
C
      DO 10 LC = 1, NLC
         CALL LOAD(A(NF),A(NQ),A(NXO),A(NX),A(NAREA),A(NEMOD),A(NCTE),
     $             IA(NMINC),IA(NMCODE),IA(NJCODE),IA(NNA),NE,NEQ,LC)
C
         CALL START(A(NXO),A(NX),A(NQ),A(NP),A(NF),A(NQO),A(NDQ),
     $              A(NQI),A(NQP),A(NQQ),A(NQT),A(NPP),A(NFF),
     $              IA(NJCODE),NE,NJ,NEQ,NLI,JOINT,JDIR,IDXDOF)
C
         IF (METHOD .EQ. 1) THEN
            CALL NEWRAP(A(NSS),A(NQ),A(NXO),A(NX),A(NAREA),A(NEMOD),
     $        A(NZI),IA(NMINC),IA(NMAXA),IA(NMCODE),IA(NJCODE),
     $        A(NP),A(NF),A(NQO),A(NDQ),A(NQI),
     $        A(NQR),A(NQP),A(NQQ),A(NQT),A(NQK),A(NPP),A(NFF),
     $        NE,NJ,NEQ,LSS,LC,METK,METF,NLI,NIMAX,QMAX,
     $        TOLF,TOLD,IDXDOF)
         ELSEIF (METHOD .EQ. 2) THEN
            CALL RIKWEM(A(NSS),A(NQ),A(NXO),A(NX),A(NAREA),A(NEMOD),
     $        A(NZI),IA(NMINC),IA(NMAXA),IA(NMCODE),IA(NJCODE),
     $        A(NP),A(NF),A(NQO),A(NQR),A(NQP),A(NQQ),
     $        A(NPP),A(NFF),NE,NJ,NEQ,LSS,LC,METK,METF,
     $        NLI,NIMAX,QMAX,TOLF,TOLD,IDXDOF,A(NDUIO),A(NDUII),
     $        A(NUO),A(NDUK),A(NQK),A(NDUK1),A(NDUK2))
         ENDIF
C
C------- SAVE FINAL OUTPUT TO DISK, ACCUMULATED RESULTS : PP,FF,QQ
C
         CALL OUTPUT(A(NFF),A(NPP),A(NQT),IA(NJCODE),NE,NJ)
C
   10 CONTINUE
C
      RETURN
      END
```

```
C*************************************************************************
C*                        NEWTON-RAPHSON METHOD                          *
C*************************************************************************
      SUBROUTINE NEWRAP(SS,Q,XO,X,AREA,EMOD,ZI,MINC,MAXA,MCODE,JCODE,
     $                  P,F,QO,DQ,QI,QR,QP,QQ,QT,QK,PP,FF,
     $                  NE,NJ,NEQ,LSS,LC,METK,METF,NLI,NIMAX,
     $                  QMAX,TOLF,TOLD,IDXDOF)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION SS(*),Q(*),XO(2,*),X(2,*),AREA(*),EMOD(3,*),ZI(*),
     $          MINC(3,*),MAXA(*),MCODE(9,*),JCODE(3,*),P(3,*),F(9,*),
     $          QO(*),DQ(*),QI(*),QR(*),QP(*),QQ(*),QT(*),
     $          QK(*),PP(3,*),FF(9,*)
      DOUBLE PRECISION LD
C
      PRINT *,' NEWTON-RAPHSON ITERATION'
      PRINT *,' '
C
C-----CALCULATE KO
C
      CALL STIFF(SS,XO,AREA,ZI,EMOD,MINC,MAXA,MCODE,NE,LSS)
      CALL FACTOR(SS,MAXA,NEQ)
      WRITE(9,'(E16.8,4X,E16.8)') 0.0, 0.0
      DLIO = 1.0D0 / NLI
      DLMAX = DLIO*4
      DLMIN = DLIO/4
      ITOT = 0
      INCR = 0
      IRCM = 10
      DLI  = DLIO
      LD   = 0.0D0
C
C-----LOAD INCREMENT LOOP
   10 IF (LD .LT. QMAX) THEN
         INCR = INCR + 1
         LD   = LD   + DLI
         CALL VCOPY(QO,DQ,NEQ,DLI)
         CALL VADD(QI,DQ,QI,NEQ)
         CALL VSUB(QI,QP,QR,NEQ)
C
C--------WHILE LOOP FOR CONVERGENCE OR ITER = NIMAX
         IFLAG = 0
         ITER  = 0
   20    IF (IFLAG .NE. 1 .AND. ITER .LT. NIMAX) THEN
            ITER = ITER + 1
            ITOT = ITOT + 1
            WRITE(*,'(A,2X,3(I3,2X),2X,E16.8,2X,E16.8)')
     $      'I,J,T,LD,Q : ',INCR,ITER,ITOT,LD,QT(IDXDOF)
C
C-----------SOLVE FOR DISPLACEMENTS INCREMENT Q AND ACCUMULATE
            CALL SOLVE(SS,QR,Q,MAXA,NEQ)
            CALL VADD(QT,Q,QT,NEQ)
C
C-----------UPDATE COORD, CALCULATE UNBALANCED FORCES, UPDATE KT
            CALL UPDATC(X,Q,JCODE,NJ)
            CALL UNBAL(XO,X,AREA,EMOD,ZI,MINC,MCODE,JCODE,NE,NJ,NEQ,
     $                 METF,METK,QT,F,FF,P,PP,QP,QI,1.0D0,QK,QR)
            CALL NSTIFF(SS,XO,X,AREA,EMOD,ZI,MINC,MAXA,MCODE,
     $                  QT,FF,NE,LSS,METK)
            CALL FACTOR(SS,MAXA,NEQ)
C
C-----------CHECK CONVERGENCE
            CALL CONVER(QI,QR,Q,QT,NEQ,TOLF,TOLD,IFLAG)
            GO TO 20
         ENDIF
         PRINT *, ' '
```

```
C--------UPDATE DLI FOR NEXT ITERATION
         RATIO = DSQRT(1.0D0 * IRCM/ITER)
         DLI   = DLI * RATIO
         IF (DLI .GT. DLMAX) DLI = DLMAX
         IF (DLI .LT. DLMIN) DLI = DLMIN
C
C--------SAVE ITERATION OUTPUT AT IDXDOF: DISPL,JFORCE
         WRITE(9,'(E16.8,4X,E16.8)') ABS(QT(IDXDOF)),LD
         GOTO 10
      ENDIF
      PRINT *,'  '
      PRINT *,'TOTAL ITERATION = ',ITOT
C
      RETURN
      END
```

```
C*********************************************************************
C*                      RIKS-WEMPNER (NON-LINEAR)                    *
C*********************************************************************
      SUBROUTINE RIKWEM(SS,Q,XO,X,AREA,EMOD,ZI,MINC,MAXA,MCODE,JCODE,
     $                  P,F,QO,QR,QP,U,PP,FF,NE,NJ,NEQ,
     $                  LSS,LC,METK,METF,NLI,NIMAX,QMAX,TOLF,
     $                  TOLD,IDXDOF,DUIO,DUII,UO,DUK,QK,DUK1,DUK2)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION SS(*),Q(*),XO(2,*),X(2,*),AREA(*),EMOD(3,*),ZI(*),
     $          MINC(3,*),MAXA(*),MCODE(9,*),JCODE(3,*),P(3,*),
     $          F(9,*),QO(*),QR(*),QP(*),U(*),PP(3,*),FF(9,*),
     $          DUIO(*),DUII(*),UO(*),DUK(*),QK(*),DUK1(*),DUK2(*)
      DOUBLE PRECISION LD,LDMAX
C
      IRCM  = 5
      LDMAX = QMAX
      PRINT *,' RIKS-WEMPNER ITERATION'
      PRINT *,'   IRCM  = ',IRCM
      PRINT *,'   LDMAX = ',LDMAX
      PRINT *,' '
      WRITE(9,'(E16.8,4X,E16.8)') 0.0, 0.0
C
C-----PREPARATION FOR ITERATION, CALCULATE INITIAL DS
C
      CALL STIFF(SS,XO,AREA,ZI,EMOD,MINC,MAXA,MCODE,NE,LSS)
      CALL FACTOR(SS,MAXA,NEQ)
      DLO = 1.0D0 / NLI
      CALL SOLVE(SS,QO,UO,MAXA,NEQ)
      DSO = DLO * DSQRT(DOTPRD(UO,UO,NEQ) + 1.0D0)
      DSMIN = DSO/4
      DSMAX = DSO*4
      DS  = DSO
C
      NGV  = 0
      LD   = 0.0D0
      DLIO = DLO
      CALL VZERO(U,NEQ)
      CALL VCOPY(UO,DUIO,NEQ,DLIO)
C
      ITOT = 0
C
C-----REPEAT STEP UNTIL LD > LDMAX
C
   10   DLII = DLIO
        LD   = LD + DLIO
        CALL VADD(U,DUIO,U,NEQ)
        CALL VCOPY(DUIO,DUII,NEQ,1.0D0)
        CALL UPDATC(X,DUIO,JCODE,NJ)
        CALL UNBAL(XO,X,AREA,EMOD,ZI,MINC,MCODE,JCODE,NE,NJ,NEQ,
     $             METF,METK,U,F,FF,P,PP,QP,QO,LD,QK,QR)
        CALL NSTIFF(SS,XO,X,AREA,EMOD,ZI,MINC,MAXA,MCODE,
     $              U,FF,NE,LSS,METK)
        CALL FACTOR(SS,MAXA,NEQ)
C
        ITER = 0
C
C-------REPEAT ITERATION : K = 1..IMAX
C
   20     ITER = ITER + 1
          ITOT = ITOT + 1
C
C---------SOLVE FOR DUK1, DUK2
          CALL SOLVE(SS,QO,DUK1,MAXA,NEQ)
          CALL SOLVE(SS,QR,DUK2,MAXA,NEQ)
          T1 = -1.0D0 * DOTPRD(DUIO,DUK2,NEQ)
          T2 = (DOTPRD(DUIO,DUK1,NEQ) + DLIO)
          DLK = T1 / T2
```

```
            DO 60 I = 1,NEQ
               DUK(I) = DLK * DUK1(I) + DUK2(I)
   60       CONTINUE
C
C---------IMPROVE SOLUTION
            LD   = LD   + DLK
            DLII = DLII + DLK
            CALL VADD(U,DUK,U,NEQ)
            CALL VADD(DUII,DUK,DUII,NEQ)
C
C---------UPDATE COORDINATE AND KT FOR NEXT ITERATION
            CALL UPDATC(X,DUK,JCODE,NJ)
            CALL UNBAL(XO,X,AREA,EMOD,ZI,MINC,MCODE,JCODE,NE,NJ,NEQ,
     $              METF,METK,U,F,FF,P,PP,QP,QO,LD,QK,QR)
            CALL NSTIFF(SS,XO,X,AREA,EMOD,ZI,MINC,MAXA,
     $                 MCODE,U,FF,NE,LSS,METK)
            CALL FACTOR(SS,MAXA,NEQ)
            CALL STURM(SS,MAXA,NEQ,NGV)
C
C---------CHECK CONVERGENCE
C*********CALL CONVER(QK,QR,DUK,U,NEQ,TOLF,TOLD,IFLAG)
            CALL ABQCNV(QR,JCODE,NJ,TOLF,TOLD,IFLAG)
C
            IF (IFLAG .NE. 1 .AND. ITER .LT. NIMAX) THEN
                GOTO 20
            ENDIF
C
C-------END REPEAT ITERATION LOOP
C
          WRITE(9,'(E16.8,4X,E16.8)') ABS(U(IDXDOF)),LD
C-------PREPARE NEXT STEP, CALCULATE DS,DLIO,DUIO
          RATIO = DSQRT(1.0D0 * IRCM/ITER)
          DS = DS * RATIO
          IF (DS .GT. DSMAX) DS = DSMAX
          IF (DS .LT. DSMIN) DS = DSMIN
C
          CALL SOLVE(SS,QO,UO,MAXA,NEQ)
          DELTA = DS / DSQRT(DOTPRD(UO,UO,NEQ)+1.0D0)
          TEMP  = (DOTPRD(DUII,UO,NEQ)+DLII)
          IF (TEMP .GT. 0.0D0) THEN
             DLIO = DELTA
          ELSE
             DLIO = -DELTA
          ENDIF
          CALL VCOPY(UO,DUIO,NEQ,DLIO)
          PRINT *,'DLIO = ',DLIO
C
          PRINT *,' ------------------------------------------------'
          PRINT *, 'ITOT,LD,U: ',ITOT,'  ',LD,'  ',U(IDXDOF)
C
          IF (LD .LE. LDMAX) THEN
             GOTO 10
          ENDIF
C
C-----END REPEAT STEP LOOP
        PRINT *,'  '
        PRINT *,'TOTAL ITERATION = ',ITOT
C
        RETURN
        END
```

```
C*************************************************************************
C*                      UNBALANCE FORCE COMPUTATION                      *
C*************************************************************************
      SUBROUTINE UNBAL(XO,X,AREA,EMOD,ZI,MINC,MCODE,JCODE,NE,
     $                 NJ,NEQ,METF,METK,U,DF,FF,DP,PP,QP,QO,LD,QK,QR)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DOUBLE PRECISION LD
      DIMENSION XO(2,*),X(2,*),AREA(*),EMOD(3,*),ZI(*),MINC(3,*),
     $          MCODE(9,*),JCODE(3,*),U(*),DF(9,*),FF(9,*),DP(3,*),
     $          PP(3,*),QP(*),QO(*),QK(*),QR(*)
C
      DO 10 J = 1,NJ
      DO 10 L = 1,3
         PP(L,J) = 0.0D0
   10 CONTINUE
C
C-----CALCULATE TOTAL ELEMENT FORCES (DF,FF)
      CALL NELEMF(U,FF,XO,X,AREA,EMOD,ZI,MINC,MCODE,NE,METF,METK)
C
C-----CALCULATE TOTAL JOINT FORCES, USING CURRENT STATE LAMBDA
      CALL JOINTF(FF,PP,X,MINC,NE)
C
C-----ACCUMULATE JOINT FORCES AND STORE AT DOF DIRECTION
      DO 20 J = 1,NJ
      DO 20 L = 1,3
         K = JCODE(L,J)
         IF (K .GT. 0) QP(K) = PP(L,J)
   20 CONTINUE
C
C-----CALCULATE UNBALANCE FORCES
      DO 30 I = 1,NEQ
         QK(I) = LD * QO(I)
         QR(I) = LD * QO(I) - QP(I)
   30 CONTINUE
C
      RETURN
      END

C*************************************************************************
C*                           UPDATE COORDINATES                          *
C*************************************************************************
      SUBROUTINE UPDATC(X,Q,JCODE,NJ)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION X(2,*),Q(*),JCODE(3,*)
C
C-----UPDATE COORDINATES, ONLY DISPLACEMENTS DX,DY
      DO 10 J = 1,NJ
      DO 10 L = 1,2
         K = JCODE(L,J)
         IF (K .GT. 0) THEN
             X(L,J) = X(L,J) + Q(K)
         ENDIF
   10 CONTINUE
      RETURN
      END
```

```
C*************************************************************************
C*                          STURM CHECKING OF D                          *
C*         CHECK IF THE DETERMINANT OF K HAS CHANGED ITS SIGN            *
C*************************************************************************
      SUBROUTINE STURM(SS,MAXA,NEQ,NEGVAL)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION SS(*),MAXA(*)
C
      NGV = NEGVAL
      NEGVAL = 0
      DO 10 I = 1,NEQ
         JJ = MAXA(I)
         IF (SS(JJ) .LT. 0.0D0) THEN
            NEGVAL = NEGVAL + 1
         ENDIF
   10 CONTINUE
      IF (NGV .NE. NEGVAL) PRINT *,'NGV = ',NEGVAL
      RETURN
      END
C
C*************************************************************************
C*                          CONVERGENCE TESTS                            *
C*************************************************************************
      SUBROUTINE CONVER(QI,QR,Q,QT,NEQ,TOLF,TOLD,IFLAG)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION QI(*),QR(*),Q(*),QT(*)
C
      IFLAG = 1
C-----TEST FOR FORCE TOLERANCE
      T1 = DOTPRD(QR,QR,NEQ)
      T2 = DOTPRD(QI,QI,NEQ)
      IF (T2 .NE. 0.0) THEN
         TEMP = DSQRT(T1/T2)
         IF (ABS(TEMP) .GT. TOLF) THEN
            IFLAG = 0
         ENDIF
      ENDIF
C
C-----TEST FOR DISPLACEMENT TOLERANCE
      T1 = DOTPRD(Q,Q,NEQ)
      T2 = DOTPRD(QT,QT,NEQ)
      IF (T2 .NE. 0.0) THEN
         TEMP = DSQRT(T1/T2)
         IF (ABS(TEMP) .GT. TOLD) THEN
            IFLAG = 0
         ENDIF
      ENDIF
      IF (IFLAG .EQ. 1) PRINT *,'CONVERGENCE'
      RETURN
      END
```

```
C*****************************************************************************
C*                          ABAQUS CONVERGENCE TESTS                         *
C*****************************************************************************
      SUBROUTINE ABQCNV(QR,JCODE,NJ,TOLF,TOLM,IFLAG)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION QR(*),JCODE(3,*)
C
C-----TEST FOR CONVERGENCE AFTER ABAQUS, PTOL,MTOL METHOD
C     F < PTOL=TOLF, M < MTOL=TOLM
C
      IFLAG = 1
      DO 20 J = 1,NJ
         DO 10 K = 1,3
            JK = JCODE(K,J)
            IF (JK .GT. 0) THEN
               IF (K .NE. 3) THEN
                 TOL = TOLF
               ELSE
                 TOL = TOLM
               ENDIF
               IF (DABS(QR(JK)) .GT. TOL) THEN
                  IFLAG = 0
                  RETURN
               ENDIF
            ENDIF
   10    CONTINUE
   20 CONTINUE
      RETURN
      END
```