Comparison of Modified Riks/Wempner and Homotopy Methods

by

B. S. Sunku

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute & State University

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Civil Engineering

Approved:

S. M. Holzer, Chairman

L. T. Watson

T. Kuppusamy

September, 1991

Blacksburg, Virginia

COMPARISON OF MODIFIED RIKS/WEMPNER AND HOMOTOPY METHODS

by

Balaji Suresh Sunku

Committee Chairman: Siegfried M. Holzer
Civil Engineering

(ABSTRACT)

A structured program has been developed to track the equilibrium path of geometrically nonlinear space structures by the modified Riks/Wempner method. The sparse normal flow code in 'HOMPACK' is used for tracking the equilibrium paths by the homotopy method. Two subroutines were written as required by HOMPACK.

Four different structures were analyzed by these two programs. Comparison of the two methods has been carried out based on the number of Jacobian matrix evaluations and the CPU time used by the programs.

The Riks/Wempner program successfully traced the equilibrium path through the limit points and bifurcation points for all the four structures analyzed while the homotopy program could not trace the complete path for two of the structures. It has been concluded that the sparse normal flow code of HOMPACK needs to be modified.

Acknowledgements

The author expresses his sincere appreciation to Dr. Siegfried M. Holzer for his guidance, encouragement, unlimited patience and helpful suggestions.

Special thanks go to Dr. Layne T. Watson for his many helpful discussions and his serving on the committee.

Appreciation is also extended to Dr. T. Kuppusamy for his review of thesis and his serving on the committee.

Discussions with fellow graduate students, especially and , are also greatly appreciated.

The author would like to thank his beloved family and friends. Without their love, support and constant encouragement this work would have been impossible.

Table of Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Purpose and Scope

The purpose of this study is to compare two solution algorithms in the analysis of geometrically nonlinear space trusses. A program has been developed, along the lines of Holzer's(1990) frame program, to trace the equilibrium path of space trusses by the modified Riks/Wempner method. The modified Riks/Wempner method has been found to successfully trace the post buckling paths(Wempner, 1971; Riks, 1979; Crisfield, 1981). The second algorithm used in the study is a homotopy method. The original program, the sparse normal flow code, is taken from HOMPACK(1987). Two user-written subroutines as required were provided to this program. The comparison is carried out by analyzing four different structures by these two methods.

1.2 Literature Review

Wempner(1971) introduced the generalized arc length approach in tracking the equilibrium paths beyond critical points. Riks(1979) also

used the same approach in his study on snapping and buckling problems. Crisfield(1981) proposed a modification to the Riks/Wempner approach so that it is suitable for use with the finite element method.

Vu(1981) used the modified Riks/Wempner method in his study of geometrically nonlinear space trusses and concluded that it can be used to trace highly nonlinear equilibrium paths in either pre-buckling or post buckling range with precision. Borst(1987) used the arc length methods in his study of post-bifurcation and post-failure behavior of strain softening solids. Forde and Steimer(1987) proposed another arc length procedure based on orthogonality principles which provided the same results as Crisfield's explicit iteration procedure with reduction in computation effort.

The modified Riks/Wempner method was used for solving nonlinear moving contact problems by Wu(1986). This method was also used by Sage(1986) to investigate the nonlinear behavior exhibited by cable structures. Kouhia and Mikkola(1989) used Riks/Wempner approach in tracking the equilibrium path beyond critical points and compared it against Fried's(1984) orthogonal trajectory method. Clarke and Hancock(1990) also compared the Riks/Wempner approach against other incremental/iterative strategies for nonlinear analyses.

The homotopy method was used for nonlinear analysis of space trusses by Hansen(1982), and he concluded that this method is capable of

successfully tracking nonlinear equilibrium paths. The software package HOMPACK(Watson, 1986) used in this study was developed in 1986. The part of the HOMPACK code used here is different from the one used by Hansen. Two user subroutines were written to use with HOMPACK the program for analyzing the geometrically nonlinear space trusses.

1.3 Overview

The element model used in this study is presented in Chapter 2. The solution methods are discussed in detail in Chapter 3. In Chapter 4, the program development and the tree charts of the programs and the NS (Nassi-Schneiderman) diagrams are presented(Holzer, 1985). The analysis results of four different structures are presented in Chapter 5. The listing of main programs and key subroutines are presented in Appendix B.

CHAPTER 2

Structural Model

2.1 Element Model

The element model used in this study of nonlinear analysis of a space truss is taken from Vu(1981). The force-displacement relations are derived by using the principle of virtual work. The truss is assumed to be geometrically nonlinear and material properties are assumed to be linear and Hooke's law applies.

The global tangent stiffness matrix for the truss member shown in Fig. 2.1 is

$$K^i = \left[\frac{\partial^2 U^i}{\partial D_j \, \partial D_k}\right] \tag{2.1}$$

where U^i is the strain energy of the truss member i and D_j, D_k are the global member-end displacements (j, k = 1, 2, . . . , 6).

The global tangent stiffness matrix for the truss member can also be written as

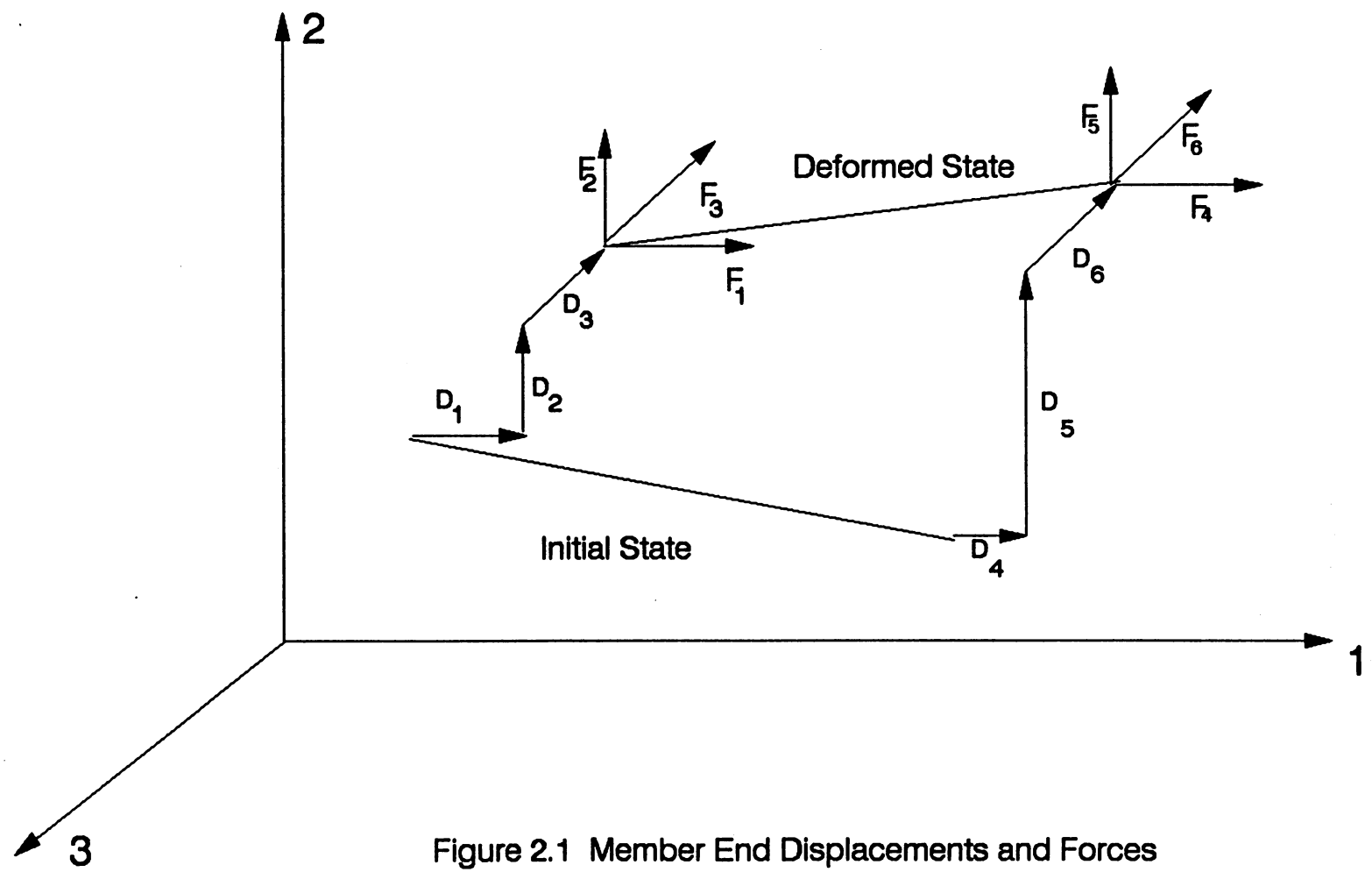

Figure 2.1 Member End Displacements and Forces

$$K^i = \begin{bmatrix} \bar{K}^i & -\bar{K}^i \\ -\bar{K}^i & \bar{K}^i \end{bmatrix} \tag{2.2}$$

Specifically,

$$\bar{K}^i = \left[\frac{\partial^2 U^i}{\partial \Delta_j \, \partial \Delta_k}\right] \qquad j, k = 1, 2, 3 \tag{2.3}$$

The coefficients of $\bar{K}^i$ derived for the member shown in Fig. 2.2 are

$$\bar{K}^i = \begin{bmatrix} K_1 & K_4 & K_6 \\ K_4 & K_2 & K_5 \\ K_6 & K_5 & K_3 \end{bmatrix} \tag{2.4}$$

where

$$K_1 = \gamma_0 \left(c_1^2 + \frac{e}{L}(1-c_1^2)\right) \qquad K_4 = \gamma_0 \left(c_1 c_2 (1-\frac{e}{L})\right)$$

$$K_2 = \gamma_0 \left(c_2^2 + \frac{e}{L}(1-c_2^2)\right) \qquad K_5 = \gamma_0 \left(c_2 c_3 (1-\frac{e}{L})\right) \tag{2.5}$$

$$K_3 = \gamma_0 \left(c_3^2 + \frac{e}{L}(1-c_3^2)\right) \qquad K_6 = \gamma_0 \left(c_1 c_3 (1-\frac{e}{L})\right)$$

and

$\gamma_o = \frac{AE}{L_0}$, A is the cross sectional area of the member, E is the Young's modulus of elasticity, L_o is the initial length of the member, L is the deformed length of the member, e is the elongation of the

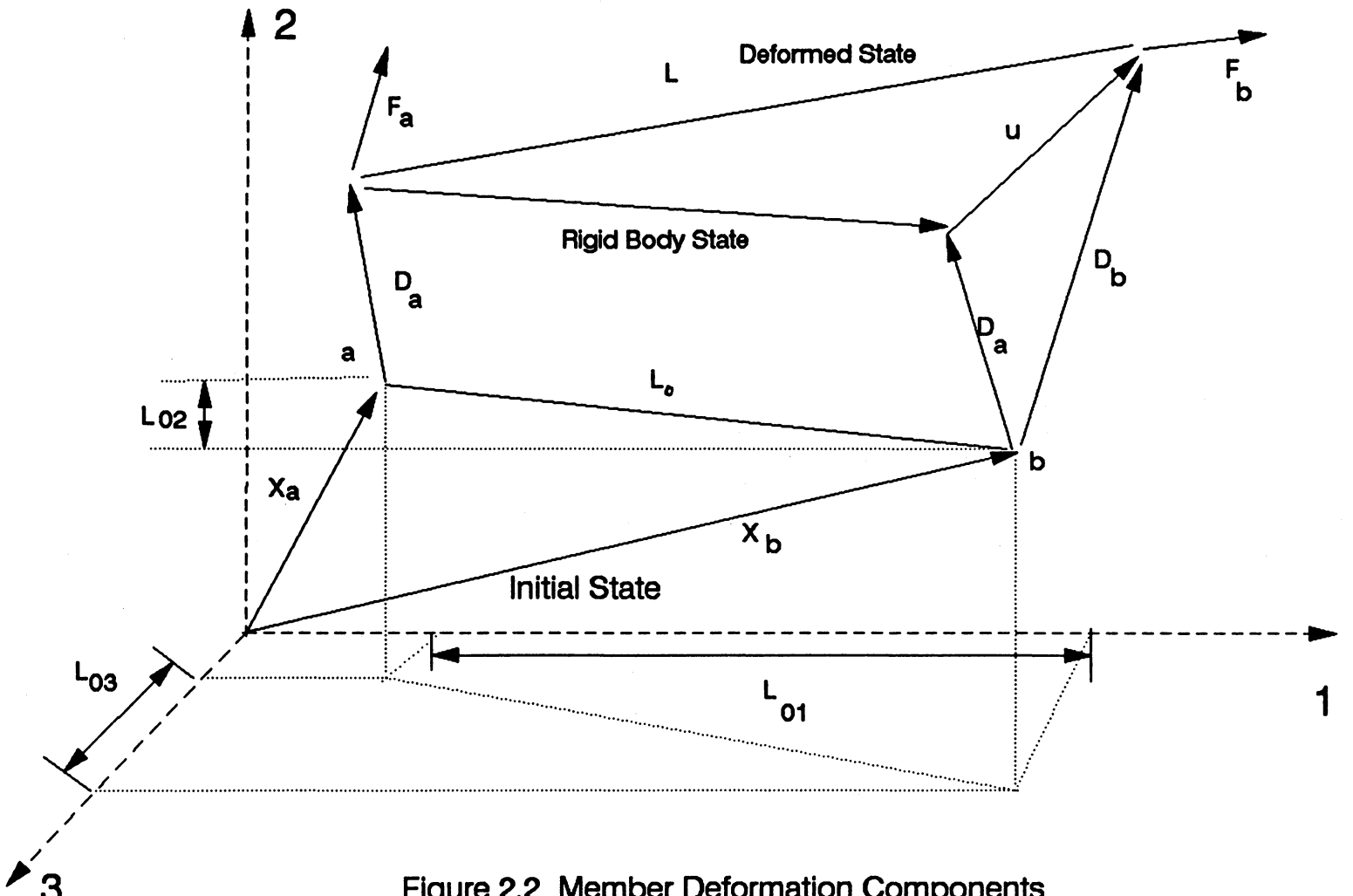

Figure 2.2 Member Deformation Components

element, and c_1, c_2, c_3 are the direction cosines of the member in the deformed configuration.

The tangent stiffness matrix of a structure is obtained by summing up the tangent stiffness matrices of each member of the structure.

CHAPTER 3

Solution of Nonlinear Equations

The nonlinear equation of equilibrium may be written as (Holzer, 1990; Bathe, 1989)

$$\mathbf{Q} - \mathbf{F} = 0 \tag{3.1}$$

where $\mathbf{Q}$ is the external nodal force vector; and it may consist of applied nodal forces $\mathbf{Q}_a$, surface tractions $\mathbf{Q}_p$ and inertia forces $\mathbf{Q}_i$. $\mathbf{F}$ is the internal nodal force vector that equilibrates the configuration $(\mathbf{q}, \mathbf{Q})$(Fig. 3.1). The solution to Eq. 3.1 is the equilibrium path in the (n+1) dimensional load-displacement space(Fig. 3.1).

3.1 Solution Methods

A solution method aims at developing an accurate approximation of the load-displacement curve for the system under a given load distribution. These solution methods generally involve incremental/iterative procedures. An incremental procedure is

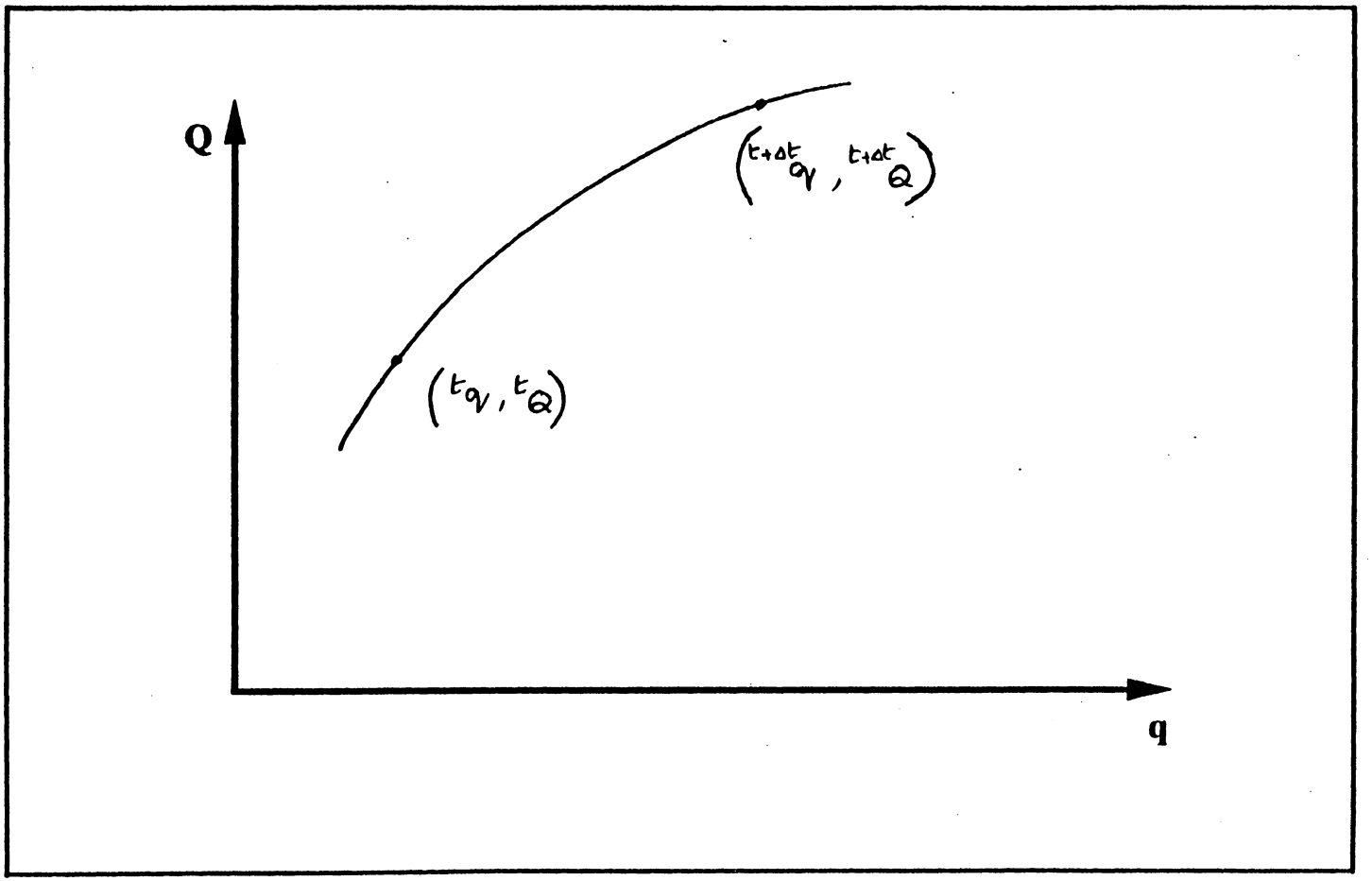

Figure 3.1 Single Degree-of-Freedom Model: Equilibrium Path

designed to trace an equilibrium path in terms of a sequence of successive, but distinct points(Riks, 1979). Each point offers a means to construct the starting configuration for the next point to be computed. The accuracy of the iterations can be controlled by keeping the distance between the known and still unknown point within certain bounds.

The two solution methods used in this study are: the modified Riks/Wempner method and the Homotopy method. The Newton-Raphson method which is contained in the modified Riks/Wempner method is also briefly discussed here. The computer program also has an option to use the Newton-Raphson method.

3.1.1 System Model

The solution to Eq. 3.1 is obtained by incremental/iterative procedure in which Eq. 3.1 is linearized. The linearized equation may be written as(Vu, 1981)

$$K^k \Delta q^k = \Delta\lambda^k \bar{Q} + R^k \tag{3.2}$$

where

$$q^{k+1} = q^k + \Delta q^k \tag{3.3}$$

$$\lambda^{k+1} = \lambda^k + \Delta\lambda^k \tag{3.4}$$

In Eqs. 3.2 to 3.4, K^k is the tangent stiffness matrix of the structure at q^k, $\bar{Q}$ is the constant load distribution vector, R^k is the unbalanced force vector at (q^k,λ^k), q^k is the generalized displacement vector, and λ^k is the loading parameter.

Equations 3.2 to 3.4 are the recurrence relations of the Riks/Wempner method. In the Newton-Raphson method, the load level is constant during iteration. Accordingly Eq. 3.2 reduces to

$$K^k \Delta q^k = R^k \tag{3.5}$$

3.1.2 Newton-Raphson Method

The Newton-Raphson method is the most commonly used iterative method. In this method an incremental/iterative analysis is performed to determine the load-displacement path of a given system. A load increment is prescribed and the corresponding displacements are computed by an iterative process. The iterations are continued until the desired convergence is achieved. The process is repeated for each load increment up to the desired load level.

The solution thus involves the generation and factorization of the tangent stiffness matrix K_T^k for each iteration. The Newton-Raphson method cannot trace the equilibrium path through limit points.

3.1.3 Modified Riks/Wempner Method

The basic idea of the Riks/Wempner method is to select a generalized arc length as the independent variable that controls the progress along the equilibrium path(Riks, 1979: Wempner 1971). The length Δs of the tangent to the current equilibrium point is prescribed and the new point is the intersection of the normal to the tangent with the equilibrium path. As an alternative to the iteration on the normal (normal plane in space) Crisfield(1981) recommended iteration on a circle (sphere in space) with the center at the current equilibrium point and radius Δs. Crisfield also suggested the reformulation and factorization of the tangent stiffness matrix only at the beginning of each load increment. In this study iteration on normal plane is used.

The modified Riks/Wempner method(Holzer et al.,1981)is graphically illustrated in Figs. 3.2 and 3.3 for a single degree of freedom system which is based on the work by Ramm(1980).

Let a point in the load-deflection space be denoted by the vector

$$r = \begin{bmatrix} q \\ \lambda \end{bmatrix} \tag{3.6}$$

$\Delta\lambda$ be the initial load increment.

At known equilibrium point

$$K\Delta q = \Delta\lambda\bar{Q} \tag{3.7}$$

where K is tangent stiffness matrix and $\bar{Q}$ is the constant load distribution vector and the unbalanced force vector R = 0. The tangent vector, t at the starting point is defined as

$$t = \Delta r = \begin{bmatrix} \Delta q \\ \Delta\lambda \end{bmatrix} \tag{3.8}$$

and

$$\Delta s = |t| = \sqrt{\left[\Delta q.\ \Delta q + (\Delta\lambda)^2\right]} \tag{3.9}$$

where $|t| = (t.t)^{\frac{1}{2}}$ = length of tangent vector at t, and $t.t = t^T.t$, inner product of t with itself.

Let $\Delta q = \Delta\lambda\Delta q^I$ (3.10)

where Δq^I is the solution to

$$K\Delta q^I = \bar{Q} \tag{3.11}$$

$$\Delta s = \Delta\lambda \sqrt{(\Delta q^I.\Delta q^I+1)} \tag{3.12}$$

In the subsequent steps the arc length is controlled by scaling as shown below

$$\Delta\hat{s} = \Delta s \left(\frac{\hat{\bar{I}}}{\bar{I}}\right)^{\frac{1}{2}} \tag{3.13}$$

where, $\hat{I}$ is the number of desired iterations and I is the number of iterations required in the previous step

First step

Consider the Fig. 3.2. 0 is the known equilibrium point and N is the new equilibrium point. The first trial solution is defined by the tangent vector at known equilibrium point 0

$$t^0 = \Delta r^0 = \begin{bmatrix} \Delta q^0 \\ \Delta \lambda^0 \end{bmatrix} \tag{3.14}$$

$$q^1 = q^0 + \Delta q^0 \tag{3.15}$$

$$\lambda^1 = \lambda^0 + \Delta \lambda^0 \tag{3.16}$$

where

$$\Delta q^0 = \Delta \lambda^0 \Delta q^{0I} \tag{3.17}$$

and Δq^{0I} is the solution to

$$K^0 \Delta q^{0I} = \bar{Q} \tag{3.18}$$

and $\Delta \lambda^0$ is obtained from Eq. 3.9

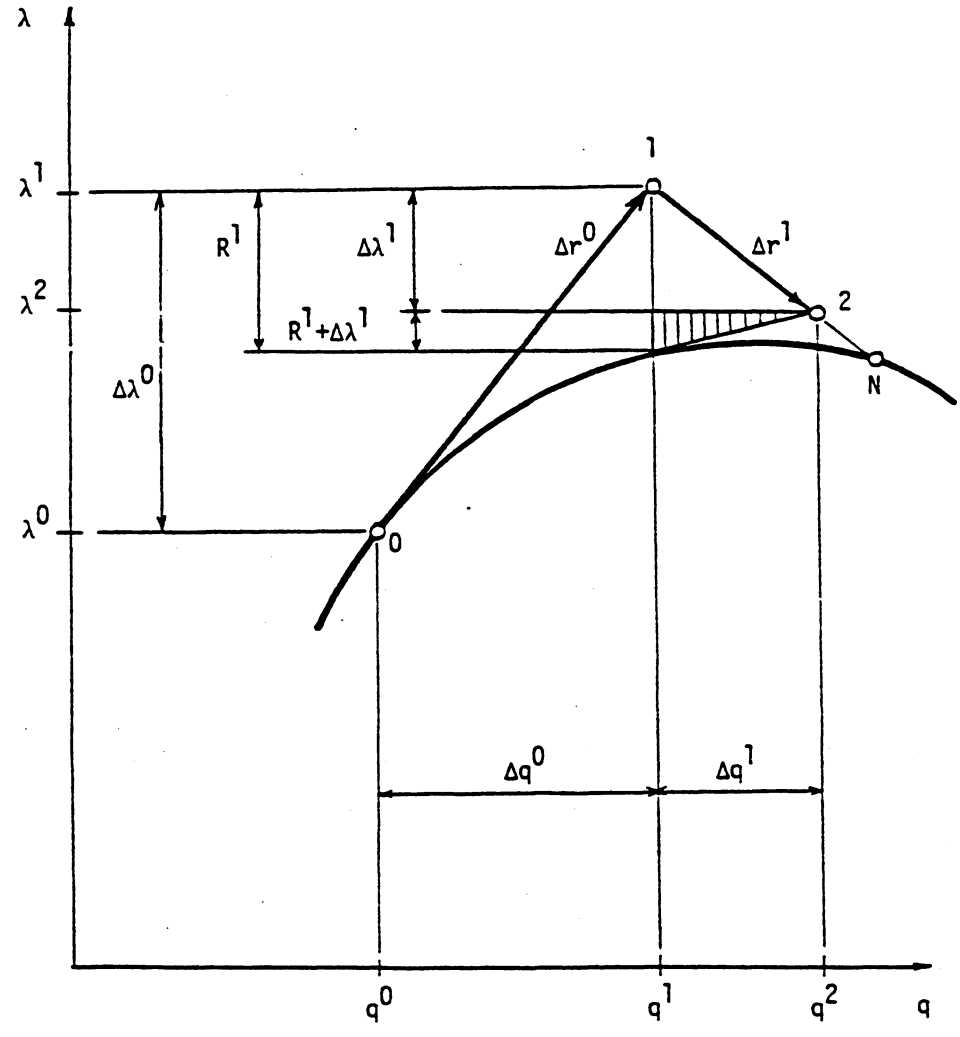

Figure 3.2 Modified Riks/Wempner Method

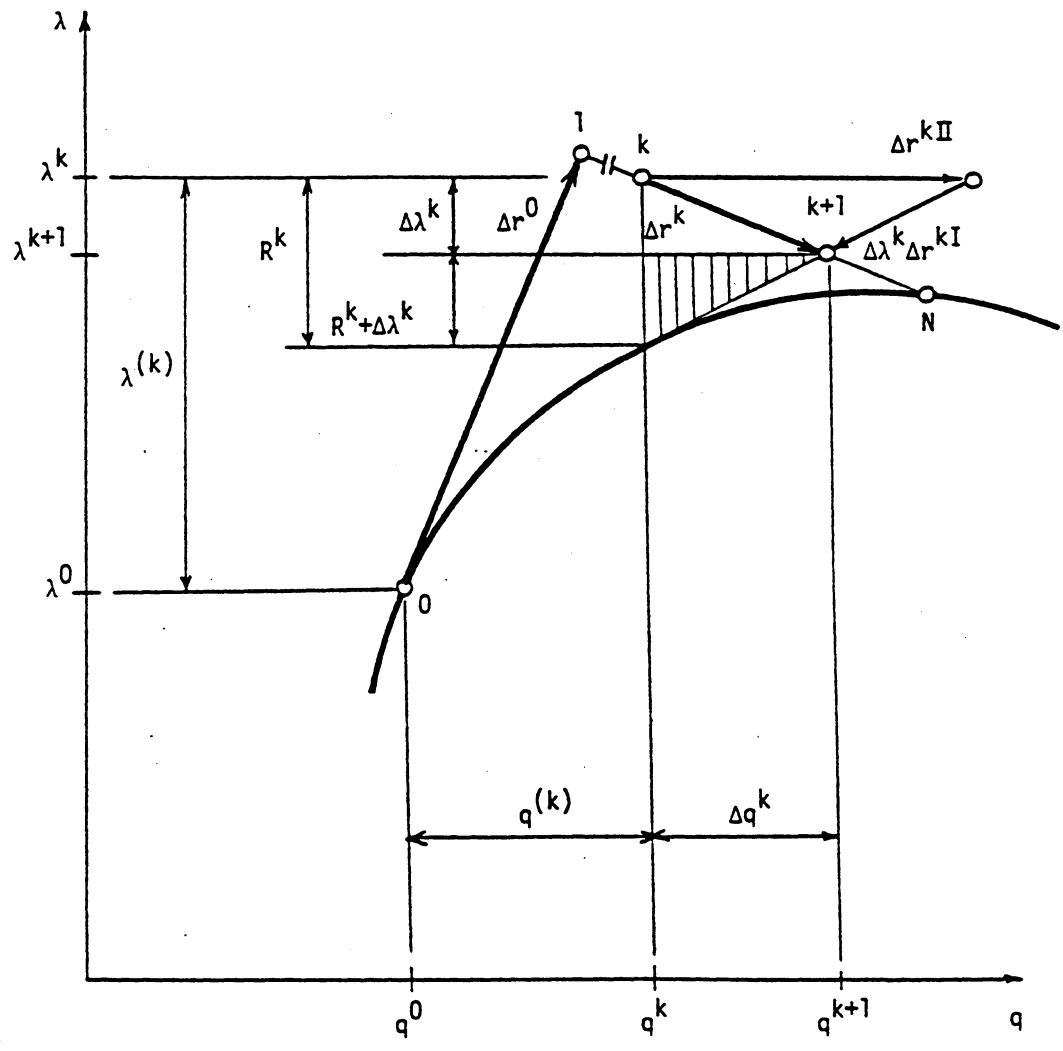

Figure 3.3 Iteration on Normal Plane

$$\Delta\lambda^0 = \pm \frac{\Delta\hat{s}}{\sqrt{(\Delta q^{0I} \Delta q^{0I} + 1)}} = \pm \frac{\Delta\hat{S}}{\left[\Delta q^{0I} \Delta q^{0I} + 1\right]^{1/2}} \qquad (3.19)$$

The positive sign indicates loading and negative sign indicates unloading

Iteration on normal plane

In Fig. 3.3, the incremental vector Δr^k from k to k+1 is decomposed as

$$\Delta r^k = \begin{bmatrix} \Delta q^k \\ \Delta\lambda^k \end{bmatrix} = \Delta\lambda^k \begin{bmatrix} \Delta q^{kI} \\ 1 \end{bmatrix} + \begin{bmatrix} \Delta q^{kII} \\ 0 \end{bmatrix} \qquad (3.20)$$

which implies

$$\Delta q^k = \Delta\lambda^k . \Delta q^{kI} + \Delta q^{kII} \qquad (3.21)$$

where Δq^{kI} and Δq^{kII} are the solutions of two sets of equations of equilibrium

$$K_T^k . \Delta q^{kI} = \bar{Q} \qquad (3.22)$$

$$K_T^k . \Delta q^{kII} = R^k \qquad (3.23)$$

To satisfy the condition of normality, the scalar product of tangent vector t^0 and the vector Δr^k containing the unknown load and

displacement increments must vanish

$$t^0 \cdot \Delta r^k = 0 \tag{3.24}$$

In matrix form the above equation may be written as

$$\Delta q^0 \Delta q^k + \Delta\lambda^0 \Delta\lambda^k = 0 \tag{3.25}$$

The unknown load increment $\Delta\lambda^k$ is determined by substituting Eq.. 3.21 into Eq. 3.25

$$\Delta\lambda^k = - \frac{\Delta q^0 \Delta q^{kII}}{\Delta q^0 . \Delta q^{kI} + \Delta\lambda^0} \tag{3.26}$$

An improved trial configuration is obtained from the equations

$$q^{k+1} = q^k + \Delta q^k \tag{3.27}$$

$$\lambda^{k+1} = \lambda^k + \Delta\lambda^k \tag{3.28}$$

This method successfully traces the response of the structure beyond the limit point.

Determination of the sign of the load increment

The value of the load increment $\Delta\lambda$ is initially suggested by the user

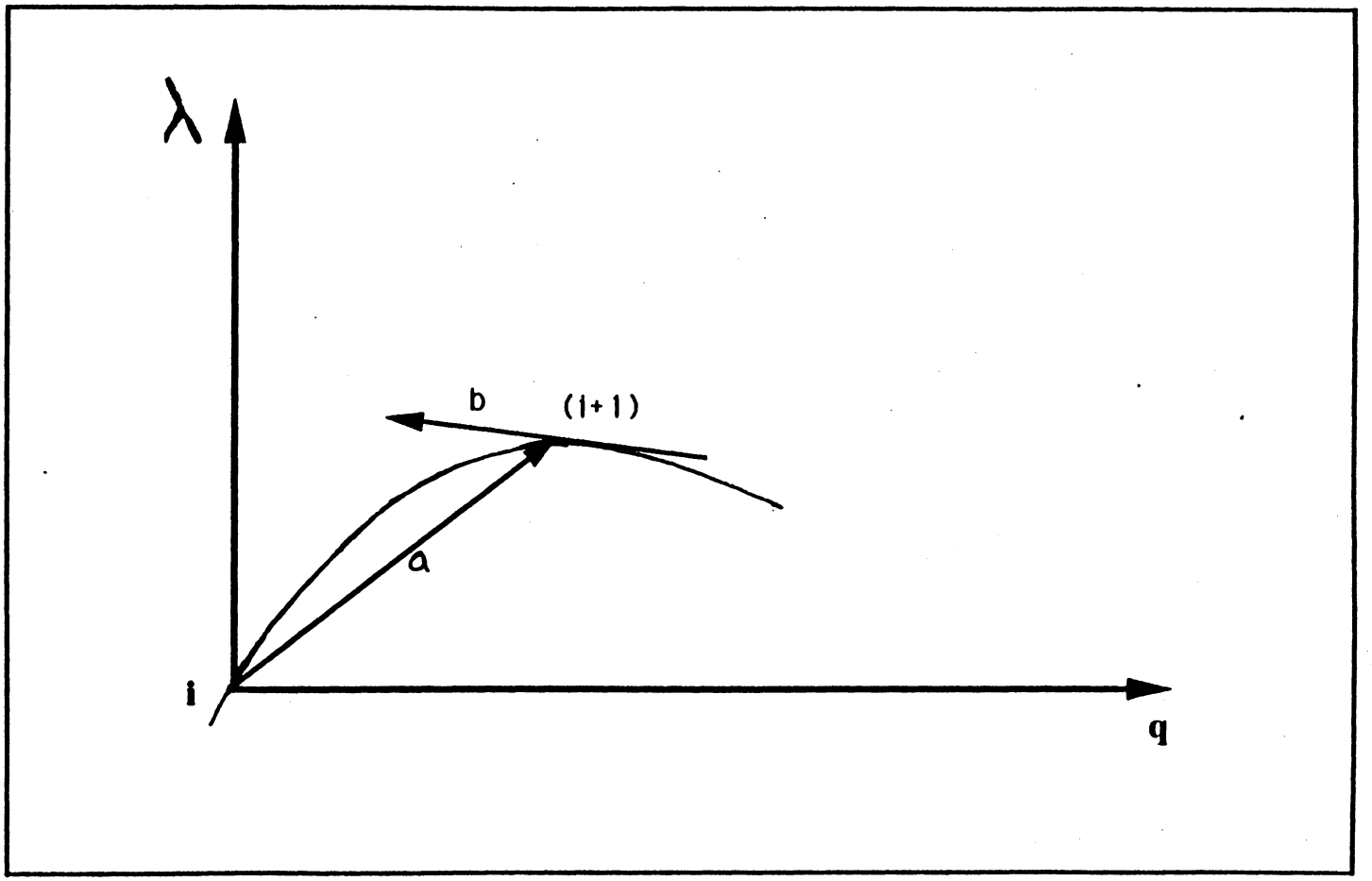

Figure 3.4 Determination of the Sign of Load Increment

and is adjusted by the program's usual automatic load incrementation algorithm. In subsequent iterations an ambiguity arises regarding the sign of $\Delta\lambda$ due to the presence of a square root sign in the equation. The sign of $\Delta\lambda$ in the direction of response along the tangent line is chosen so that the dot product of $(\Delta^{i+1}q^0, \Delta^{i+1}\lambda^0)$ and the solution to the previous increment $(\Delta^i q, \Delta^i\lambda)$ is always positive(Hibbit et al., 1989). This is illustrated with an example.

In Fig. 3.4, let the point 0 represent the current equilibrium configuration of the structure after i iterations. The solution at the current configuration is represented by a vector

$$a = (10, 1)$$

and the increments in load and displacements are represented by the vector

$$b = \Delta\lambda(-3, 1)$$

To make sure that the response is progressing in the right direction the product

$$a.b = (10, 1) \; . \; \Delta\lambda(-3, 1) > 0$$

$$\Delta\lambda(-29) > 0$$

which implies $\Delta\lambda < 0$.

Thus the correct the sign of the load increment is determined.

3.1.4 Convergence Criteria

The iterative procedures used in the solution process are essentially trial and error methods. Hence, the convergence criteria used in the termination of the iterative process should be effective and feasible. In this study, the two solution variables suggested by Bathe and Cimento(1979) are used for convergence criteria. They are displacements and forces.

The convergence criteria for displacement is stated as :

$$\frac{||\Delta\ {}^{n+1}q^{i}||}{||{}^{n+1}q||} \quad <= \quad \epsilon_D \tag{3.29}$$

where $||\Delta\ {}^{n+1}q^{i}||$ is the Euclidian norm of incremental displacements in the i^{th} iteration of $(n+1)^{th}$ load step, $||{}^{n+1}q||$ is the Euclidian norm of the total displacements at $(n+1)^{th}$ load step. Since ${}^{n+1}q$ is not known in advance, it is approximated by ${}^{n+1}q^{i}$, and ϵ_D is a prescribed displacement convergence tolerance

The unbalanced force criterion requires that the norm of unbalanced force be with in a preset tolerance ϵ_F of the original load increment,

$$\frac{||{}^{n+1}Q^{i} - {}^{n+1}F^{i}||}{||{}^{n+1}Q^{1} - {}^{n+1}F||} \quad \leq \quad \epsilon_F$$

where ${}^{n+1}Q^{1}$ and ${}^{n+1}Q^{i}$ are the total external force vector at the

first and I^{th} iterations of the $(n+1)^{th}$ load step respectively, ${}^{n+1}F^{i}$ is the internal nodal force vector in i^{th} iteration of $(n+1)^{th}$ load step, ${}^{n}F$ is the internal nodal force vector obtained in the nth load step (i.e. the previous load step), and ϵ_F is prescribed force convergence tolerance.

In this study both ϵ_D and ϵ_F are taken as 10^{-4}

3.1.5 Homotopy Method

The second method used in this study of space trusses is a homotopy method. In the Newton-Raphson method in the vicinity of limit points several problems arise: (i) The number of iterations required to converge increases (ii) Constant lambda will miss the curve (iii) No points can be obtained beyond limit points (iv) At the limit point the tangent stiffness matrix becomes singular. The Riks/Wempner method, though able to trace past the limit point, destroys the symmetry and bandedness of the stiffness matrix. The modified version of the Riks/Wempner method preserves the symmetry and bandedness, but it still fails when the tangent stiffness matrix becomes singular.

The homotopy method is also based on the arc length as the independent variable (Watson et al., 1983). If a large number of points on the curve and very accurate values for the the limit points are desired

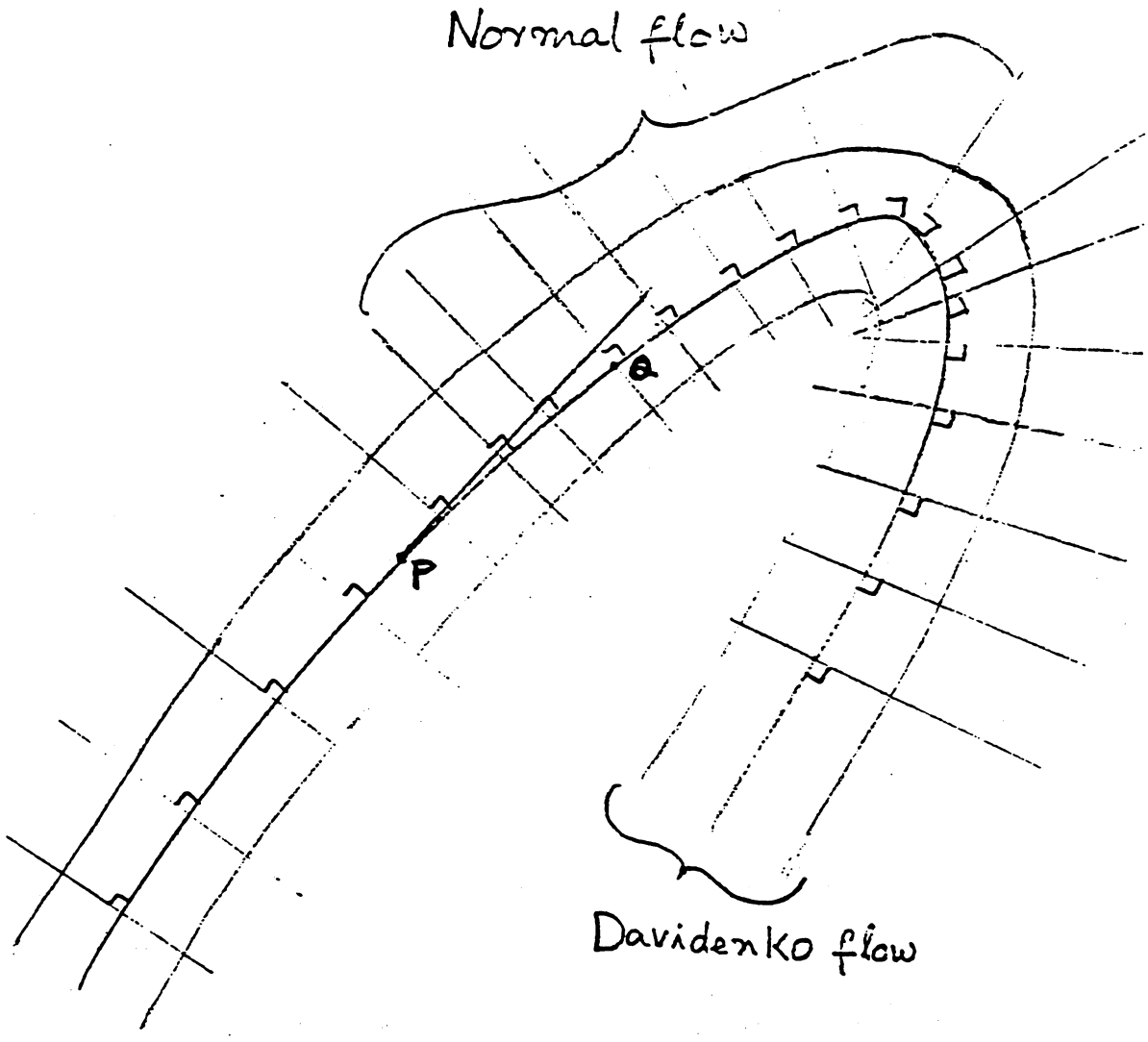

Figure 3.5 Homotopy Method

the homotopy method used in this study guarantees the non-singularity of the stiffness matrix. The homotopy method has no inherent instability near a limit point, in fact, a limit point is no different from (as far as the method is concerned) from any other point on the load-displacement curve.

The homotopy method used in this study uses the sparse normal flow algorithm of the HOMPACK. The family of zero curves, which contains the desired equilibrium path is called the 'Davidenko flow'(Fig. 3.5). The family of curves which are normal to the Davidenko flow is called the 'normal flow'. If a point P is known on the equilibrium path the new equilibrium configuration Q is obtained by first proceeding a prescribed arc length along the tangent to the curve at point P and then tracing to the new point along the normal to the curve(Watson et al., 1983).

The mathematical model of space trusses and the solution procedure are discussed by Watson et al., (1983). The application of this method in the analysis of space trusses was studied earlier by Hansen(1981). Some of the recommendations of Hansen for an improvement of the homotopy program have been implemented in this study.

CHAPTER 4

The Computer Program

In this chapter the development of the computer programs and a brief description of the functions of the subroutines are presented. The programming language used is standard FORTRAN 77. The listing of the main programs and key subroutines is presented in Appendix B, which includes the format required for the input and sample input files.

4.1 Program Development

The tree-charts of the programs are shown in Figs. 4.1 and 4.2.

The structure of the program I shown in Fig. 4.1 follows the form presented by Holzer(1990) in his work on structural analysis. This program is an extension of the program developed by Ahmed(1988) for nonlinear analysis of plane trusses and performs the nonlinear analysis of space trusses by the Newton-Raphson method and the modified Riks-Wempner method.

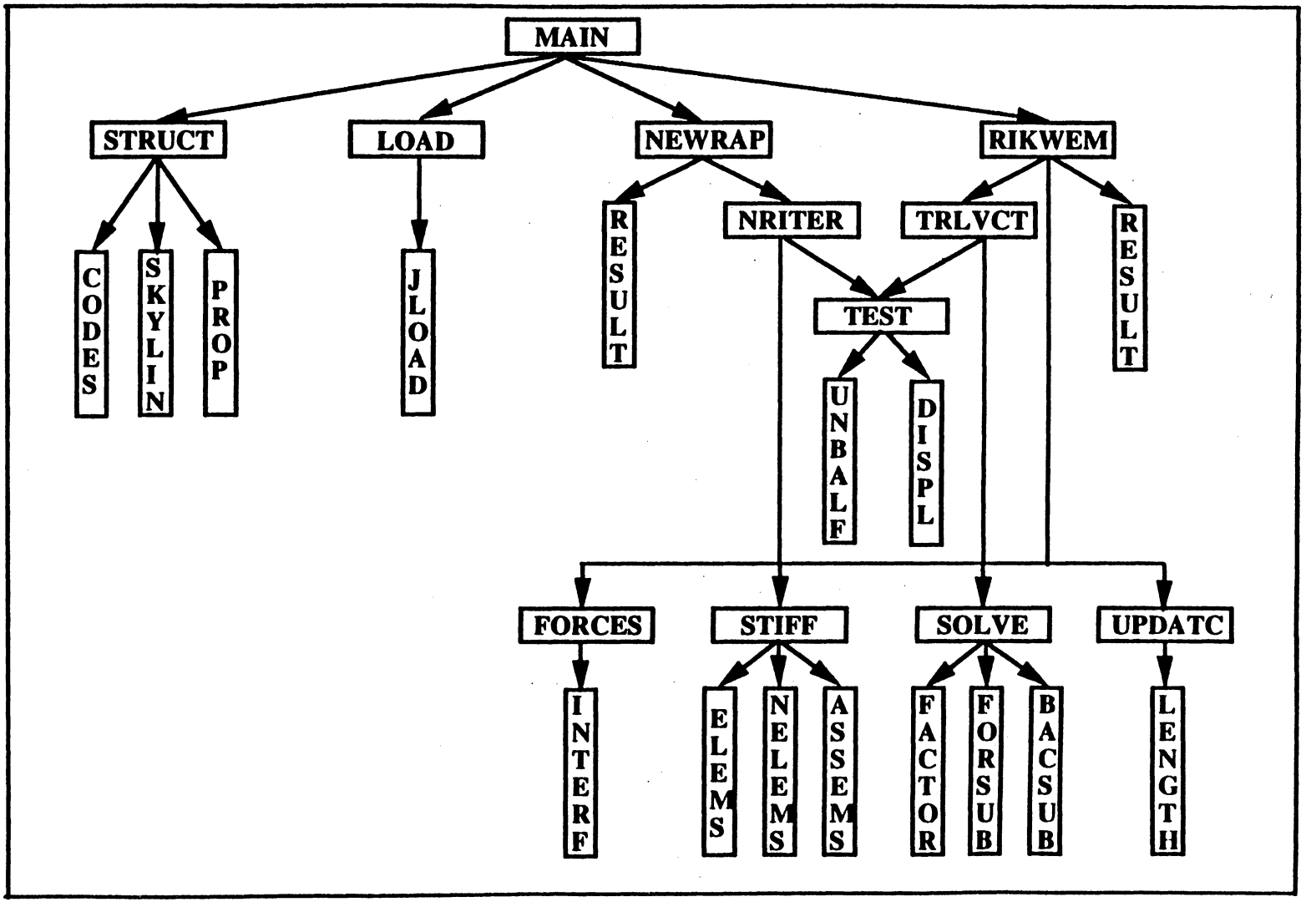

Figure 4.1 Program I : Tree Chart

The program II shown in Fig. 4.2 performs the nonlinear analysis of space trusses by a homotopy method. This program uses the sparse normal flow algorithm of HOMPACK, a software package which provides three qualitatively different algorithms for tracing the homotopy zero curve(Watson, 1989). In this study the sparse normal flow algorithm is used to trace the load-displacement path of space trusses. Two subroutines RHO and RHOJS are written as required by HOMPACK. These subroutines call the other subroutines, which are modified forms of the subroutines used in program I.

The STRUCT and LOAD subroutines are taken from Holzer's(1990) Frame program and are modified to suit the present needs. The solution routines(SOLVE, FACTOR, FORSUB and BACSUB) used in program I are taken directly from Bathe(1985). In program II the sparse normal flow algorithm solves the equations by the method of conjugate gradients.

4.2 Program Subroutines

The functions of all the subroutines are presented in this section.

Program I (Riks/Wempner program)

MAIN PROGRAM

Reserve memory. Read and echo control variables, the number of

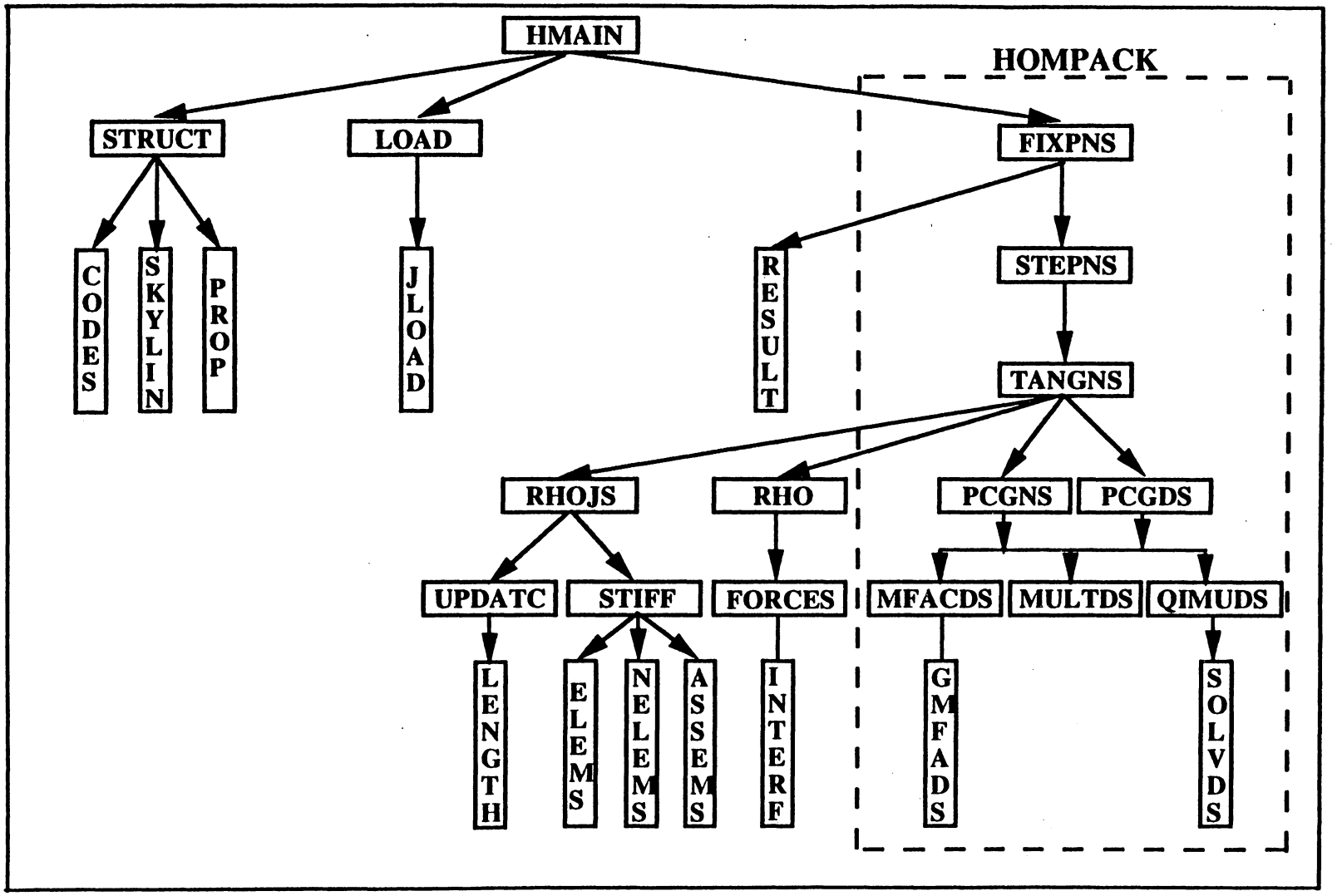

Figure 4.2 Program II : Tree Chart

elements NE, and the number of joints, NJ. Call STRUCT and LOAD. Initialize the initial, maximum, and incremental load factors, tolerance factors and iteration control variables.

Subroutine STRUCT

Read and echo member incidences. Initialize the joint code matrix JCODE and the member code matrix MCODE. Read and echo joint constraints and initialize the corresponding locations of JCODE to zero. Call CODES, SKYLIN, and PROP.

Subroutine CODES

Form the joint code matrix JCODE and the member code matrix MCODE.

Subroutine SKYLIN

Determine the length of the tangent stiffness vector(in skyline storage)SS, the vector which stores the addresses of the main diagonal terms MAXA, the number of the elements below the skyline of each column, KHT.

Subroutine PROP

Read and echo joint coordinates XC. For each element read and echo AREA, EMOD. Compute the element length and direction cosines for each element.

Subroutine LOAD

Initialize the joint load vector Q to Zero. Call JLOAD.

Subroutine JLOAD

Read and echo joint numbers, joint directions and joint forces. Develop the load vector Q.

Subroutine NEWRAP

Compute the proportional load vector using the initial load parameter. Call NRITER and RESULT. Terminate process upon reaching the desired load level or a prescribed displacement or a certain number of load increments.

Subroutine NRITER

Perform the Newton-Raphson iteration for a given load level until the convergence is reached or the maximum number of iterations is exceeded. Call STIFF, SOLVE, FORCES, and TEST.

Subroutine RIKWEM

Compute the generalized arc length. Terminate process upon reaching the desired load level or a prescribed displacement or a certain number of load increments.

Subroutine TRLVCT

Perform iterations to obtain vectors orthogonal to the generalized arc

length. Obtain the next equilibrium configuration by using the two step procedure. Continue iterations until convergence is achieved or the maximum number of iterations is exceeded. Call FORCES, STIFF, SOLVE, DOTPRD, UPDATC, AND RESULT.

Subroutine STIFF

Initialize the system stiffens vector to zero. Call ELEMS, NELEMS, and ASSEMS.

Subroutine ELEMS

Compute the global linear stiffness coefficients.

Subroutine NELEMS

Compute the global nonlinear stiffness coefficients.

Subroutine ASSEMS

Map the element stiffness matrix in to the system stiffness vector using MAXA and MCODE.

Subroutine FORCES

Initialize the FORCE vector to zero. Call INTERF

Subroutine INTERF

Compute the internal forces for the global unconstrained degree-of-freedom of each element in the new configuration.

Subroutine TEST

Test for convergence. Call DISPL and UNBALF.

Subroutine DISPL

Compute the Euclidian vector norm of displacements and test for convergence in displacements.

Subroutine UNBALF

Compute the Euclidian vector norm of unbalanced forces test for convergence in unbalanced forces.

Subroutine UPDATC

Update the joint coordinates XC by adding the current incremental displacements to the previous joint coordinates. Call LENGTH.

Subroutine LENGTH

Compute the element lengths and the new direction cosines of each element.

Subroutine RESULT

Read the degrees of freedom for which the output is required. Print the output.

Program II (Homotopy program)

HMAIN PROGRAM

Read and echo NE, NJ. Call STRUCT and LOAD. Initialize the maximum load increment, tolerance factors, the type of algorithm to be used, the displacement vector, the elongation and the deformed length. Call FIXPNS.

Subroutines FIXPNS, STEPNS, TANGNS, PCGNS, PCGDS, MFACDS, MULTDS, QIMUDS, GMFADS, and SOLVDS are taken from HOMPACK.

Subroutines STRUCT, CODES, SKYLIN, PROP, LOAD, JLOAD, STIFF, ELEMS, NELEMS, ASSEMS, FORCES, INTERF and RESULT have the same function as in Program I

Subroutine RHOJS

Control module to compute the updated coordinates and the stiffness matrix at the current configuration. Calls UPDATC and STIFF.

Subroutine RHO

Control module for computing the vector rho. Calls FORCES.

Subroutine UPDATC

Update the joint coordinates XC by adding the current total displacements to the initial joint coordinates. Call LENGTH.

Subroutine LENGTH

Compute new element length and new direction cosines.

4.3 NS Diagrams

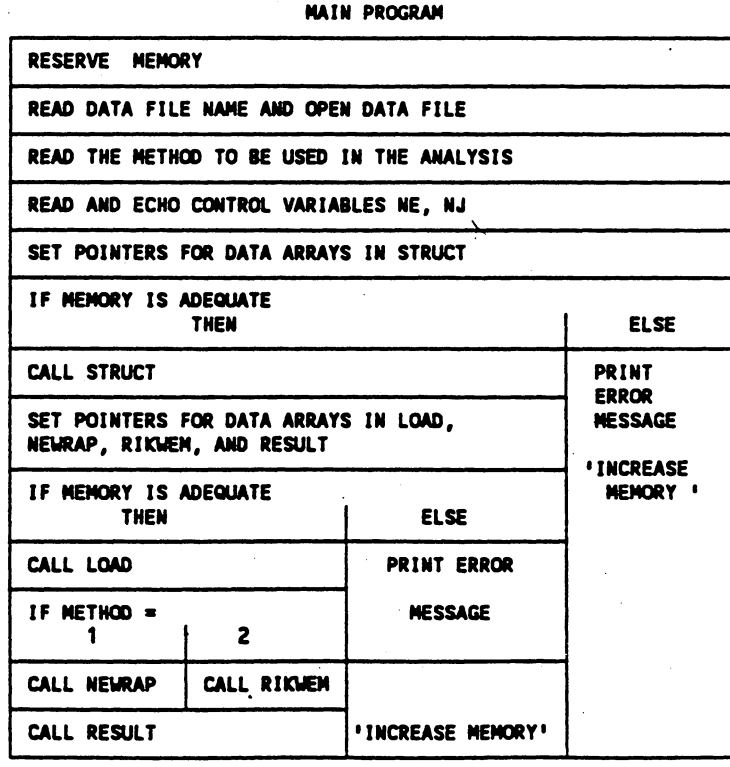

Figure 4.3 Main Program

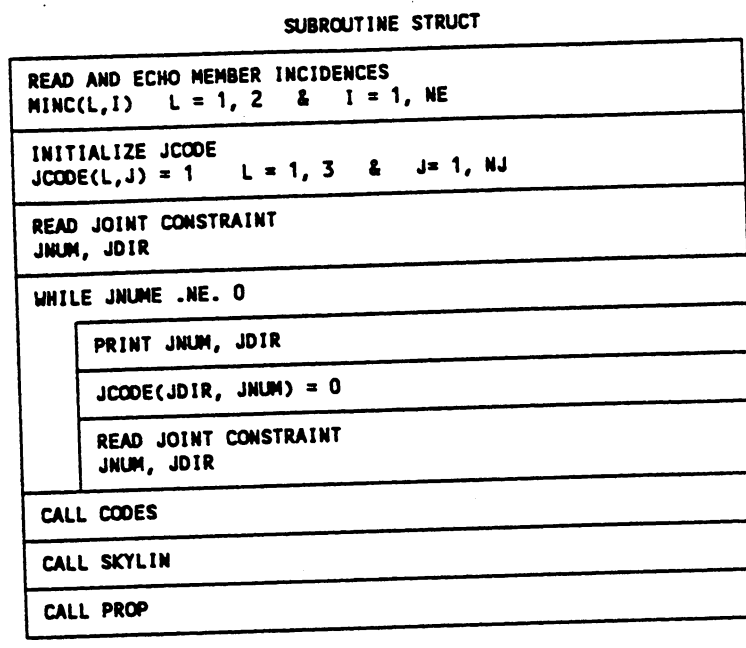

Figure 4.4 Subroutine STRUCT

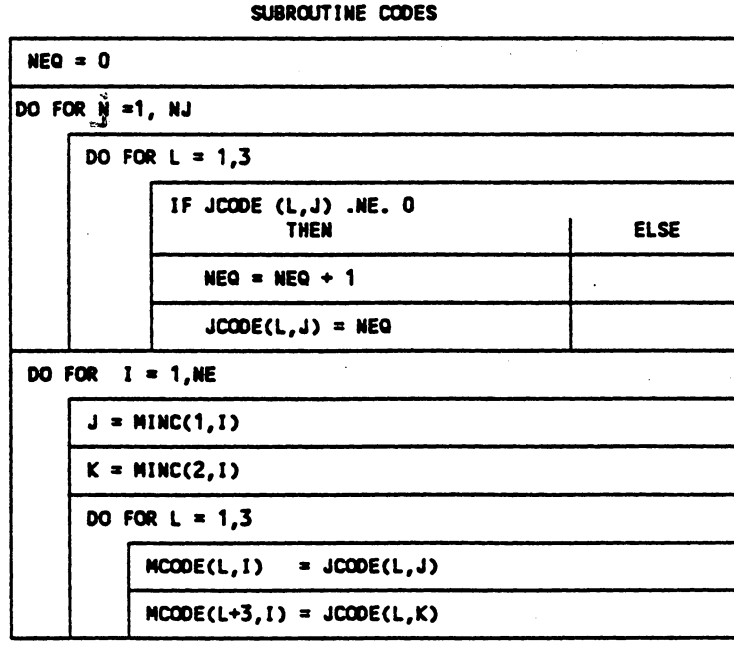

Figure 4.5 Subroutine CODES

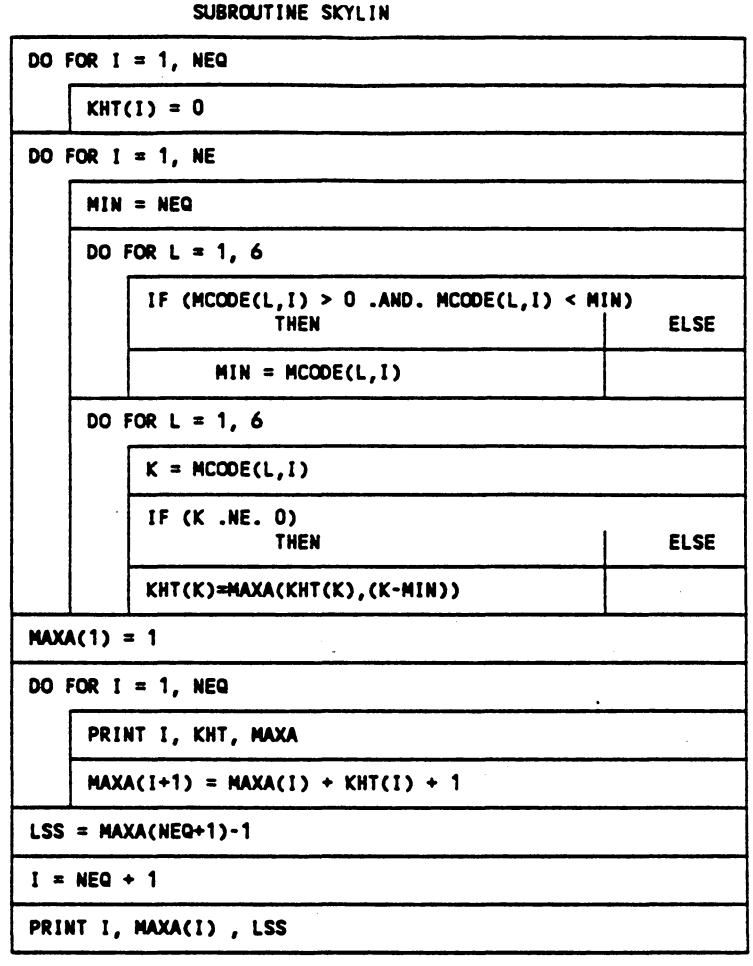

Figure 4.6 Subroutine SKYLIN

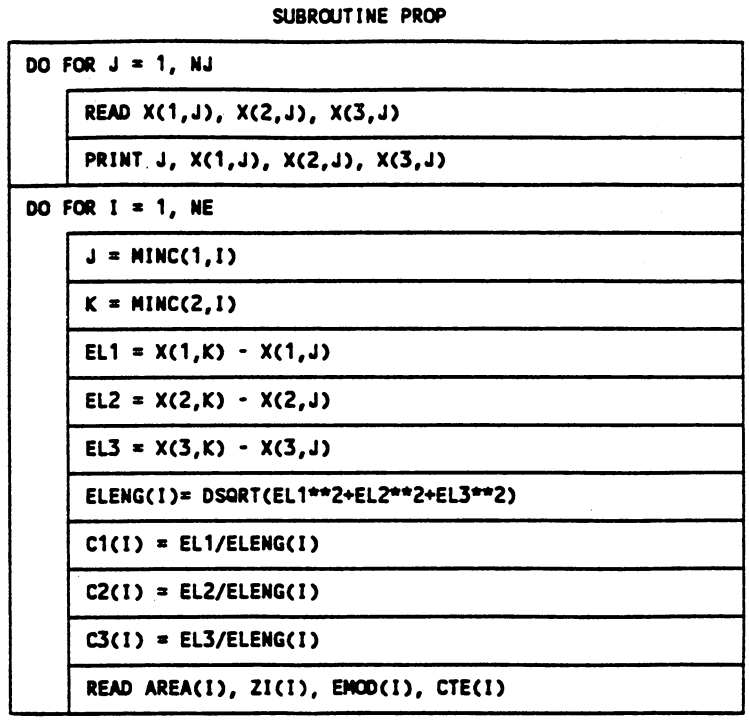

Figure 4.7 Subroutine PROP

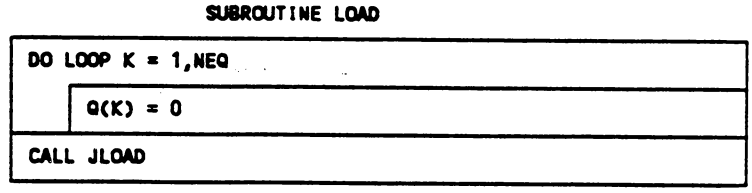

Figure 4.8 Subroutine LOAD

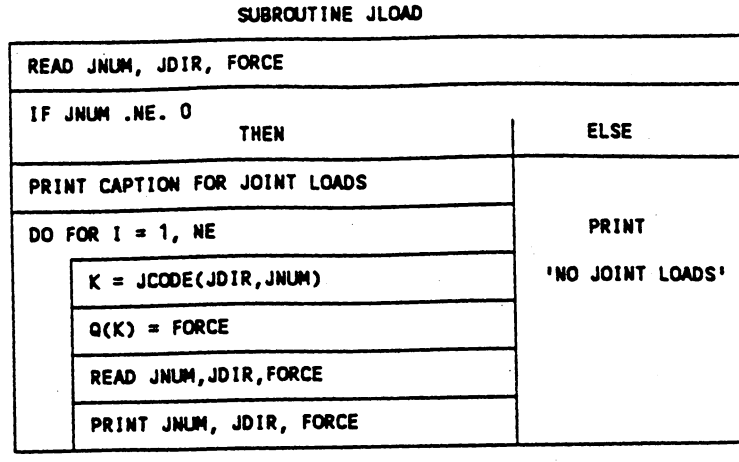

Figure 4.9 Subroutine JLOAD

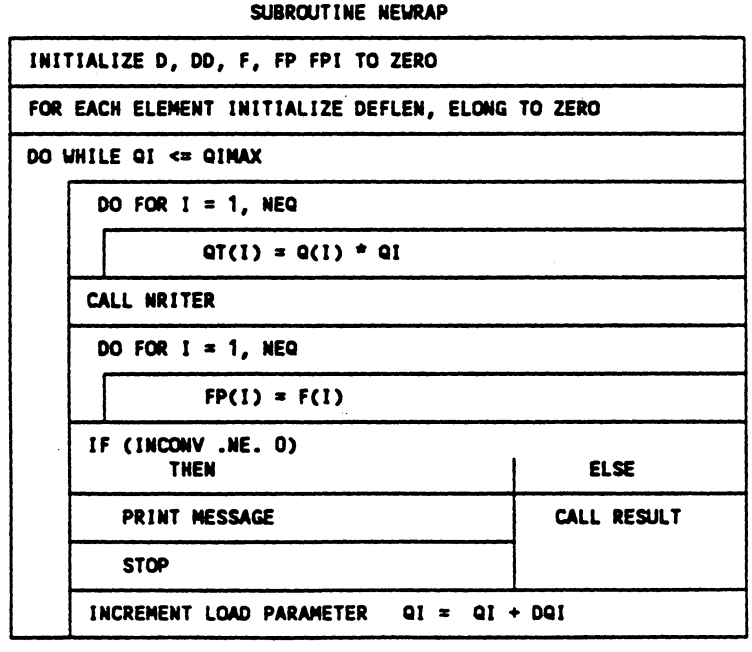

Figure 4.10 Subroutine NEWRAP

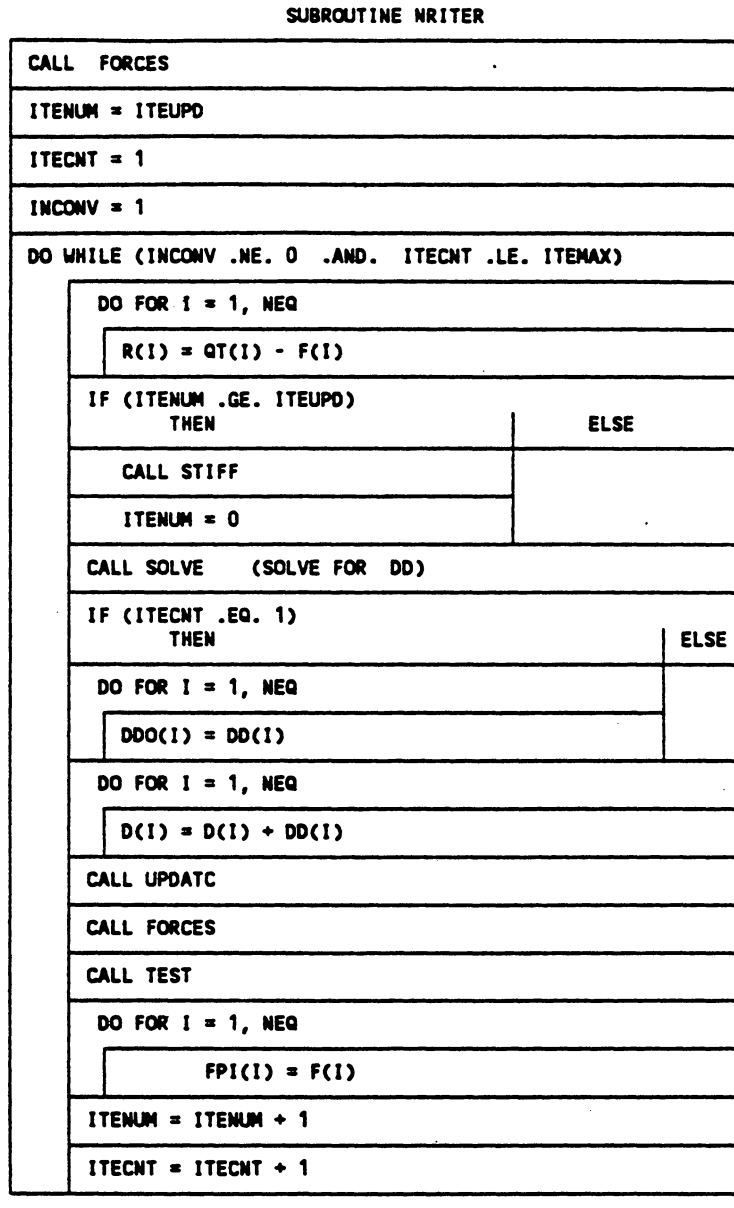

Figure 4.11 Subroutine NRITER

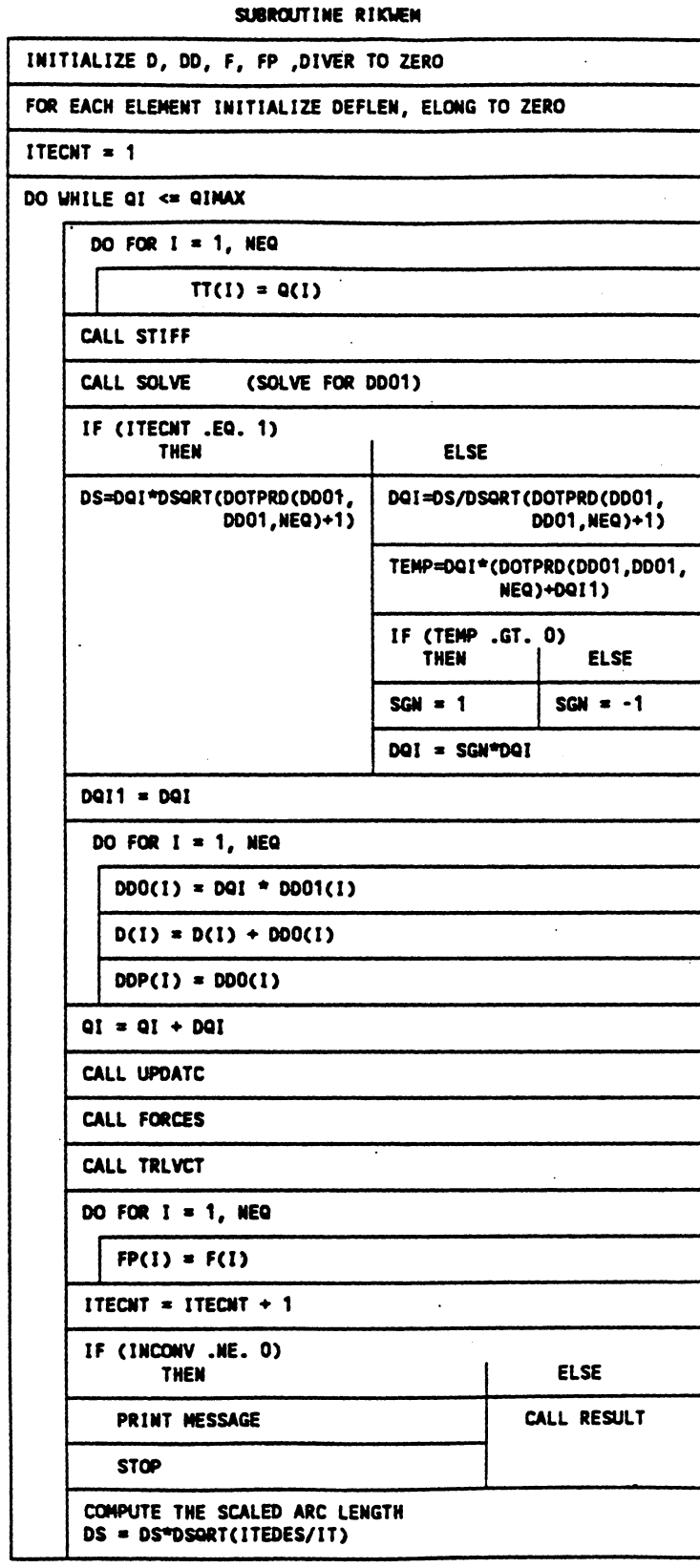

Figure 4.12 Subroutine RIKWEM

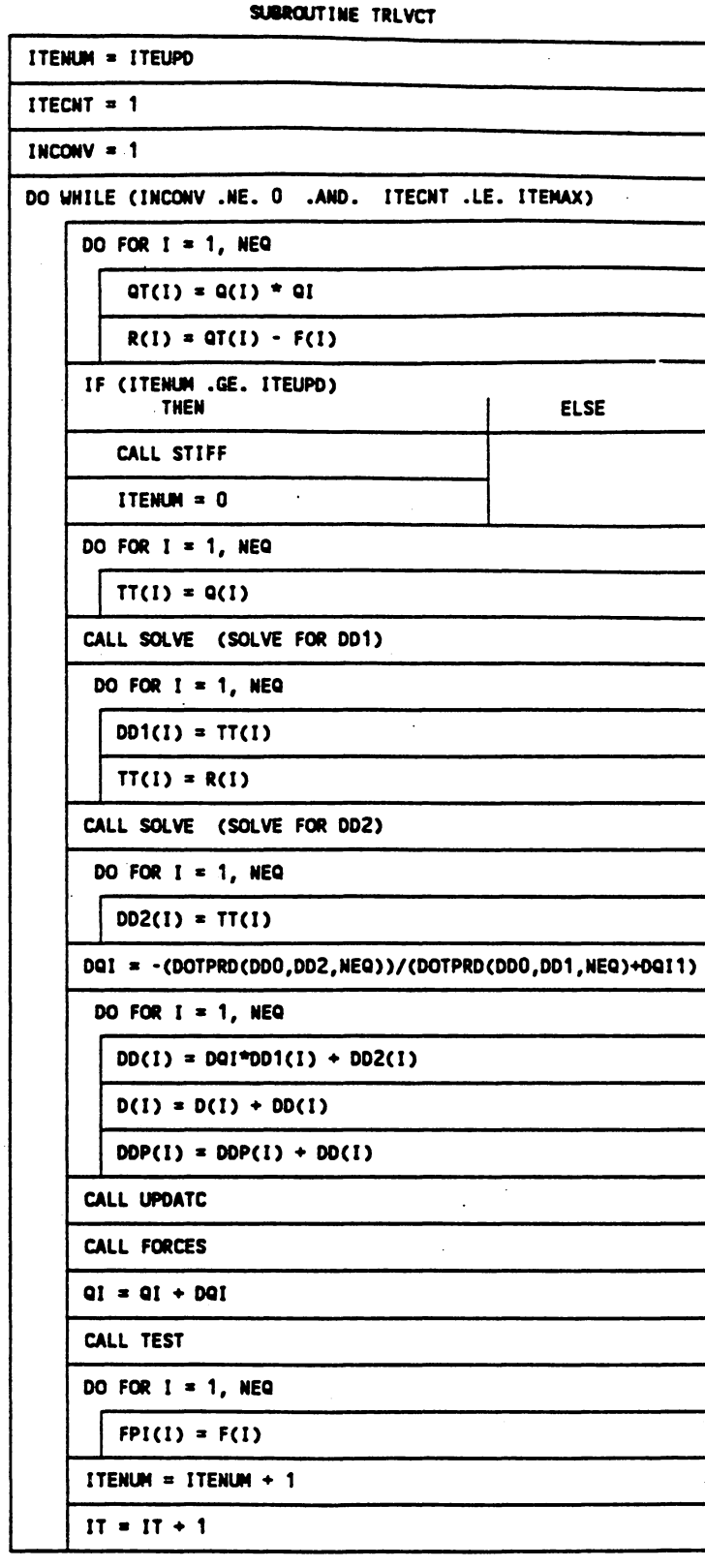

Figure 4.13 Subroutine TRLVCT

SUBROUTINE STIFF

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

Figure 4.14 Subroutine STIFF

SUBROUTINE ELEMS

```
GAMMA = AREA(I)*EMOD(I)/ELENG(I)
G(1) = GAMMA*C1(I)**2
G(2) = GAMMA*C2(I)**2
G(3) = GAMMA*C3(I)**2
G(4) = GAMMA*CI(I)*C2(I)
G(5) = GAMMA*C2(I)*C3(I)
G(6) = GAMMA*C1(I)*C3(I)
```

Figure 4.15 Subroutine ELEMS

SUBROUTINE NELEMS

```
GAMMA = AREA(I)*EMOD(I)/ELENG(I)
H(1) = GAMMA*(ELONG(I)/DEFLEN(I))*(1-C1(I)**2)
H(2) = GAMMA*(ELONG(I)/DEFLEN(I))*(1-C2(I)**2)
H(3) = GAMMA*(ELONG(I)/DEFLEN(I))*(1-C3(I)**2)
H(4) = -GAMMA*(ELONG(I)/DEFLEN(I))*C1(I)*C2(I)
H(5) = -GAMMA*(ELONG(I)/DEFLEN(I))*C2(I)*C3(I)
H(6) = -GAMMA*(ELONG(I)/DEFLEN(I))*C1(I)*C3(I)
```

Figure 4.16 Subroutine NELEMS

SUBROUTINE FORCES

```
DO FOR I = 1, NEQ
    F(I) = 0.DO
DO FOR I = 1, NE
    CALL INTERF
DO FOR I = 1, NEQ
    R(I) = QT(I) - F(I)
```

Figure 4.17 Subroutine FORCES

SUBROUTINE INTERF

```
GAMMA = AREA(I)*EMOD(I)/ELENG(I)
DO FOR L = 1, 6
    K = MCODE(L,J)
    IF (K .NE. 0)
            THEN                                   | ELSE
    IF L =
    1 | F(K) = -GAMMA*ELONG(I)*C1(I) + F(K)
    2 | F(K) = -GAMMA*ELONG(I)*C2(I) + F(K)
    3 | F(K) = -GAMMA*ELONG(I)*C3(I) + F(K)
    4 | F(K) = GAMMA*ELONG(I)*C1(I) + F(K)
    5 | F(K) = GAMMA*ELONG(I)*C2(I) + F(K)
    6 | F(K) = GAMMA*ELONG(I)*C3(I) + F(K)
```

Figure 4.18 Subroutine INTERF

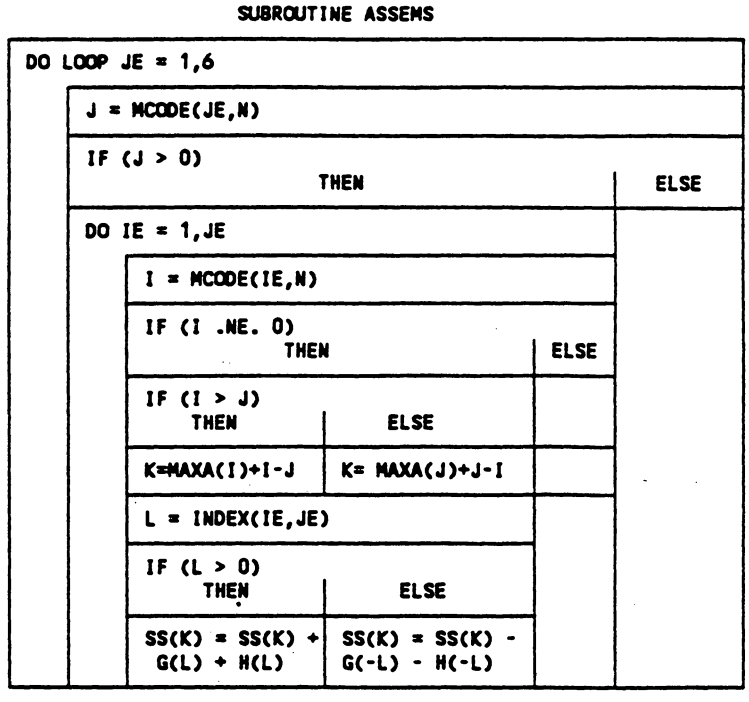

Figure 4.19 Subroutine ASSEMS

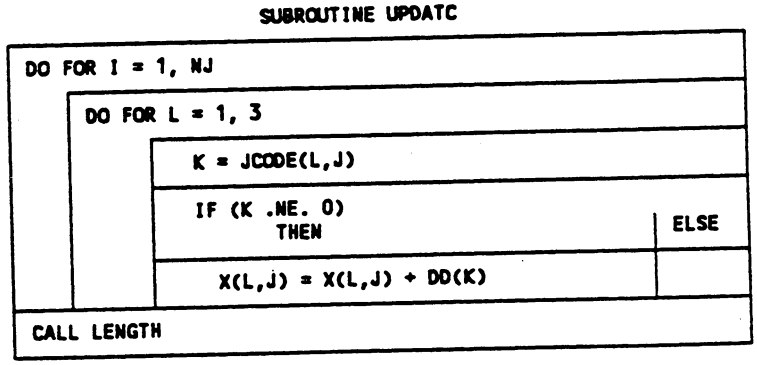

Figure 4.20 Subroutine UPDATC

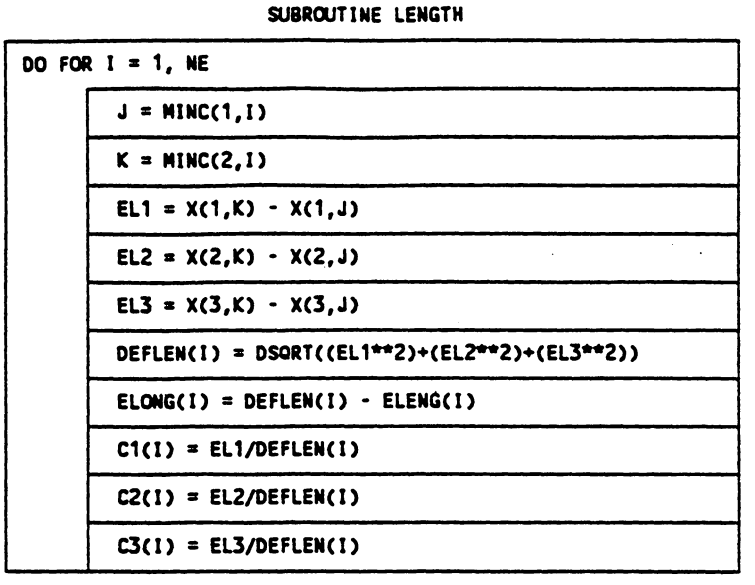

Figure 4.21 Subroutine LENGTH

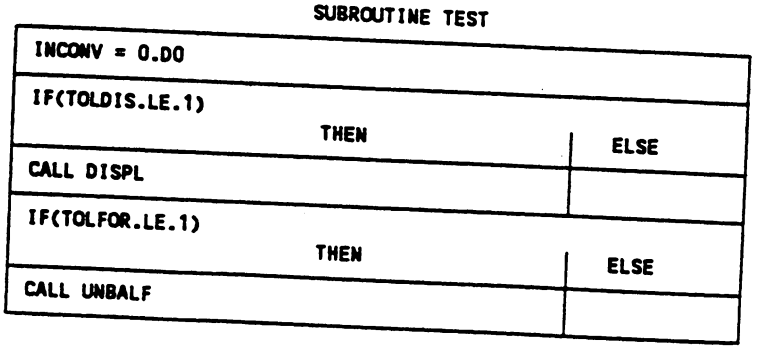

Figure 4.22 Subroutine TEST

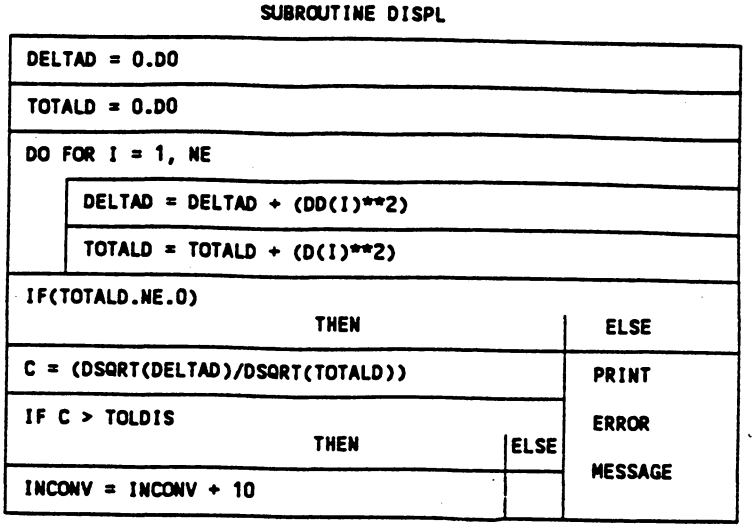

Figure 4.23 Subroutine DISPL

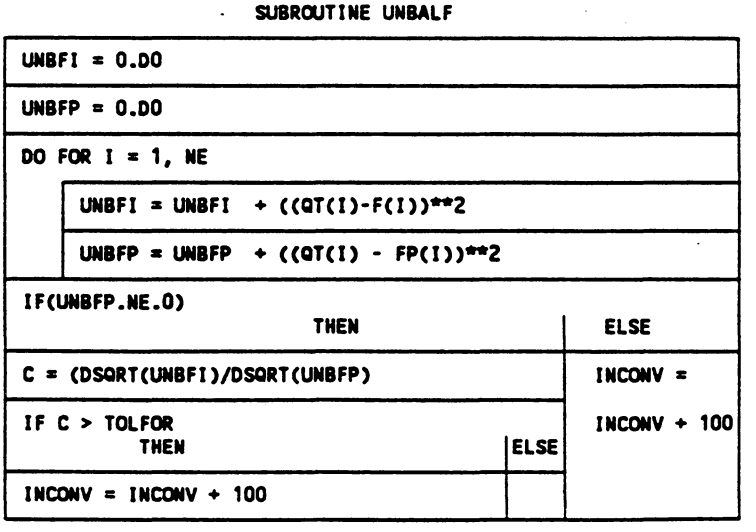

Figure 4.24 Subroutine UNBALF

CHAPTER 5

Discussion of Results

Four different structures have been examined in this study by the modified Riks/Wempner method and a homotopy method. The four structures were also analyzed with ABAQUS(Hibbit et al., 1986) using the Riks/Wempner method to test the accuracy of the results obtained by the programs. In Riks/Wempner program error tolerances were set at 10^{-4} for both the unbalanced force and the displacement. In the homotopy program the ARCRE and ARCAE relative and absolute error tolerances that allow the normal flow iteration along the zero curve are also set at 10^{-4}. In ABAQUS a force tolerance(PTOL) of 10^{-4} is used. The displacement tolerance has been suppressed in program 1 to make a meaningful comparison.

5.1 Structure 1

The first structure analyzed in this study is a two-bar plane truss. This structure is shown in Fig 5.1. This truss consists of two linearly elastic truss members and three joints. Joints 1 and 3 are fixed. Joint 2 is constrained in the direction of global 3-axis. The

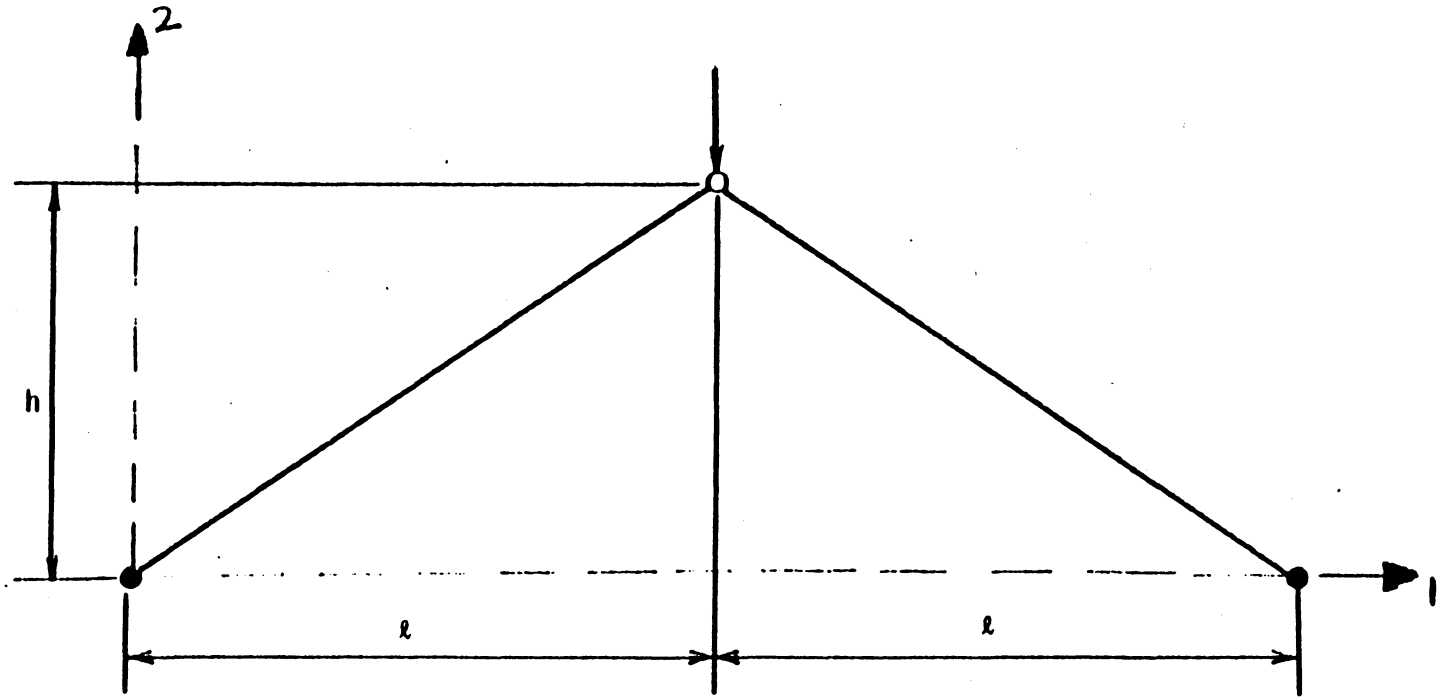

Figure 5.1 Structure 1

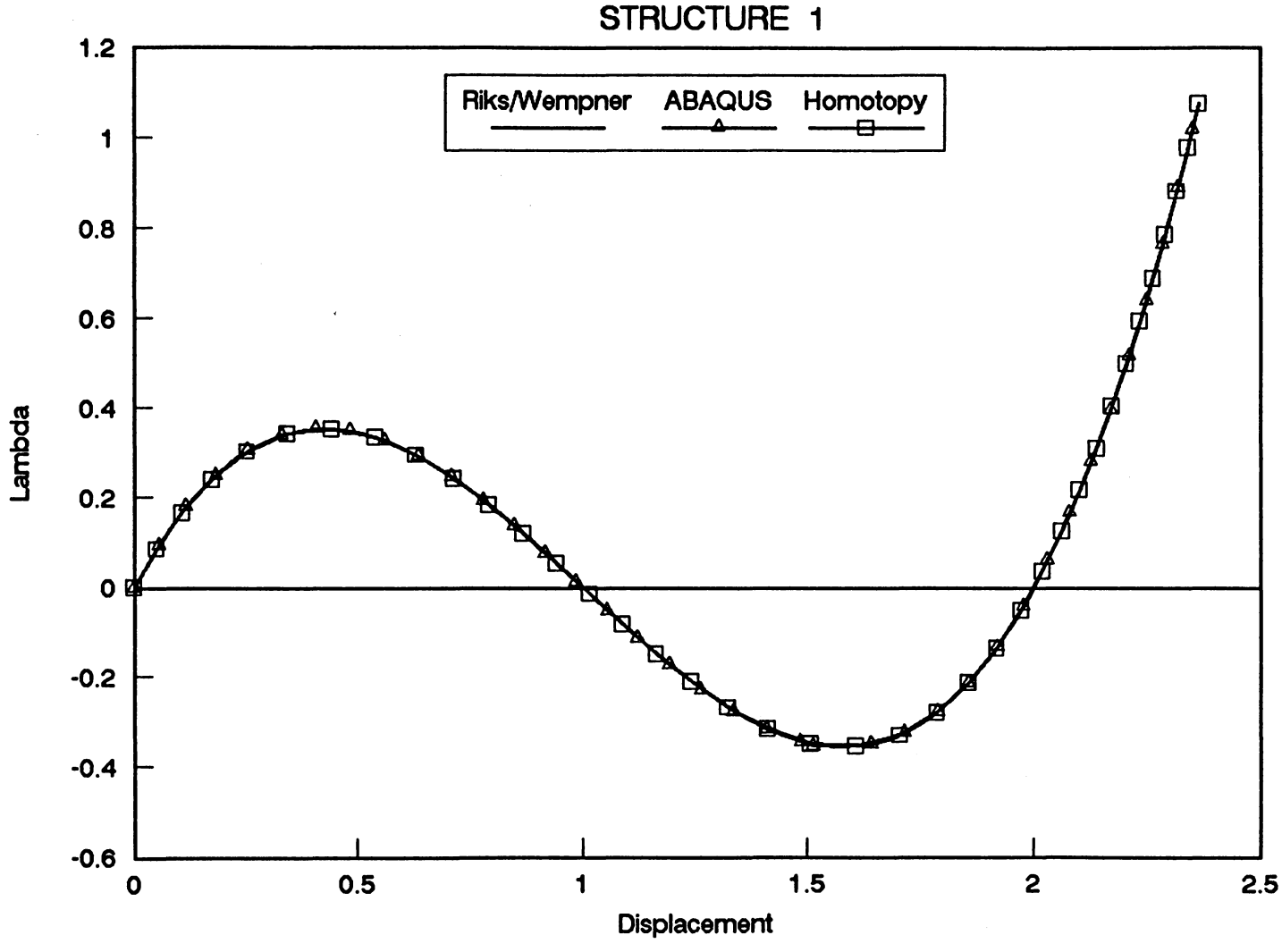

Figure 5.2 Vertical Displacement vs. Lambda for node 2

type of loading used on this structure is a single point load applied at the joint 2 in the global 2 direction. Cross-sectional area of each member is 0.18 square inches, and elastic modulus is 29000 ksi. A vertical load of 2 kips is applied at node 2.

The vertical downward displacement at the node 2 is plotted against the load proportionality factor(lambda) for both methods (Fig. 5.2). Both the programs successfully traced the post buckling path. The equilibrium path perfectly coincided with that obtained by ABAQUS.

5.2 Structure 2

The second structure analyzed in this study is a 21 degree-of-freedom lamella dome(Fig 5.3). This structure has 24 members and 13 joints. The joints lie on the surface of a spherical cap with a radius of 157.25 inches. The support ring of the cap has a radius of 50 inches. All joints are located so that the structure is symmetric about the global 1-axis which extends vertically through the apex. The six support joints are fixed in all three directions. Each member of the structure has a cross-sectional area of 0.18 square inches and a elastic modulus of 29000 ksi. Three different load conditions considered for this structure are: (i) Vertical load of 2 kips at the apex, (ii) symmetric vertical loads of 1 kip each on nodes 2 through 7, and (iii) un-symmetric vertical loads on nodes 2 through 7.

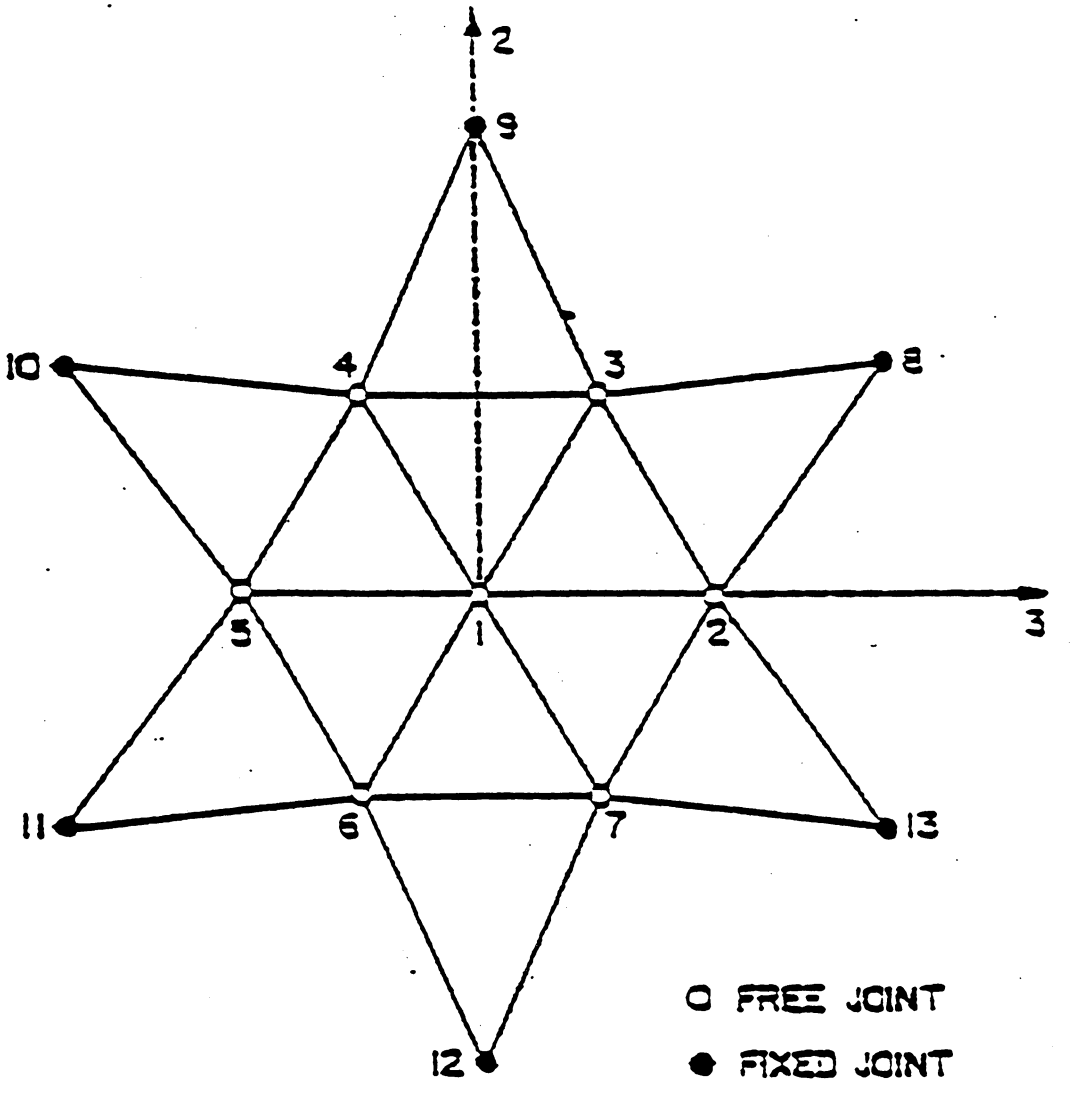

Figure 5.3 Structure 2

For the first load condition the response of the structure is plotted in Fig 5.4. Both programs successfully traced past the limit points. For the second load condition, a bifurcation point occurred at a load level of 8.55(Fig. 5.5). Both the programs successfully traced past this bifurcation point. For the third load condition a limit point occurred at a load level of 7.56(Fig. 5.6). In this case also both the programs trace past the limit point successfully. These responses are also in conformity with the results obtained from ABAQUS.

5.3 Structure 3

The third structure analyzed in this study is another lamella dome(Fig 5.7). This structure has 31 joints and 70 members. The joints are on a spherical cap. The radius of the sphere is 100 feet and the cap extends over a sector of 40^0 of the sphere. The structure is symmetric about global 1 axis and and extends vertically downward through joint 1. Joint 2 through 11 are positioned so that members extending radially outward are of the same length.

This dome was analyzed with a tension ring at the base. The tension ring members are of steel having an area of 1.0 square inches and an elastic modulus of 29000 ksi. All the other members are of wood and have a cross-sectional area of 6.0 square inches and an elastic modulus of 1800 ksi. The following constraints were imposed on the

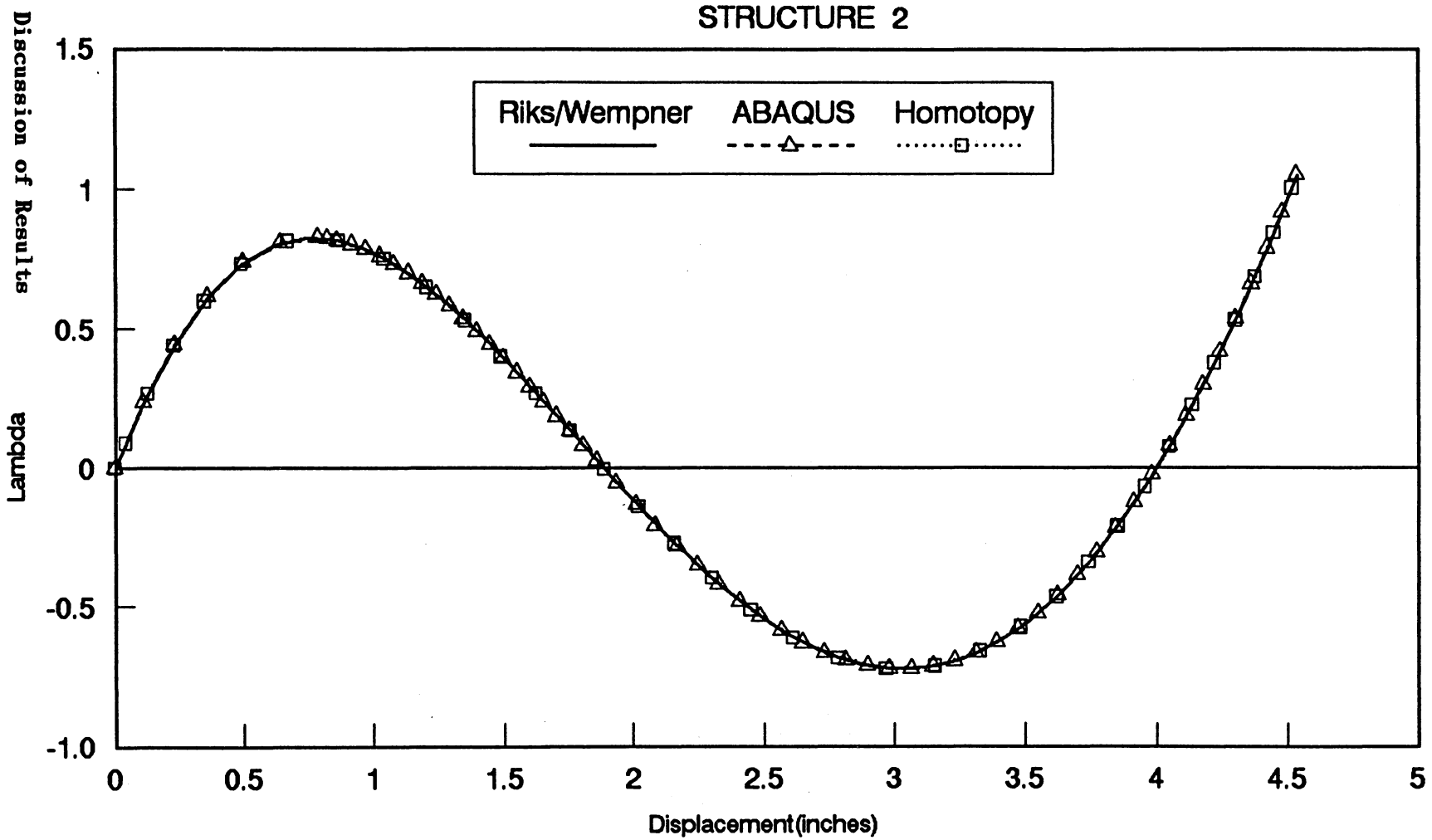

Figure 5.4 Vertical Displacement vs. Lambda for Apex

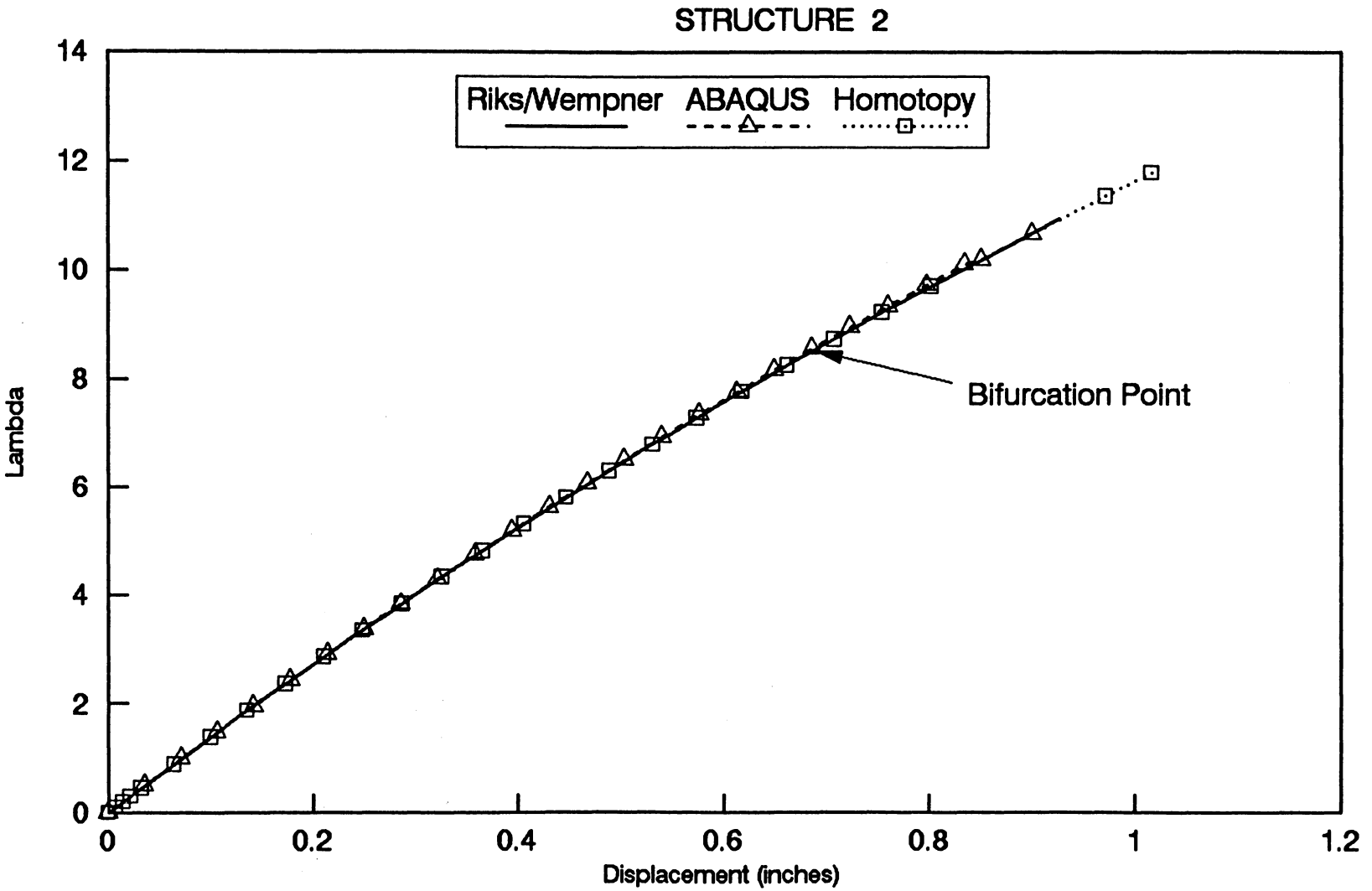

Figure 5.5 Lambda vs. Vertical Displacement at node 2 (Load Condition 2)

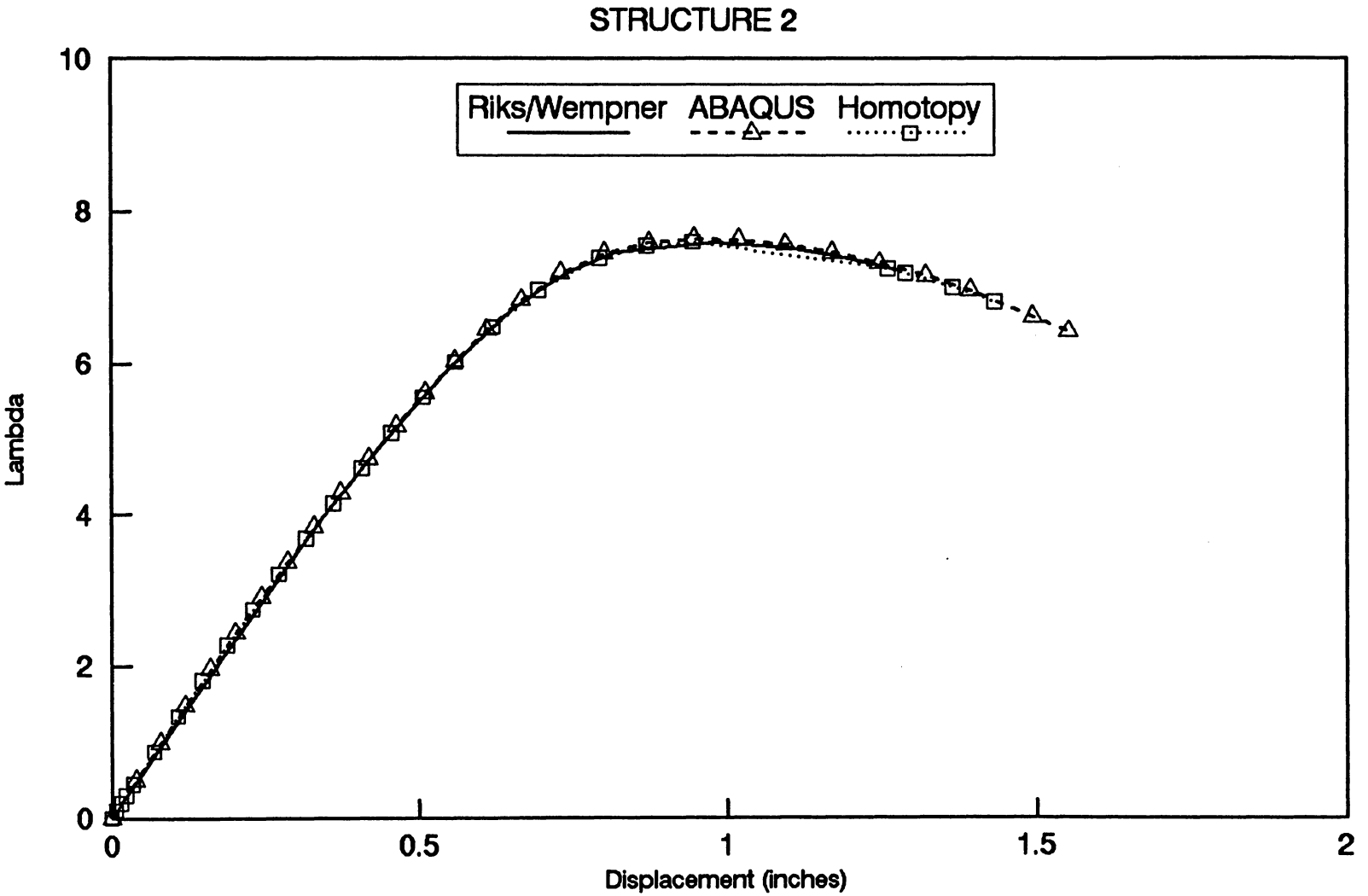

Figure 5.6 Lambda vs. Vertical Displacement at node 2 (Load Condition 3)

structure. Joint 12 to 31 are constrained in the direction of global 1 axis. Joints 17 and 27 are constrained in the direction of global 3-axis and joint 1 is constrained in the direction of global 2 and 3 axes to prevent rigid body motions. Two load conditions examined for this dome are: (i) symmetric vertical loads of 1 kip each at nodes 2 through 11, and (ii) un-symmetric vertical loads on nodes 2 through 11.

In the first load condition, a bifurcation point occurred at a load level of 0.8(Fig. 5.8). The Riks/Wempner program traced past the bifurcation point while the homotopy program failed to converge to a solution after the bifurcation point. The algorithm failed to trace the curve any further. The failure to trace any further was attributed to the failure of equation solver which makes use of preconditioned conjugate gradient. This needs modification to make it work for all problems. In the second loading condition a limit point occurred at a load level of 0.47(Fig. 5.9). Again Riks/Wempner program successfully traced the curve through the limit point and homotopy program could not trace the curve after the limit point due to the failure of the conjugate gradient equation solver.

5.4 Structure 4

The fourth structure analyzed in this study is a large dome(Fig. 5.10)

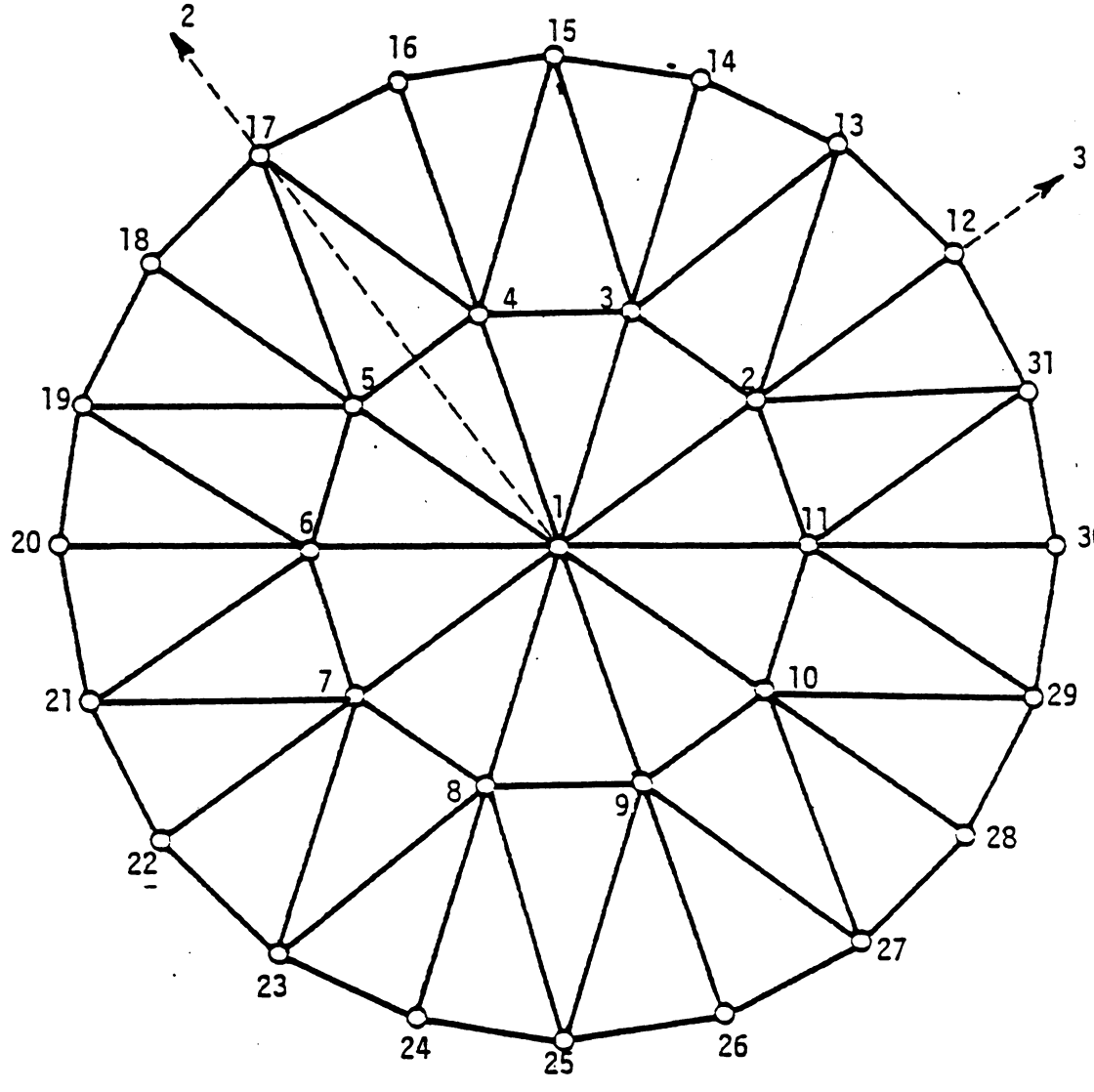

Figure 5.7 Structure 3

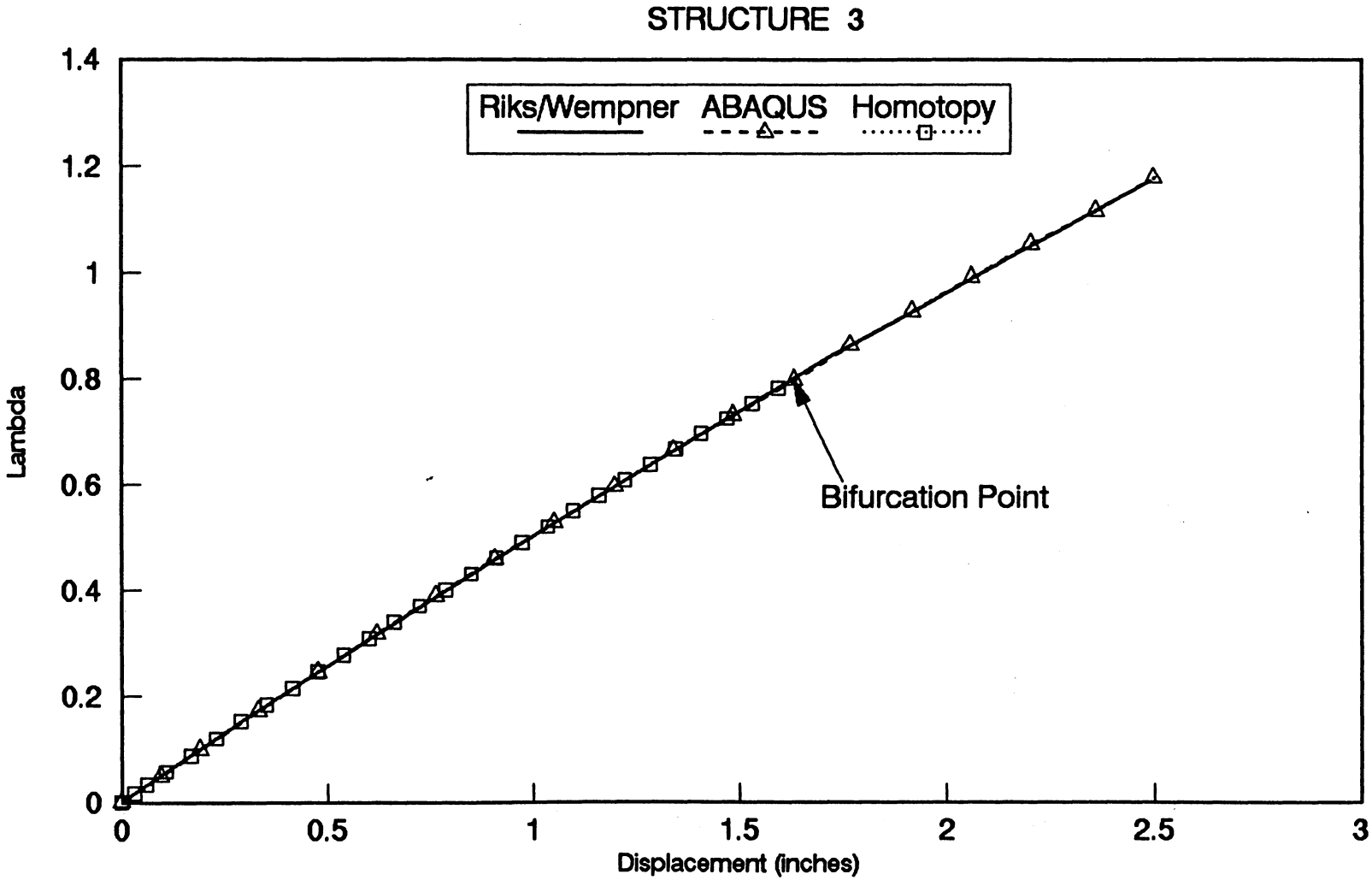

Figure 5.8 Lambda vs. Vertical Displacement at node 2 (Load Condition 1)

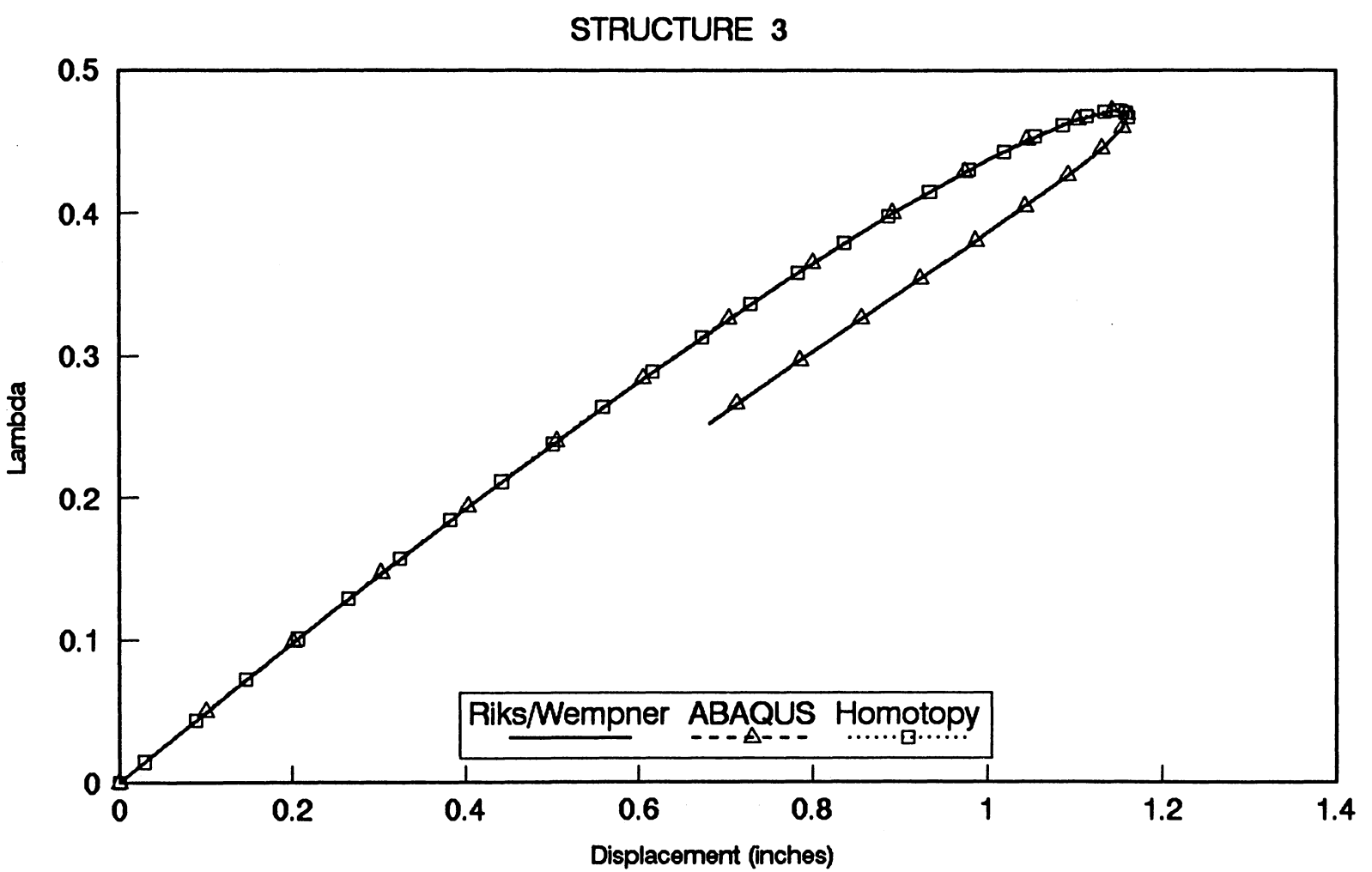

Figure 5.9 Lambda vs. Vertical Displacement at node 2 (Load Condition 2)

with 285 members and 106 joints. This dome was generated on I-DEAS (SDRC, 1986) from the data available for 1/10 th of the dome(Kissel, 1987).

All the members are considered to be made of aluminum. This dome has a tension ring. The tension ring elements have an area of 2.114 square inches and the other members have an area of 3.168 square inches. The elastic modulus of all the members is taken as 10,000 ksi. The nodes 102 through 106 are constrained in all the three directions. The remaining outer ring nodes are constrained in the vertical direction.

Two load conditions considered for this structure are: (i) symmetric vertical loads of 9 kips each at nodes 23, 35, 36, 47, and 48. (ii) un-symmetric vertical loads at nodes 23, 35, 36, 47, and 48.

The Riks/Wempner program successfully traced past the bifurcation and limit points for the first and second load conditions respectively. The homotopy program could not trace past the bifurcation and limit points.

5.5 Comparison of the two algorithms

The comparison of the two algorithms is based on the number of

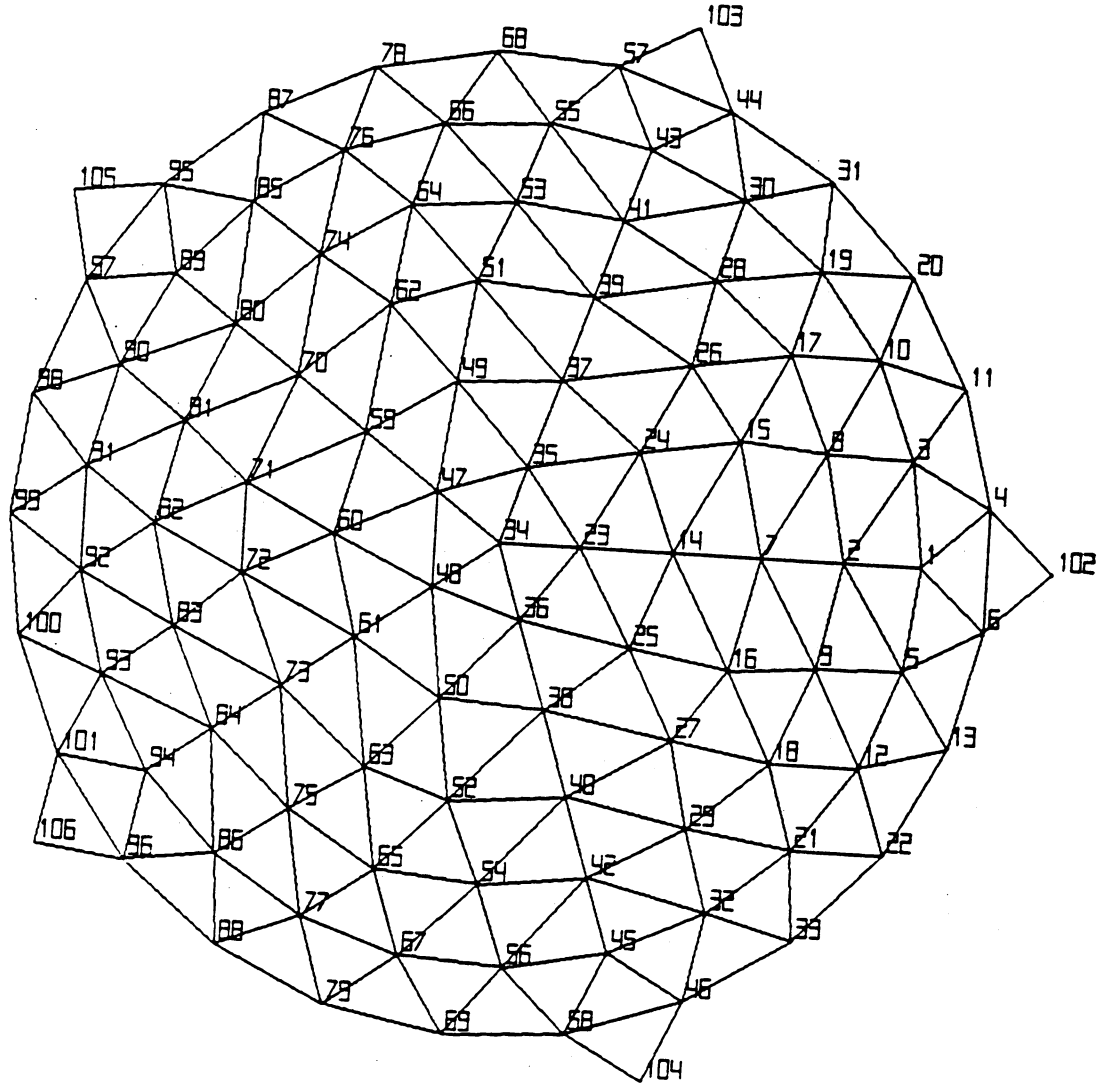

Figure 5.10 Structure 4

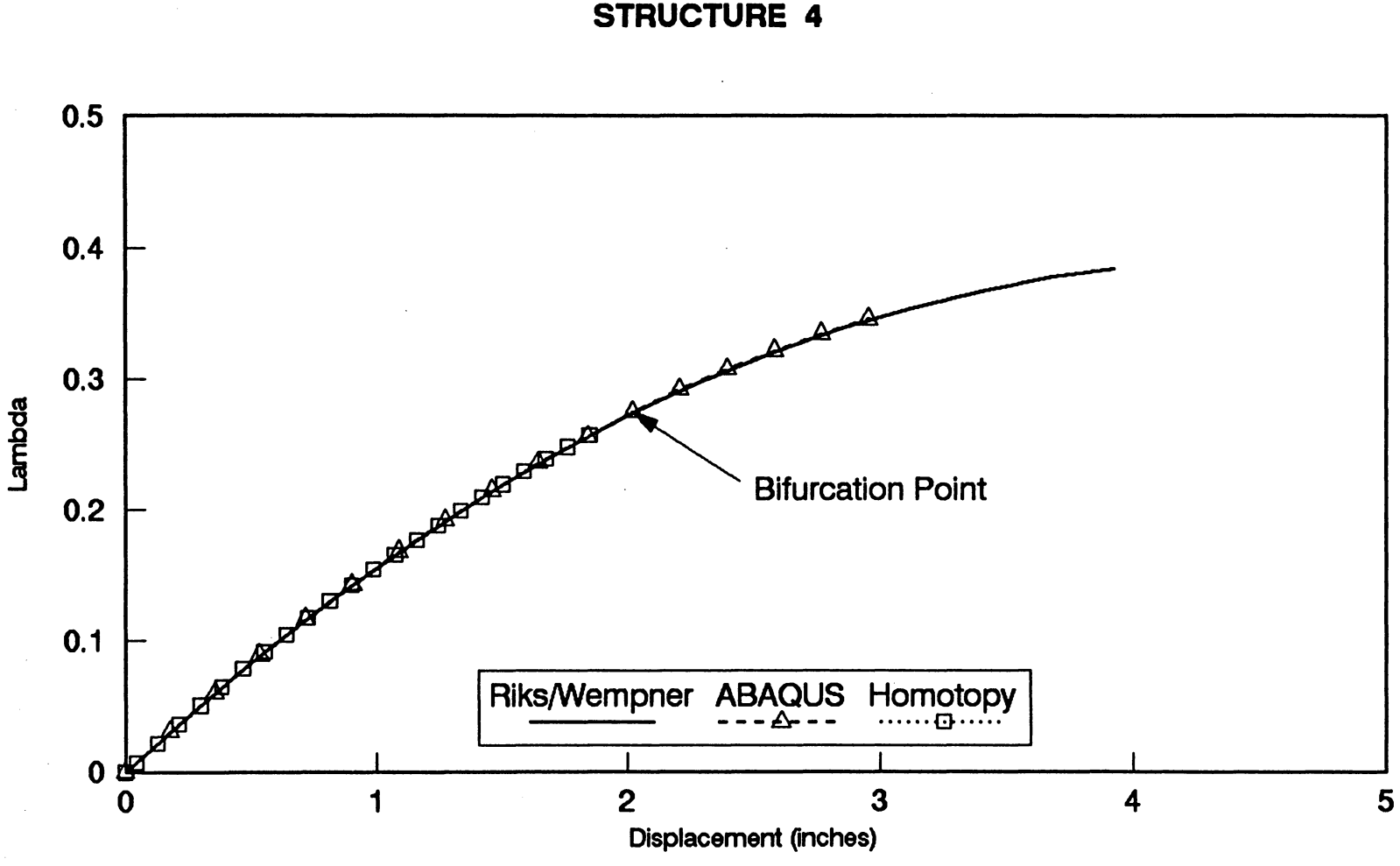

Figure 5.11 Lambda vs. Vertical Displacement at node 23 (Load Condition 1)

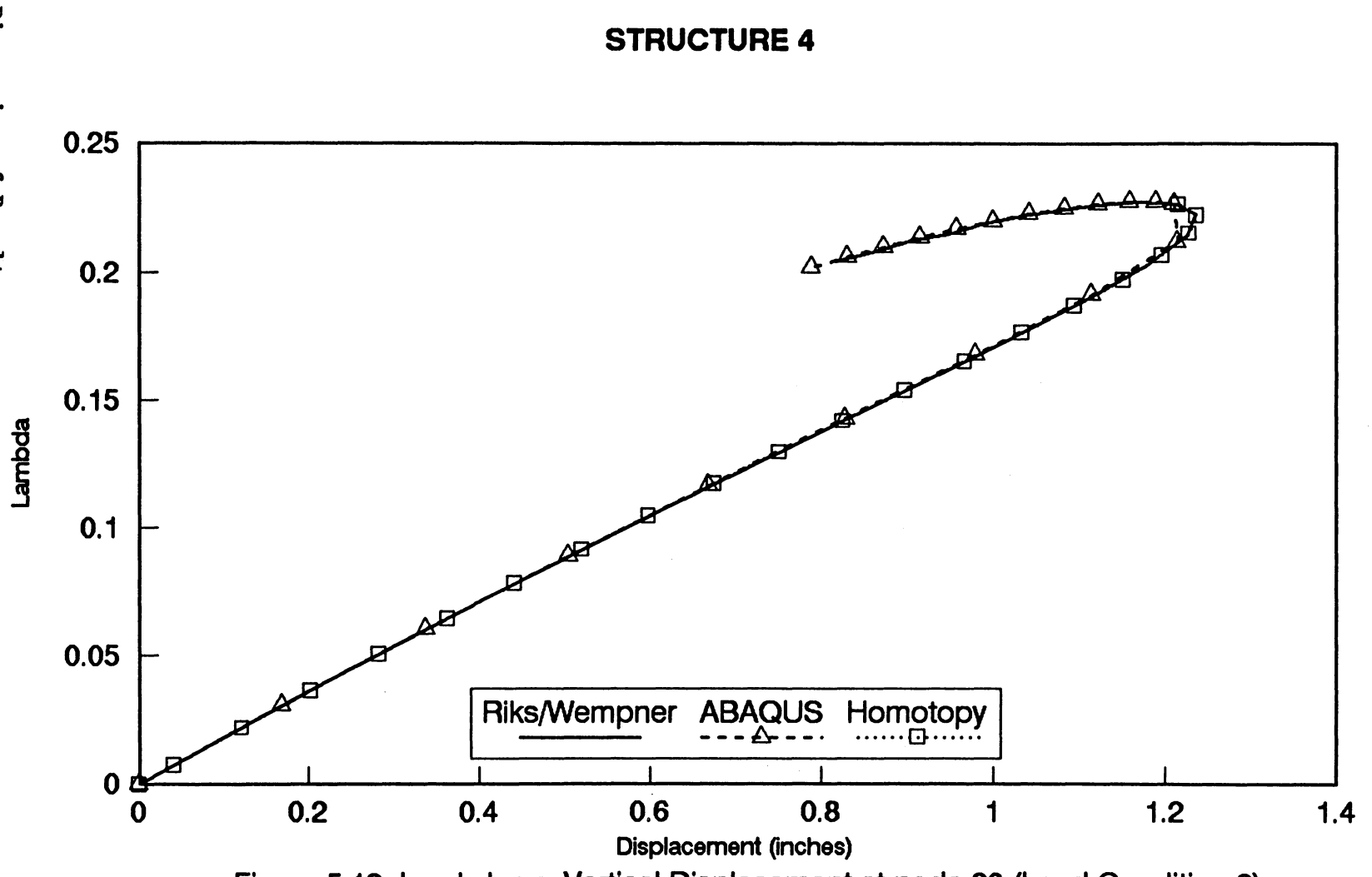

Figure 5.12 Lambda vs. Vertical Displacement at node 23 (Load Condition 2)

Jacobian matrix evaluations(tangent stiffness matrix evaluations in case of modified Riks/Wempner method) and the CPU time. This comparison has been made for Structure 2 (see Table 5.1).

For the first load condition the homotopy program required 48 Jacobian matrix evaluations while the Riks/Wempner program required 154 Jacobian matrix evaluations. In the second and third load conditions the difference in the number of Jacobian matrix evaluations is much less. But it is also to be noted that the two methods use two different stepping algorithms. In the Riks/Wempner program only initial load increment is prescribed by the user, and the algorithm follows an automatic stepping procedure based on arc length. In the homotopy program only the maximum and minimum load increments allowed are prescribed.

The Riks/Wempner program required about 1/4 of the CPU time required by the homotopy program for the first load condition. For the second load conditions the CPU time required by the homotopy method was as high as 12 times that of the Riks/Wempner program. And in the third load condition the homotopy program required 6 times the CPU time required by the modified Riks/Wempner method. In general, it is observed that modified Riks/Wempner method is much more economical in solving engineering problems than the sparse normal flow algorithm of the HOMPACK.

Table 5.1

Comparison of Riks/Wempner and Homotopy programs for Structure 2

Load Condition	Number of Jacobian matrix* Evaluations		CPU Time (in microseconds)	
	Riks/Wempner	Homotopy	Riks/Wempner	Homotopy
1	154	48	1,228,865	5,075,147
2	53	46	404,370	4,975,002
3	79	51	597,237	3,948,393

* Tangent stiffness matrix evaluations for modified Riks/Wempner method

The fact that the homotopy program requires lesser number of iterations and more CPU time may be explained as follows. The algorithm follows a normal flow iterative procedure to trace the next equilibrium configuration from the known equilibrium configuration. The tracking of equilibrium point by this procedure requires more computation.

The homotopy program was found to be less sensitive to the error tolerances as compared to the Riks/Wempner program. The accuracy of the displacements computed by the homotopy program was barely affected by the changes in error tolerances.

From the results obtained for the above four structures, it is understood that the sparse normal flow code of the HOMPACK needs to be modified to use an equation solver which will work for all kinds of problems.

CHAPTER 6

Conclusions and Recommendations

In this study, the modified Riks/Wempner method and a homotopy method were evaluated for their ability to track the equilibrium path of geometrically nonlinear space trusses. The most important conclusion of this study is that the modified Riks/Wempner method is capable of tracing the equilibrium path of all types of structures. The sparse normal flow algorithm of the HOMPACK method is capable of tracking the equilibrium path, but the HOMPACK code needs to be modified to make it useful in analyzing different types of structures.

The comparison of the two solution algorithms was carried out by (i) comparing the number of tangent stiffness matrix evaluations for the modified Riks/Wempner method and the number of Jacobian matrix evaluations for the sparse normal flow homotopy algorithm, (ii) the CPU time used by the the algorithm. During this study some recommendations for modification and improvement of the program were noted.

1. Modify the HOMPACK code to make it capable of tracking equilibrium paths of different types of structures. Specifically, the equation

solver being used by the HOMPACK needs to be modified or replaced.

2. Modify the Riks/Wempner program to compute the eigenvalues. This feature will enable the user to know if a bifurcation point has been passed.

References

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

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

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

4. Borst, R. de, "Computation of Post-bifurcation and Post-failure Behavior of Strain Softening Solids, Computers and Structures, Vol. 25, 1987, pp.211-224.

5. Clarke, M. J. and Hancock, G. J., " A Study of Incremental/Iterative Strategies for Nonlinear Analyses," International Journal for Numerical Methods in Engineering, Vol. 29, 1990, pp. 1365-1391.

6. Cook, R. D., Malkus, D.S., and Plesha, M. E., "Concepts and Applications of Finite Element Analysis," John Wiley & Sons, 1989.

7. Crisfield, M. A., "A Fast Incremental/Iterative Solution that Handles Snap-through," Computers and Structures, Vol. 13, 1981, pp. 55-62.

8. Forde, B. W. R., and Steimer, F. S., "Improved Arc Length

Orthogonality Methods for Nonlinear Finite Element Analysis," Computers and Structures, Vol. 27, no. 5, 1987, pp.625-630.

9. Fried, I., "Orthogonal Trajectory accession to Nonlinear Equilibrium Curves," Computer Methods in Applied Mechanics and Engineering, Vol. 47, 1984, pp. 283-297.

10. Hansen, M. C., "A comparison of Two Solution Techniques for Snap-through Instability of Trusses," M.S. Thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, Blacksburg, March 1981.

11. Hibbit, Karlsson and Sorensen Inc., ABAQUS, Version 4.7, Theory Manual, 1987

12. Holzer, S. M., Watson, L. T., and Vu, P. D., "Stability Analysis of Lamella Domes," Proceedings of the ASCE 1981 Convention and Exposition, Committee on Special Structures, Structural Division, ASCE, October 1981, pp. 179-209.

13. Holzer, S. M., Chapter 7, "Computer Analysis of Structures," Elsevier Science Publishing, 1985.

14. Holzer, S. M., "Class notes on Dynamics of Structures," Virginia Polytechnic Institute and State University, Blacksburg, Virginia, Fall Semester, 1990.

15. Holzer, S. M., "Class notes on computer analysis of Structures," Virginia Polytechnic Institute and State University, Blacksburg, Virginia, Spring Semester, 1990.

16. Kissel, J. R., Conservatek, Conservatek Place, Conroe, Texas, 1987, Private Communication.

17. Kouhia, R. and Mikkola, M.,"Tracing Equilibrium Paths Beyond Critical points," International Journal for Numerical Methods in Engineering, Vol. 28, 1989, pp. 2923-2943.

18. Ramm, E., "Strategies for Tracing Nonlinear Response near limit Points," Europe-U.S. Workshop: Nonlinear Finite Element Analysis in Structural Mechanics, July 28-31, 1980, Bochum, G. F. R.

19. Riks, E., " An Incremental Approach to the Solution of Snapping and Buckling Problems," International Journal of Solids and Structures, Vol. 15, 1979, pp. 529-551.

20. Sage, W. M., "Nonlinear Analysis of Cable Structures," M.S. Thesis, Virginia Polytechnic Institute and State University, Blacksburg , Virginia, December 1986.

21. Structural Dynamics Research Corporation, Integrated Design Engineering Analysis Software, MacNeal Schwendler, U. S. A., 1986.

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

23. Watson, L. T., Holzer, S. M., and Hansen, M. C., "Tracking Nonlinear Equilibrium Paths by a Homotopy Method," Nonlinear Analysis, Theory, methods and Applications, Vol. 7, No. 11, pp. 1271-1282, 1983.

24. Watson, L. T., Billups, S. C., and Morgan, A.P., "A Suite of Codes for Globally Convergent Algorithms," Research Publication, GMR-5344 Rev, General Motors Research Laboratories, June 1986

25. Watson, L. T., "Globally Convergent Homotopy Methods: A Tutorial,"

Applied Mathematics and Computation, Vol. 31BK, 1989, pp. 369-396.

26. Wempner, G. A., " Discrete Approximation to Related Nonlinear Theories of Solids", International Journal of solids and Structures, Vol. 7, 1971, pp. 1581-1599

27. Wu, J. Y., "A Study of Moving Contact Algorithm," M.S. Thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, February 1987.

Appendix B

LISTING OF PROGRAM 1

```
C
C   THIS PROGRAM TRACES THE EQUILIBRIUM PATH OF THE GEOMETRICALLY
C   NONLINEAR SPACE TRUSSES BY THE NEWTON-RAPHSON METHOD AND MODIFIED
C   RIKS/WEMPNER METHOD.
C
C   WRITTEN BY BALAJI SURESH SUNKU
C
C*************************************************************************
C*                              INPUT DATA                               *
C*************************************************************************
C     LIST-DIRECTED INPUT:
C     INPUT UNITS: KIP, INCH, RADIAN, FAHRENHEIT
C
C     1. ENTER DATE IN THE FORM  01/14/91  (IN MAIN)
C           DATE
C
C     2. ENTER THE METHOD TO BE USED (IN MAIN)
C        NEWTON-RAPHSON: 1    MODIFIED RIKS/WEMPNER: 2
C           METHOD
C
C     3. ENTER NUMBER OF ELEMENTS AND NUMBER OF JOINTS (IN MAIN)
C           NE, NJ
C
C     4. ENTER MEMBER INCIDENCES (IN STRUCT)
C           MINC(1,I), MINC(2,I)  I=1,NE
C
C     5. ENTER FOR EACH JOINT CONSTRAINT (IN STRUCT)
C           JNUM, JDIR
C           AFTER LAST JOINT CONSTRAINT ENTER
C           0, 0
C
C     6. ENTER JOINT COORDINATES FOR EACH JOINT(IN PROP)
C           X(1,J), X(2,J), X(3,J)   J=1,NJ
C
C     7. ENTER MEMBER PROPERTIES CROSS SECTIONAL AREA
C           AND ELASTIC MODULUS FOR EACH MEMBER (IN PROP)
C           AREA(I), EMOD(I)    I=1,NE
C
C     8. IF THERE ARE JOINT LOADS ENTER (IN JLOAD)
C              JNUM, JDIR, FORCE
C              AFTER LAST JOINT LOAD ENTER
C              0, 0, 0
C           ELSE ENTER
C              0, 0, 0
C           END IF
C
C     9. ENTER THE MAXIMUM VALUE OF LOAD PROPORTIONALITY FACTOR(LAMBDA),
C        INITIAL VALUE OF LAMBDA AND THE INCREMENT IN LAMBDA   (IN MAIN)
C           QIMAX, QI, DQI
C
C    10. ENTER THE NUMBER OF ITERATIONS FOR UPDATING STIFFNESS MATRIX,
C        MAXIMUM AND DESIRED NUMBER OF ITERATIONS (IN MAIN)
C           ITEUPD, ITEMAX, ITEDES
C
C    11. ENTER THE TOLERANCES IN DISPLACEMENT, AND FORCE (IN MAIN)
C           TOLDIS, TOLFOR
C
C    12. ENTER THE DOF FOR WHICH DISPLACEMENT AND LAMBDA VALUES ARE TO
C        BE PRINTED AFTER EACH LOAD INCREMENT (IN RESULT)
C        END WITH 0
C
```

```
C*********************************************************************
C*                          MAIN PROGRAM                             *
C*********************************************************************
C
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DOUBLE PRECISION TIMEA, TIMEB, DTIME
      CHARACTER*(*) TITLE, UNITS, DATE*8, FNAME*12
      PARAMETER (LIM = 100000, TITLE = 'SPACE TRUSS ANALYSIS',
     $           UNITS = 'UNITS: KIP, INCH, RADIAN, FAHRENHEIT')
      DIMENSION A(LIM),IA(LIM)
C
C     RESERVE MEMORY; READ AND ECHO NE, NJ; SET POINTERS FOR DATA
C     ARRAYS; IF MEMORY IS ADEQUATE CALL STRUCT, ELSE SEND MESSAGE AND
C     STOP; SET POINTERS FOR SOLUTION ARRAYS; IF MEMORY IS ADEQUATE,
C     CALL LOAD. THEN CALL NEWRAP OR RIKWEM
C-------------------------------------------------
C     OPEN DATA FILES: ********* FOR PC ONLY*******
C     WRITE(6,*) 'INPUT DATA FILE: '
C     READ(*,'(A)') FNAME
C     OPEN(5,FILE = FNAME)
C     OPEN(7,FILE = 'RW.OUT', STATUS = 'UNKNOWN')
C-------------------------------------------------
C
C
      READ(5,'(A)') DATE
      WRITE(6,5) TITLE,'(SUNKU, 1991)',
     $'DATE: ',DATE,UNITS
 5    FORMAT('1',T5,73('*')/T5,'*',T32,A,T77,'*'/T5,'*',T35,A,T77,'*'/
     $          T5,73('*')//T64,2(A)///T5,A)
      READ(5,*)METHOD
      READ(5,*) NE, NJ
      WRITE(6,15) 'CONTROL VARIABLES',
     $            'NUMBER OF ELEMENTS',NE,'NUMBER OF JOINTS',NJ
 15   FORMAT(///T5,A/2(T5,A,T30,I4/))
C
C---- SET POINTERS FOR DATA ARRAYS (IN STRUCT):
C
      NP=1
      NAREA=NP+3*NJ
      NEMOD=NAREA+NE
      NELENG=NEMOD+NE
      NC1=NELENG+NE
      NC2=NC1+NE
      NC3=NC2+NE
      NMCODE=NC3+NE
      NJCODE=NMCODE+6*NE
      NMINC=NJCODE+3*NJ
      NMAXA=NMINC+2*NE
C     TEMPORARILY LET NEQ=3*NJ TO SET POINTERS FOR NKHT AND MAX
      NEQ=3*NJ
      NKHT=NMAXA+(NEQ+1)
      MAX=NKHT+NEQ-1
C
C---- IF MEMORY IS ADEQUATE CALL STRUCT, ELSE SEND MESSAGE AND STOP.
C
      IF(MAX .LE. LIM) THEN
         CALL STRUCT(A(NP),A(NAREA),A(NEMOD),A(NELENG),
     $               A(NC1),A(NC2),A(NC3),IA(NMAXA),IA(NKHT),
     $               IA(NMCODE),IA(NJCODE),IA(NMINC),NE,NJ,NEQ,LSS)
C
C---- SET POINTERS FOR SOLUTION ARRAYS (IN LOAD AND NEWRAP OR RIKWEM)
C
C---- TEMPORARILY SET LSS=NEQ
C        LSS = NEQ
C
         NF=NKHT+NEQ
         NSS=NF+NEQ
         NQ=NSS+LSS
         NQT=NQ+NEQ
         NR=NQT+NEQ
```

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

```
     $                    A(NC1),A(NC2),A(NC3),A(NQ),A(NQT),
     $                    A(NSS),A(NTT),A(NP),A(NELONG),A(NDEFLN),
     $                    A(NF),A(NFP),A(NFPI),A(NR),
     $                    IA(NJCODE),IA(NMAXA),IA(NMCODE),IA(NMINC),
     $                    IT,INCONV,ITECNT,ITENUM,ITEMAX,ITEUPD,ITEDES,
     $                    NE,NEQ,NJ,LSS,QIMAX,QI,DQI,
     $                    TOLDIS,TOLFOR,LSTEP,NFE)
C
          ENDIF
C
        ELSE
          WRITE(6,'(T10,A/)') 'ERROR MESSAGE: INCREASE MEMORY'
        END IF
      ELSE
        WRITE(6,'(T10,A/)') 'ERROR MESSAGE: INCREASE MEMORY'
      ENDIF
C
C
C*************************************************************************
C---- INSERT CALL TO RETURN EXECUTION TIME IN MICROSECONDS IN DTIME.
C
       CALL CPUTIME(TIMEB, IRCD_B)
       IF(IRCD_A .NE. 8 .AND. IRCD_B .EQ. 0)THEN
            DTIME = (TIMEB - TIMEA)
       ENDIF
C
C*************************************************************************
C
        WRITE(7,*) ' CPUTIME (MICROSECONDS)  = ', DTIME
        WRITE(7,*) '                   NFE   = ', NFE
C
      STOP
      END
C*************************************************************************
C*                              NEWRAP                                   *
C*************************************************************************
      SUBROUTINE NEWRAP(D,DD,Q,QT,TT,SS,AREA,EMOD,ELENG,C1,C2,C3,
     $                  ELONG,DEFLEN,F,FP,R,X,DDO,FPI,
     $                  MAXA,MCODE,JCODE,MINC,
     $                  INCONV,ITECNT,ITENUM,ITEUPD,ITEMAX,NE,NEQ,NJ,
     $                  LSS,QIMAX,QI,DQI,TOLDIS,TOLFOR,LSTEP,NFE)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION D(*),DD(*),Q(*),QT(*),TT(*),SS(*),AREA(*),EMOD(*),
     $          ELENG(*),C1(*),C2(*),C3(*),ELONG(*),DEFLEN(*),F(*),
     $          FP(*),R(*),X(3,*),DDO(*),FPI(*),
     $          MAXA(*),MCODE(6,*),JCODE(3,*),MINC(2,*)
C
C---- INITIALIZE THE VECTORS D, DD, F, FP, FPI TO ZERO
      DO 10 I = 1, NEQ
         D(I) = 0.D0
         DD(I) = 0.D0
         F(I) = 0.D0
         FP(I) = 0.D0
         FPI(I) = 0.D0
 10   CONTINUE
C
C---- INITIALIZE DEFLEN(DEFORMED LENGTH) AND ELONG(ELONGATION) TO ZERO
      DO 15 I = 1, NE
        DEFLEN(I) = ELENG(I)
        ELONG(I)  = 0.D0
 15   CONTINUE
C
C---- START THE NEWTON-RAPHSON PROCEDURE AND CONTINUE UP TO GIVEN MAXIMUM
C      VALUE OF LOAD PROPORTIONALITY FACTOR(LAMBDA)
C
 30   IF (QI .LE. QIMAX) THEN
C
C---- COMPUTE THE LOAD VECTOR USING THE CURRENT LAMBDA
         DO 40 I = 1, NEQ
            QT(I) = Q(I) * QI
```

```
 40      CONTINUE
C
C---- CALL THE SUBROUTINE NRITER TO PERFORM NEWTON-RAPHSON ITERATION
      CALL NRITER(D,DD,Q,QT,TT,SS,AREA,EMOD,ELENG,C1,C2,C3,
     $               ELONG,DEFLEN,F,FP,R,X,DDO,FPI,
     $               MAXA,MCODE,JCODE,MINC,
     $               INCONV,ITECNT,ITENUM,ITEUPD,ITEMAX,NE,NEQ,NJ,
     $               LSS,QIMAX,QI,DQI,TOLDIS,TOLFOR,NFE)
C
C---- ASSIGN F TO FP
         DO 60 I = 1, NEQ
              FP(I) = F(I)
 60      CONTINUE
C
C---- IF CONVERGENCE IS NOT ACHIEVED PRINT MESSAGE. OTHERWISE CALL RESULT.
         IF (INCONV .NE. 0) THEN
            WRITE(6,*)'*****ERROR *****SOLUTION FAILS TO CONVERGE IN',
     $                 ' GIVEN NUMBER OF ITERATIONS'
            STOP
         ELSE
              CALL RESULT(D,MCODE,JCODE,MINC,NE,NJ,NEQ,LSTEP,QI)
         ENDIF
C
C---- INCREMENT THE LOAD PARAMETER QI
         QI = QI + DQI
         GO TO 30
      ENDIF
C
      RETURN
      END
C*************************************************************************
C*                              NRITER                                   *
C*************************************************************************
      SUBROUTINE NRITER(D,DD,Q,QT,TT,SS,AREA,EMOD,ELENG,C1,C2,C3,
     $                  ELONG,DEFLEN,F,FP,R,X,DDO,FPI,
     $                  MAXA,MCODE,JCODE,MINC,
     $                  INCONV,ITECNT,ITENUM,ITEUPD,ITEMAX,NE,NEQ,NJ,
     $                  LSS,QIMAX,QI,DQI,TOLDIS,TOLFOR,NFE)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION D(*),DD(*),Q(*),QT(*),TT(*),SS(*),AREA(*),EMOD(*),
     $          ELENG(*),C1(*),C2(*),C3(*),ELONG(*),DEFLEN(*),F(*),
     $          FP(*),R(*),X(3,*),DDO(*),FPI(*),
     $          MAXA(*),MCODE(6,*),JCODE(3,*),MINC(2,*)
C
C
      CALL FORCES(F,QT,AREA,EMOD,ELENG,C1,C2,C3,ELONG,R,MCODE,MINC,
     $                  NE,NEQ)
C
C---- INITIALIZE THE COUNTERS ITENUM AND ITECNT AND THE CONVERGENCE FLAG
C     INCONV
C
         ITENUM = ITEUPD
         ITECNT = 1
         INCONV = 1
C
C
 10   IF (INCONV .NE. 0 .AND. ITECNT .LE. ITEMAX) THEN
C
C---- COMPUTE THE UNBALANCED FORCE VECTOR R
         DO 15 I = 1, NEQ
             R(I) = QT(I) - F(I)
 15      CONTINUE
C
C---- COMPUTE STIFFNES MATRIX
         IF (ITENUM .GE. ITEUPD) THEN
            CALL STIFF(SS,AREA,EMOD,ELENG,C1,C2,C3,ELONG,DEFLEN,
     $                 MAXA,MCODE,NE,LSS)
            ITENUM = 0
         ENDIF
```

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

```
        F(I) = 0.D0
        FP(I) = 0.D0
        FPI(I) = 0.D0
 10   CONTINUE
C
C---- INITIALIZE THE VARIABLES DEFLEN, ELONG
C
      DO 15 I = 1, NE
        DEFLEN(I) = ELENG(I)
        ELONG(I)  = 0.D0
 15   CONTINUE
C
      NFE = 0
      ITECNT = 1
C
C---- START THE RIKS/WEMPNER PROCRDURE AND PROCEED UNTIL THE GIVEN
C     MAXIMUM LAMBDA OR THE MAXIMUM NUMBER OF ITERATIONS IS EXCEEDED
C
 20   IF (QI .LE. QIMAX .AND. ITECNT .LE. 50)THEN
C
C---- COMPUTE THE TANGENT STIFFNESS MATRIX
C
         CALL STIFF(SS,AREA,EMOD,ELENG,C1,C2,C3,ELONG,DEFLEN,
     $              MAXA,MCODE,NE,LSS)
C
C---- COMPUTE THE FIRST TRIAL SOLUTION
C
         DO 40 I = 1, NEQ
             TT(I) = 0.D0
             TT(I) = Q(I)
 40      CONTINUE
C
         CALL SOLVE(SS,TT,MAXA,NEQ,1)
C
         DO 50 I = 1, NEQ
             DD01(I) = TT(I)
             TT(I) = 0.D0
 50      CONTINUE
C
C---- COMPUTE THE ARC LENGTH FOR THE FIRST ITERATION
C     FOR SUBSEQUENT ITERATIONS COMPUTE THE LOAD INCREMENT
C
         IF(ITECNT .EQ. 1)THEN
             DS=DQI*DSQRT(DOTPRD(DD01,DD01,NEQ)+1.D0)
             DSMAX=DS*1.000
         ELSE
             DQI=DS/DSQRT(DOTPRD(DD01,DD01,NEQ)+1.D0)
             TEMP=DQI*(DOTPRD(DD01,DDP,NEQ)+DQII)
             IF(TEMP .GT. 0)THEN
                SGN=1
             ELSE
                SGN=-1
             ENDIF
             DQI = SGN*DQI
         ENDIF
C
         DQII = 0.0
         DQI1 = DQI
         DQII = DQII + DQI
C
C---- COMPUTE THE INCREMENT IN DISPLACEMENT AND TOTAL DISPLACEMENT
C
         DO 60 I = 1, NEQ
            DDO(I) = DQI*DD01(I)
            D(I) = D(I) + DDO(I)
            DDP(I) = DDO(I)
 60      CONTINUE
C
C---- INCREMENT THE LOAD PARAMETER
C
```

```
            QI = QI + DQI
C---- UPDATE THE COORDINATES
          CALL UPDATC(X,DDO,C1,C2,C3,ELONG,ELENG,DEFLEN,MINC,JCODE,
     $               NE,NJ,NEQ)
C
          CALL FORCES(F,QT,AREA,EMOD,ELENG,C1,C2,C3,ELONG,R,MCODE,MINC,
     $               NE,NEQ)
C
C---- CALL SUBROUTINE TRLVCT
C
          CALL TRLVCT(D,DD,DDO,DD1,DD2,DDP,AREA,EMOD,ELENG,
     $           C1,C2,C3,Q,QT,SS,TT,X,ELONG,DEFLEN,F,FP,FPI,R,
     $           JCODE,MAXA,MCODE,MINC,
     $           IT,INCONV,ITECNT,ITENUM,ITEMAX,ITEUPD,NE,NEQ,
     $           NJ,LSS,QIMAX,QI,DQI,TOLDIS,TOLFOR,DQI1,DQII,NFE)
C
C
          DO 80 I = 1, NEQ
               FP(I) = F(I)
 80       CONTINUE
C
C---- IF CONVERGENCE IS NOT ACHIEVED PRINT MESSAGE. OTHERWISE CALL RESULT.
C
          IF(INCONV .NE. 0)THEN
             WRITE(6,*) '**** ERROR **** SOLUTION FAILS TO CONVERGE',
     $       ' IN GIVEN NUMBER OF ITERATIONS'
             STOP
          ELSE
             CALL RESULT(D,MCODE,JCODE,MINC,NE,NJ,NEQ,LSTEP,QI)
C
          ENDIF
C
              ITECNT = ITECNT + 1
C
C----COMPUTE THE SCALED ARC LENGTH
C
           DS=DS*DSQRT((1.D0*ITEDES)/(1.D0*IT))
           IF(DABS(DS) .GT. DSMAX)THEN
                IF(DS .LT. 0)THEN
                   DS = -DSMAX
                ELSE
                   DS = DSMAX
                ENDIF
           ENDIF
C
           GO TO 20
      ENDIF
C
           RETURN
           END
C*********************************************************************
C*                              TRLVCT                               *
C*********************************************************************
      SUBROUTINE TRLVCT(D,DD,DDO,DD1,DD2,DDP,AREA,EMOD,ELENG,
     $                  C1,C2,C3,Q,QT,SS,TT,X,ELONG,DEFLEN,F,FP,FPI,R,
     $                  JCODE,MAXA,MCODE,MINC,
     $                  IT,INCONV,ITECNT,ITENUM,ITEMAX,ITEUPD,NE,NEQ,
     $                  NJ,LSS,QIMAX,QI,DQI,TOLDIS,TOLFOR,
     $                  DQI1,DQII,NFE)
C
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
C
      DIMENSION D(*),DD(*),DDO(*),DD1(*),DD2(*),DDP(*),AREA(*),
     $          EMOD(*),ELENG(*),C1(*),C2(*),C3(*),Q(*),QT(*),
     $          SS(*),TT(*),X(3,*),ELONG(*),DEFLEN(*),
     $          F(*),FP(*),FPI(*),R(*),
     $          JCODE(3,*),MAXA(*),MCODE(6,*),MINC(2,*)
C
C
      ITENUM = ITEUPD
```

```
      IT = 1
      INCONV = 1
C
C---- PERFORM THE ITERATIONS UNTIL CONVERGENCE IS ACHIEVED OR THE
C     PRECRIBED MAXIMIUM NUMBER OF ITERATIONS IS EXCEEDED.
C
  10  IF(INCONV .NE. 0 .AND. IT .LE. ITEMAX)THEN
C
          DO 15 I = 1, NEQ
               FPI(I) = F(I)
 15       CONTINUE
C
C---- UPDATE THE LOAD VECTOR
          DO 20 I = 1, NEQ
             QT(I) = Q(I) * QI
 20       CONTINUE
C
C
C---- COMPUTE  UNBALANCED FORCES
          DO 25 I = 1, NEQ
              R(I) = QT(I) - F(I)
 25       CONTINUE
C
C---- COMPUTE THE TANGENT STIFFNESS MATRIX
          IF (ITENUM .GE. ITEUPD) THEN
             CALL STIFF(SS,AREA,EMOD,ELENG,C1,C2,C3,ELONG,DEFLEN,
     $                  MAXA,MCODE,NE,LSS)
             ITENUM = 0
          ENDIF
C
C---- SOLVE FOR DD1
          DO 30 I = 1, NEQ
              TT(I) = 0.D0
              TT(I) = Q(I)
 30       CONTINUE
C
          CALL SOLVE(SS,TT,MAXA,NEQ,1)
C
          DO 40 I = 1, NEQ
              DD1(I) = TT(I)
              TT(I) = 0.D0
 40       CONTINUE
C
C---- SOLVE FOR DD2
          DO 50 I = 1, NEQ
              TT(I) = 0.D0
              TT(I) = R(I)
 50       CONTINUE
C
          CALL SOLVE(SS,TT,MAXA,NEQ,2)
C
          DO 60 I = 1, NEQ
              DD2(I) = TT(I)
              TT(I) = 0.D0
 60       CONTINUE
C
C---- COMPUTE THE INCREMENT IN LOAD PARAMETER
          DQI = -(DOTPRD(DDO,DD2,NEQ))/(DOTPRD(DDO,DD1,NEQ)+DQI1)
          DQII = DQII + DQI
          QI = QI + DQI
C
C---- COMPUTE THE INCREMENTAL AND TOTAL DISPLACEMENTS
          DO 70 I = 1, NEQ
             DD(I) = DQI*DD1(I) + DD2(I)
             D(I) = D(I) + DD(I)
             DDP(I) = DDP(I) + DD(I)
 70       CONTINUE
C
C---- UPDATE THE JOINT COORDINATES
           CALL UPDATC(X,DD,C1,C2,C3,ELONG,ELENG,DEFLEN,MINC,JCODE,
```

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

```
C     DEFINED IN EQS. 5.15 (HOLZER, 1985)
C
      GAMMA = AREA(N)*EMOD(N)/ELENG(N)
C
      G(1) = (GAMMA)*(C1(N)**2)
      G(2) = (GAMMA)*(C2(N)**2)
      G(3) = (GAMMA)*(C3(N)**2)
      G(5) =  GAMMA*C2(N)*C3(N)
      G(6) =  GAMMA*C1(N)*C3(N)
C
      RETURN
      END
C*****************************************************************************
C*                              NELEMS                                       *
C*****************************************************************************
      SUBROUTINE NELEMS(AREA,EMOD,ELENG,C1,C2,C3,H,ELONG,DEFLEN,N)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION AREA(*),EMOD(*),ELENG(*),C1(*),C2(*),C3(*),H(6),
     $          ELONG(*),DEFLEN(*)
C
C     FOR ELEMENT N, COMPUTE THE NON-LINEAR GLOBAL STIFFNESS COEFFICIENTS,
C     H(6)
C
C
      GAMMA = AREA(N)*EMOD(N)/ELENG(N)
      P = ELONG(N)/DEFLEN(N)
C
      H(1) = GAMMA*P*(1-(C1(N)*C1(N)))
      H(2) = GAMMA*P*(1-(C2(N)*C2(N)))
      H(3) = GAMMA*P*(1-(C3(N)*C3(N)))
      H(4) = -GAMMA*C1(N)*C2(N)*P
      H(5) = -GAMMA*C2(N)*C3(N)*P
      H(6) = -GAMMA*C1(N)*C3(N)*P
C
      RETURN
      END
C*****************************************************************************
C*                              ASSEMS                                       *
C*****************************************************************************
      SUBROUTINE ASSEMS(SS,G,H,MCODE,MAXA,N,LSS)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION SS(*),G(*),H(*),MCODE(6,*),MAXA(*),INDEX(6,6)
      DATA INDEX/1,4,6,-1,-4,-6, 4,2,5,-4,-2,-5, 6,5,3,-6,-5,-3,
     $          -1,-4,-6,1,4,6, -4,-2,-5,4,2,5, -6,-5,-3,6,5,3/
C
C
C     INITIALIZE INDEX. ; ASSIGN STIFFNESS COEFFICIENTS, G(L),
C     OF ELEMENT N TO THE SYSTEM STIFFNESS MATRIX, SS, BY INDEX, MCODE,
C     AND MAXA.
C
      DO 20 JE = 1, 6
         J = MCODE(JE,N)
         IF (J .NE. 0) THEN
            DO 10 IE = 1, JE
               I = MCODE(IE,N)
               IF (I .NE. 0) THEN
                  IF (I .GT. J) THEN
                     K = MAXA(I) + I - J
                  ELSE
                     K = MAXA(J) + J - I
                  END IF
                  L = INDEX(IE,JE)
                  IF (L .GT. 0) THEN
                     SS(K) = SS(K) + G(L) + H(L)
                  ELSE
                     SS(K) = SS(K) - G(-L) - H(-L)
                  END IF
               END IF
   10       CONTINUE
         END IF
```

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

```
C
      RETURN
      END
C*********************************************************************
C*                              INTERF                               *
C*********************************************************************
      SUBROUTINE INTERF(F,AREA,EMOD,ELENG,C1,C2,C3,ELONG,MCODE,I)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION F(*),AREA(*),EMOD(*),ELENG(*),C1(*),C2(*),C3(*),
     $          ELONG(*),MCODE(6,*)
C
      GAMMA = AREA(I)*EMOD(I)/ELENG(I)
C
      DO 20 L = 1, 6
         K = MCODE(L,I)
         IF (K .NE. 0) THEN
             IF (L .EQ. 1) THEN
                 F(K) = -GAMMA*ELONG(I)*C1(I) + F(K)
             ELSEIF (L .EQ. 2) THEN
                 F(K) = -GAMMA*ELONG(I)*C2(I) + F(K)
             ELSEIF (L .EQ. 3) THEN
                 F(K) = -GAMMA*ELONG(I)*C3(I) + F(K)
             ELSEIF (L .EQ. 4) THEN
                 F(K) = GAMMA*ELONG(I)*C1(I) + F(K)
             ELSEIF (L .EQ. 5) THEN
                 F(K) = GAMMA*ELONG(I)*C2(I) + F(K)
             ELSEIF (L .EQ. 6) THEN
                 F(K) = GAMMA*ELONG(I)*C3(I) + F(K)
             ENDIF
         ENDIF
 20   CONTINUE
C
      RETURN
      END
C
C*********************************************************************
```

SAMPLE DATA FILE

THE DATA FILE USED FOR STRUCTURE 1 IS PRESENTED HERE.

```
09/07/91
2
2 3
1 2
2 3
1 1
1 2
1 3
2 3
3 1
3 2
3 3
0 0
0.D0 0.D0 0.D0
10.D0 1.D0 10.D0
20.D0 0.D0 20.D0
0.181 29000.0
0.181 29000.0
2 2 -2
0 0 0
1.0 0.00 0.10
1 6 3
0.0001 0.0001
1
2
0
```

LISTING OF PROGRAM 2

```
C
C   THIS PROGRAM TRACES THE EQUILIBRIUM PATH OF THE GEOMETRICALLY
C   NONLINEAR SPACE TRUSSES BY THE SPARSE NORMAL FLOW HOMOTOPY ALGORITHM
C   OF 'HOMPACK'.
C
C
C*********************************************************************
C*                          INPUT DATA                               *
C*********************************************************************
C     LIST-DIRECTED INPUT:
C     INPUT UNITS: KIP, INCH, RADIAN, FAHRENHEIT
C
C
C     1. ENTER NUMBER OF ELEMENTS AND NUMBER OF JOINTS (IN HMAIN)
C           NE, NJ
C
C     2. ENTER MEMBER INCIDENCES (IN STRUCT)
C           MINC(1,I), MINC(2,I)  I=1,NE
C
C     3. ENTER FOR EACH JOINT CONSTRAINT (IN STRUCT)
C           JNUM, JDIR
C           AFTER LAST JOINT CONSTRAINT ENTER
C           0, 0
C
C     4. ENTER JOINT COORDINATES FOR EACH JOINT(IN PROP)
C           X(1,J), X(2,J), X(3,J)   J=1,NJ
C
C     5. ENTER MEMBER PROPERTIES CROSS SECTIONAL AREA
C           AND ELASTIC MODULUS FOR EACH MEMBER (IN PROP)
C           AREA(I), EMOD(I)    I=1,NE
C
C     6. IF THERE ARE JOINT LOADS ENTER (IN JLOAD)
C              JNUM, JDIR, FORCE
C              AFTER LAST JOINT LOAD ENTER
C              0, 0, 0
C           ELSE ENTER
C              0, 0, 0
C           END IF
C
C     7. ENTER THE MAXIMUM LOAD STEP ALLOWED 'HMAX' (IN HMAIN)
C
C     8. ENTER THE RELATIVE ERROR TOLERANCE 'ARCRE' (IN HMAIN)
C
C     9. ENTER THE ABSOLUTE ERROR TOLERANCE 'ARCAE' (IN HMAIN)
C
C    10. ENTER THE DOF FOR WHICH DISPLACEMENT AND LAMBDA VALUES ARE TO
C        BE PRINTED AFTER EACH LOAD INCREMENT (IN RESULT)
C        END WITH 0
C
C*********************************************************************
C                          MAIN PROGRAM
C*********************************************************************
C
      PROGRAM HMAIN
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DOUBLE PRECISION Y(600),
     +  YP(600),YOLD(600),YPOLD(600),A(600),QR(31000),WORK(40000),
     +  SSPAR(8),PAR(1),PP(600),RHOVEC(600)
      DOUBLE PRECISION ARCRE,ARCAE,ANSRE,ANSAE,ARCLEN,TIMEA,TIMEB
      INTEGER PIVOT(601),IPAR(1)
      INTEGER IFLAG,J,LENQR,N,NFE,NP1,TRACE,LSTEP
C
      COMMON/NUM/NE,NEQ,NJ
      COMMON/LSYS/LSS
      COMMON/STRT/XC(3,217),AREA(624),EMOD(624),ELENG(624),
     $            C1(624),C2(624),C3(624)
```

```
      COMMON/UCR/XZ(3,217)
      COMMON/LODE/JCODE(3,217),MINC(2,624),MCODE(6,624)
      COMMON/KYL/MEXA(624),KHT(624)
      COMMON/CLOAD/QL(600)
      COMMON/FORC/ELONG(624),DEFLEN(624),FG(600)
      COMMON/STF/SS(31000)
C
      CHARACTER*6 NAME
      INTEGER TIME,CODE
      REAL DTIME
C
C CALL STRUCT AND LOAD
C
     READ(5,*) NE, NJ
      CALL STRUCT
      CALL LOAD
C
C INITIALIZE TIMER VARIABLES.
C
          CODE=2
          TIME=0
          DTIME=0.0
C
C DFEFINE ARGUMENTS FOR CALL TO HOMPACK PROCEDURE.
C
          N=NEQ
          LSTEP=1
          DO 7 J = 1, 8
 7           SSPAR(J)=0.0D0
C
      READ(5,*)HMAX
      READ(5,*)ARCRE
      READ(5,*)ARCAE
           SSPAR(5)=HMAX
        ANSRE=1.0D-12
        ANSAE-1.0D-12
C
     TRACE=0
          IFLAG=-2
          LENQR=LSS
          NP1=N+1
          DO 40 J=1,N
 40           Y(J)=0.D00
          DO 42 I = 1, NE
              ELONG(I) = 0.D0
              DEFLEN(I) = ELENG(I)
C             A(I) = 0.D0
 42       CONTINUE
C
C*********************************************************************
C INSERT CALL TO INITIALIZE SYSTEM TIMER HERE.
C*****     MODIFIED FOR VM1   **********
      CALL CPUTIME(TIMEA, IRCD_A)
C
C*********************************************************************
C
C CALL TO HOMPACK ROUTINE.
C
            NAME='FIXPNS'
            CALL FIXPNS(N,Y,IFLAG,ARCRE,ARCAE,ANSRE,ANSAE,TRACE,A,
     +        NFE,ARCLEN,YP,YOLD,YPOLD,QR,LENQR,PIVOT,WORK,
     +        SSPAR,PAR,IPAR,LSTEP)
C
C
C*********************************************************************
C INSERT CALL TO RETURN EXECUTION TIME IN SECONDS IN DTIME.
C *******  MODIFIED FOR VM1  **********
C
      CALL CPUTIME(TIMEB, IRCD_B)
      IF(IRCD_A .NE. 8 .AND. IRCD_B .EQ. 0)THEN
```

```
            DTIME = (TIMEB - TIMEA)
         ENDIF
C*************************************************************************
C
C          WRITE (6,45) NAME
C45        FORMAT(//,8X,'TESTING',1X,6A)
          WRITE(7,50) Y(NP1),IFLAG,NFE,ARCLEN
50        FORMAT(/' LAMBDA =',F11.8,'  FLAG =',I2,I8,' JACOBIAN ',
     +    'EVALUATIONS',/,1X,'  ARC LENGTH =',F8.3)
          WRITE(7,*) ' CPUTIME(MICROSECONDS) =',   DTIME
C
         STOP
         END
C
C*************************************************************************
C*                        SUBROUTINE RHO                                 *
C*************************************************************************
C
      SUBROUTINE RHO(A,LAMBDA,X,V,PAR,IPAR)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DOUBLE PRECISION A(*),LAMBDA,X(*),V(*),PAR(*)
      INTEGER IPAR(*)
C
        COMMON/NUM/NE,NEQ,NJ
        COMMON/LSYS/LSS
        COMMON/STRT/XC(3,217),AREA(624),EMOD(624),ELENG(624),
     $              C1(624),C2(624),C3(624)
        COMMON/LODE/JCODE(3,217),MINC(2,624),MCODE(6,624)
        COMMON/KYL/MEXA(624),KHT(624)
        COMMON/CLOAD/QL(600)
        COMMON/FORC/ELONG(624),DEFLEN(624),FG(600)
        COMMON/STF/SS(31000)
C
C PAR(1:*) AND IPAR(1:*) ARE ARRAYS FOR (OPTIONAL) USER PARAMETERS,
C    WHICH ARE SIMPLY PASSED THROUGH TO THE USER WRITTEN SUBROUTINES
C    RHO, RHOJAC.
C
C EVALUATE  RHO(A,LAMBDA,X)  AND RETURN IN THE VECTOR  V .
C
      N = NEQ
C
      CALL FORCES
C
      DO 10 I = 1, N
        V(I) = FG(I) - LAMBDA*QL(I)
 10   CONTINUE
C
      RETURN
      END
C
C*************************************************************************
C*                        SUBROUTINE RHOJS                               *
C*************************************************************************
C
      SUBROUTINE RHOJS(A,LAMBDA,X,QR,LENQR,PIVOT,PP,PAR,IPAR)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      INTEGER IPAR(*),LENQR,N,PIVOT(*)
      DOUBLE PRECISION A(*),LAMBDA,PAR(*),PP(*),QR(*),X(*)
C
        COMMON/NUM/NE,NEQ,NJ
        COMMON/LSYS/LSS
        COMMON/STRT/XC(3,217),AREA(624),EMOD(624),ELENG(624),
     $              C1(624),C2(624),C3(624)
        COMMON/LODE/JCODE(3,217),MINC(2,624),MCODE(6,624)
        COMMON/KYL/MEXA(624),KHT(624)
        COMMON/CLOAD/QL(600)
        COMMON/FORC/ELONG(624),DEFLEN(624),FG(600)
        COMMON/STF/SS(31000)
C
C PAR(1:*) AND IPAR(1:*) ARE ARRAYS FOR (OPTIONAL) USER PARAMETERS,
```

```
C    WHICH ARE SIMPLY PASSED THROUGH TO THE USER WRITTEN SUBROUTINES
C    RHOA, RHOJS.
C
C Evaluate the N X N symmetric Jacobian matrix [D RHO/DX] at (A,X,LAMBDA),
C and return the result in packed skyline storage format in QR.  LENQR is
C the length of QR, and PIVOT contains the indices of the diagonal elements
C of [D RHO/DX] within QR.  PP contains -[D RHO/D LAMBDA] evaluated at
C (A,X,LAMBDA).  Note the minus sign in the definition of PP.
C
C
      CALL UPDATC(X)
C
       N = NEQ
       CALL STIFF
       DO 10 I = 1, LSS
            QR(I) = SS(I)
 10    CONTINUE
C
       DO 20 I = 1, N+1
            PIVOT(I) = MEXA(I)
 20    CONTINUE
C
       DO 30 I = 1, N
             PP(I) =  QL(I)
 30    CONTINUE
C
      RETURN
      END
C
C*********************************************************************
C*                              UPDATC                               *
C*********************************************************************
      SUBROUTINE UPDATC(X)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION X(*)
C
        COMMON/NUM/NE,NEQ,NJ
        COMMON/LSYS/LSS
        COMMON/STRT/XC(3,217),AREA(624),EMOD(624),ELENG(624),
     $              C1(624),C2(624),C3(624)
        COMMON/UCR/XZ(3,217)
        COMMON/LODE/JCODE(3,217),MINC(2,624),MCODE(6,624)
        COMMON/KYL/MEXA(624),KHT(624)
        COMMON/FORC/ELONG(624),DEFLEN(624),FG(600)
        COMMON/STF/SS(31000)
C
      DO 30 J = 1, NJ
         DO 20 L = 1, 3
            K = JCODE(L,J)
            IF(K.NE.0) THEN
                XZ(L,J) = XC(L,J) + X(K)
            ELSE
                XZ(L,J) = XC(L,J)
            ENDIF
 20         CONTINUE
 30   CONTINUE
C
      DO 40 I = 1, NE
C
         J = MINC(1,I)
         K = MINC(2,I)
C
         EL1 = XZ(1,K)-XZ(1,J)
         EL2 = XZ(2,K)-XZ(2,J)
         EL3 = XZ(3,K)-XZ(3,J)
         DEFLEN(I) = DSQRT((EL1**2)+(EL2**2)+(EL3**2))
         ELONG(I) =  DEFLEN(I) - ELENG(I)
         C1(I) = EL1/DEFLEN(I)
         C2(I) = EL2/DEFLEN(I)
         C3(I) = EL3/DEFLEN(I)
```

```
 40   CONTINUE
C
      RETURN
      END
C
C*************************************************************************
```

SAMPLE DATA FILE

THE DATA FILE USED FOR STRUCTURE 1 IS PRESENTED HERE.

```
2 3
1 2
2 3
1 1
1 2
1 3
2 3
3 1
3 2
3 3
0 0
0.D0 0.D0 0.D0
10.D0 1.D0 10.D0
20.D0 0.D0 20.D0
0.181 29000.0
0.181 29000.0
2 2 -2.0
0 0 0
0.1
1.0D-04
1.0D-04
1
2
0
```

The vita has been removed from the scanned document